

activemq-cpp-3.4.1

Generated by Doxygen 1.7.6.1

Tue Feb 28 2012 17:47:31

Contents

1	Namespace Index	1
1.1	Namespace List	1
2	Data Structure Index	3
2.1	Class Hierarchy	3
3	Data Structure Index	21
3.1	Data Structures	21
4	File Index	51
4.1	File List	51
5	Namespace Documentation	65
5.1	activemq Namespace Reference	65
5.1.1	Detailed Description	65
5.2	activemq::cmsutil Namespace Reference	66
5.3	activemq::commands Namespace Reference	67
5.4	activemq::core Namespace Reference	68
5.5	activemq::core::policies Namespace Reference	69
5.6	activemq::exceptions Namespace Reference	70
5.7	activemq::io Namespace Reference	70
5.8	activemq::library Namespace Reference	70
5.9	activemq::state Namespace Reference	70
5.10	activemq::threads Namespace Reference	71
5.11	activemq::transport Namespace Reference	71
5.12	activemq::transport::correlator Namespace Reference	72
5.13	activemq::transport::failover Namespace Reference	72

5.14	activemq::transport::inactivity Namespace Reference	73
5.15	activemq::transport::logging Namespace Reference	73
5.16	activemq::transport::mock Namespace Reference	73
5.17	activemq::transport::tcp Namespace Reference	73
5.18	activemq::util Namespace Reference	74
5.19	activemq::wireformat Namespace Reference	75
5.20	activemq::wireformat::openwire Namespace Reference	75
5.21	activemq::wireformat::openwire::marshal Namespace Reference	75
5.22	activemq::wireformat::openwire::marshal::generated Namespace Reference	76
5.23	activemq::wireformat::openwire::utils Namespace Reference	79
5.24	activemq::wireformat::stomp Namespace Reference	79
5.25	cms Namespace Reference	80
5.25.1	Detailed Description	83
5.26	decaf Namespace Reference	83
5.26.1	Detailed Description	83
5.27	decaf::internal Namespace Reference	83
5.28	decaf::internal::io Namespace Reference	84
5.29	decaf::internal::net Namespace Reference	84
5.30	decaf::internal::net::ssl Namespace Reference	85
5.31	decaf::internal::net::ssl::openssl Namespace Reference	85
5.32	decaf::internal::net::tcp Namespace Reference	86
5.33	decaf::internal::nio Namespace Reference	86
5.34	decaf::internal::security Namespace Reference	87
5.35	decaf::internal::util Namespace Reference	87
5.36	decaf::internal::util::concurrent Namespace Reference	87
5.37	decaf::io Namespace Reference	88
5.38	decaf::lang Namespace Reference	89
5.38.1	Function Documentation	91
5.38.1.1	operator!=	91
5.38.1.2	operator!=	91
5.38.1.3	operator!=	91
5.38.1.4	operator!=	91
5.38.1.5	operator==	91

5.38.1.6	operator==	92
5.38.1.7	operator==	92
5.38.1.8	operator==	92
5.39	decaf::lang::exceptions Namespace Reference	92
5.40	decaf::net Namespace Reference	92
5.41	decaf::net::ssl Namespace Reference	94
5.42	decaf::nio Namespace Reference	94
5.43	decaf::security Namespace Reference	95
5.44	decaf::security::auth Namespace Reference	95
5.45	decaf::security::auth::x500 Namespace Reference	95
5.46	decaf::security::cert Namespace Reference	96
5.47	decaf::util Namespace Reference	96
5.48	decaf::util::comparators Namespace Reference	98
5.49	decaf::util::concurrent Namespace Reference	98
5.50	decaf::util::concurrent::atomic Namespace Reference	100
5.51	decaf::util::concurrent::locks Namespace Reference	100
5.52	decaf::util::logging Namespace Reference	101
5.52.1	Enumeration Type Documentation	102
5.52.1.1	Levels	102
5.53	decaf::util::zip Namespace Reference	102
5.54	std Namespace Reference	103
6	Data Structure Documentation	105
6.1	decaf::util::concurrent::ThreadPoolExecutor::AbortPolicy Class Reference	105
6.1.1	Detailed Description	105
6.1.2	Constructor & Destructor Documentation	105
6.1.2.1	AbortPolicy	106
6.1.2.2	~AbortPolicy	106
6.1.3	Member Function Documentation	106
6.1.3.1	rejectedExecution	106
6.2	decaf::util::AbstractCollection< E > Class Template Reference	106
6.2.1	Detailed Description	109
6.2.2	Constructor & Destructor Documentation	110
6.2.2.1	AbstractCollection	110

6.2.2.2	~AbstractCollection	110
6.2.3	Member Function Documentation	110
6.2.3.1	add	110
6.2.3.2	addAll	110
6.2.3.3	clear	111
6.2.3.4	contains	112
6.2.3.5	containsAll	113
6.2.3.6	copy	113
6.2.3.7	equals	114
6.2.3.8	isEmpty	114
6.2.3.9	lock	115
6.2.3.10	notify	115
6.2.3.11	notifyAll	116
6.2.3.12	operator=	116
6.2.3.13	remove	116
6.2.3.14	removeAll	117
6.2.3.15	retainAll	118
6.2.3.16	toArray	119
6.2.3.17	tryLock	119
6.2.3.18	unlock	120
6.2.3.19	wait	120
6.2.3.20	wait	120
6.2.3.21	wait	121
6.2.4	Field Documentation	121
6.2.4.1	mutex	121
6.3	decaf::util::concurrent::AbstractExecutorService Class Reference	122
6.3.1	Detailed Description	122
6.3.2	Constructor & Destructor Documentation	122
6.3.2.1	AbstractExecutorService	122
6.3.2.2	~AbstractExecutorService	122
6.4	decaf::util::AbstractList< E > Class Template Reference	123
6.4.1	Detailed Description	124
6.4.2	Constructor & Destructor Documentation	124
6.4.2.1	AbstractList	124

6.4.2.2	~AbstractList	124
6.4.3	Member Function Documentation	125
6.4.3.1	add	125
6.4.3.2	add	126
6.4.3.3	addAll	126
6.4.3.4	clear	127
6.4.3.5	indexOf	128
6.4.3.6	iterator	128
6.4.3.7	iterator	129
6.4.3.8	lastIndexOf	129
6.4.3.9	listIterator	130
6.4.3.10	listIterator	131
6.4.3.11	listIterator	131
6.4.3.12	listIterator	132
6.4.3.13	removeAt	133
6.4.3.14	removeRange	133
6.4.3.15	set	133
6.4.4	Field Documentation	134
6.4.4.1	modCount	134
6.5	decaf::util::AbstractMap< K, V, COMPARATOR > Class Template - Reference	134
6.5.1	Detailed Description	134
6.5.2	Constructor & Destructor Documentation	135
6.5.2.1	~AbstractMap	135
6.6	decaf::util::concurrent::locks::AbstractOwnableSynchronizer Class - Reference	135
6.6.1	Detailed Description	135
6.6.2	Constructor & Destructor Documentation	136
6.6.2.1	~AbstractOwnableSynchronizer	136
6.6.2.2	AbstractOwnableSynchronizer	136
6.6.3	Member Function Documentation	136
6.6.3.1	getExclusiveOwnerThread	136
6.6.3.2	setExclusiveOwnerThread	136
6.7	decaf::util::AbstractQueue< E > Class Template Reference	136

6.7.1	Detailed Description	138
6.7.2	Constructor & Destructor Documentation	139
6.7.2.1	AbstractQueue	139
6.7.2.2	~AbstractQueue	139
6.7.3	Member Function Documentation	139
6.7.3.1	add	139
6.7.3.2	addAll	140
6.7.3.3	clear	141
6.7.3.4	element	142
6.7.3.5	remove	142
6.8	decaf::util::AbstractSequentialList< E > Class Template Reference . . .	143
6.8.1	Detailed Description	145
6.8.2	Constructor & Destructor Documentation	146
6.8.2.1	~AbstractSequentialList	146
6.8.3	Member Function Documentation	146
6.8.3.1	add	146
6.8.3.2	addAll	147
6.8.3.3	get	148
6.8.3.4	iterator	148
6.8.3.5	iterator	149
6.8.3.6	listIterator	149
6.8.3.7	listIterator	149
6.8.3.8	listIterator	149
6.8.3.9	listIterator	150
6.8.3.10	removeAt	150
6.8.3.11	set	151
6.9	decaf::util::AbstractSet< E > Class Template Reference	152
6.9.1	Detailed Description	153
6.9.2	Constructor & Destructor Documentation	153
6.9.2.1	~AbstractSet	153
6.9.3	Member Function Documentation	153
6.9.3.1	removeAll	153
6.10	activemq::transport::AbstractTransportFactory Class Reference	154
6.10.1	Detailed Description	155

6.10.2	Constructor & Destructor Documentation	155
6.10.2.1	~AbstractTransportFactory	155
6.10.3	Member Function Documentation	155
6.10.3.1	createWireFormat	155
6.11	activemq::core::ActiveMQAckHandler Class Reference	156
6.11.1	Detailed Description	156
6.11.2	Constructor & Destructor Documentation	156
6.11.2.1	~ActiveMQAckHandler	156
6.11.3	Member Function Documentation	156
6.11.3.1	acknowledgeMessage	156
6.12	activemq::commands::ActiveMQBlobMessage Class Reference	157
6.12.1	Constructor & Destructor Documentation	158
6.12.1.1	ActiveMQBlobMessage	158
6.12.1.2	~ActiveMQBlobMessage	158
6.12.2	Member Function Documentation	158
6.12.2.1	clone	158
6.12.2.2	cloneDataStructure	158
6.12.2.3	copyDataStructure	158
6.12.2.4	equals	159
6.12.2.5	getDataStructureType	159
6.12.2.6	getMimeType	159
6.12.2.7	getName	159
6.12.2.8	getRemoteBlobUrl	160
6.12.2.9	isDeletedByBroker	160
6.12.2.10	setDeletedByBroker	160
6.12.2.11	setMimeType	160
6.12.2.12	setName	160
6.12.2.13	setRemoteBlobUrl	161
6.12.2.14	toString	161
6.12.3	Field Documentation	161
6.12.3.1	BINARY_MIME_TYPE	161
6.12.3.2	ID_ACTIVEMQBLOBMESSAGE	161
6.13	activemq::wireformat::openwire::marshal::generated::ActiveMQBlob- MessageMarshaller Class Reference	161

6.13.1	Detailed Description	162
6.13.2	Constructor & Destructor Documentation	162
6.13.2.1	ActiveMQBlobMessageMarshaller	162
6.13.2.2	~ActiveMQBlobMessageMarshaller	163
6.13.3	Member Function Documentation	163
6.13.3.1	createObject	163
6.13.3.2	getDataStructureType	163
6.13.3.3	looseMarshal	163
6.13.3.4	looseUnmarshal	164
6.13.3.5	tightMarshal1	164
6.13.3.6	tightMarshal2	165
6.13.3.7	tightUnmarshal	165
6.14	activemq::commands::ActiveMQBytesMessage Class Reference	166
6.14.1	Constructor & Destructor Documentation	168
6.14.1.1	ActiveMQBytesMessage	168
6.14.1.2	~ActiveMQBytesMessage	168
6.14.2	Member Function Documentation	168
6.14.2.1	clearBody	168
6.14.2.2	clone	168
6.14.2.3	cloneDataStructure	169
6.14.2.4	copyDataStructure	169
6.14.2.5	equals	169
6.14.2.6	getBodyBytes	170
6.14.2.7	getBodyLength	170
6.14.2.8	getDataStructureType	170
6.14.2.9	onSend	171
6.14.2.10	readBoolean	171
6.14.2.11	readByte	171
6.14.2.12	readBytes	172
6.14.2.13	readBytes	172
6.14.2.14	readChar	173
6.14.2.15	readDouble	173
6.14.2.16	readFloat	174
6.14.2.17	readInt	174

6.14.2.18	readLong	175
6.14.2.19	readShort	175
6.14.2.20	readString	176
6.14.2.21	readUnsignedShort	176
6.14.2.22	readUTF	176
6.14.2.23	reset	177
6.14.2.24	setBodyBytes	177
6.14.2.25	toString	178
6.14.2.26	writeBoolean	178
6.14.2.27	writeByte	178
6.14.2.28	writeBytes	179
6.14.2.29	writeBytes	179
6.14.2.30	writeChar	179
6.14.2.31	writeDouble	180
6.14.2.32	writeFloat	180
6.14.2.33	writeInt	181
6.14.2.34	writeLong	181
6.14.2.35	writeShort	181
6.14.2.36	writeString	182
6.14.2.37	writeUnsignedShort	182
6.14.2.38	writeUTF	183
6.14.3	Field Documentation	183
6.14.3.1	ID_ACTIVEMQBYTESMESSAGE	183
6.15	activemq::wireformat::openwire::marshal::generated::ActiveMQBytes- MessageMarshaller Class Reference	183
6.15.1	Detailed Description	184
6.15.2	Constructor & Destructor Documentation	184
6.15.2.1	ActiveMQBytesMessageMarshaller	184
6.15.2.2	~ActiveMQBytesMessageMarshaller	184
6.15.3	Member Function Documentation	184
6.15.3.1	createObject	184
6.15.3.2	getDataStructureType	185
6.15.3.3	looseMarshal	185
6.15.3.4	looseUnmarshal	185

6.15.3.5	tightMarshal1	186
6.15.3.6	tightMarshal2	186
6.15.3.7	tightUnmarshal	187
6.16	activemq::core::ActiveMQConnection Class Reference	187
6.16.1	Detailed Description	193
6.16.2	Constructor & Destructor Documentation	193
6.16.2.1	ActiveMQConnection	193
6.16.2.2	~ActiveMQConnection	193
6.16.3	Member Function Documentation	193
6.16.3.1	addDispatcher	194
6.16.3.2	addProducer	194
6.16.3.3	addSession	194
6.16.3.4	addTransportListener	195
6.16.3.5	checkClosed	195
6.16.3.6	checkClosedOrFailed	195
6.16.3.7	cleanup	195
6.16.3.8	close	195
6.16.3.9	createSession	196
6.16.3.10	createSession	196
6.16.3.11	destroyDestination	196
6.16.3.12	destroyDestination	197
6.16.3.13	disconnect	197
6.16.3.14	ensureConnectionInfoSent	197
6.16.3.15	fire	197
6.16.3.16	getBrokerURL	198
6.16.3.17	getClientID	198
6.16.3.18	getCloseTimeout	198
6.16.3.19	getCompressionLevel	198
6.16.3.20	getConnectionId	199
6.16.3.21	getConnectionInfo	199
6.16.3.22	getExceptionListener	199
6.16.3.23	getFirstFailureError	199
6.16.3.24	getMetaData	200
6.16.3.25	getNextLocalTransactionId	200

6.16.3.26 getNextSessionId	200
6.16.3.27 getNextTempDestinationId	200
6.16.3.28 getPassword	201
6.16.3.29 getPrefetchPolicy	201
6.16.3.30 getProducerWindowSize	201
6.16.3.31 getProperties	201
6.16.3.32 getRedeliveryPolicy	201
6.16.3.33 getResourceManagerId	202
6.16.3.34 getScheduler	202
6.16.3.35 getSendTimeout	202
6.16.3.36 getTransport	202
6.16.3.37 getUsername	203
6.16.3.38 isAlwaysSyncSend	203
6.16.3.39 isClosed	203
6.16.3.40 isDispatchAsync	203
6.16.3.41 isMessagePrioritySupported	203
6.16.3.42 isStarted	203
6.16.3.43 isTransportFailed	204
6.16.3.44 isUseAsyncSend	204
6.16.3.45 isUseCompression	204
6.16.3.46 onAsyncException	204
6.16.3.47 onCommand	204
6.16.3.48 oneway	205
6.16.3.49 onException	205
6.16.3.50 removeDispatcher	205
6.16.3.51 removeProducer	205
6.16.3.52 removeSession	206
6.16.3.53 removeTransportListener	206
6.16.3.54 sendPullRequest	206
6.16.3.55 setAlwaysSyncSend	207
6.16.3.56 setBrokerURL	207
6.16.3.57 setClientId	207
6.16.3.58 setCloseTimeout	208
6.16.3.59 setCompressionLevel	208

6.16.3.60	setDefaultClientId	208
6.16.3.61	setDispatchAsync	209
6.16.3.62	setExceptionListener	209
6.16.3.63	setMessagePrioritySupported	209
6.16.3.64	setPassword	209
6.16.3.65	setPrefetchPolicy	210
6.16.3.66	setProducerWindowSize	210
6.16.3.67	setRedeliveryPolicy	210
6.16.3.68	setSendTimeout	210
6.16.3.69	setTransportInterruptionProcessingComplete	211
6.16.3.70	setUseAsyncSend	211
6.16.3.71	setUseCompression	211
6.16.3.72	setUsername	211
6.16.3.73	signalInterruptionProcessingComplete	211
6.16.3.74	start	211
6.16.3.75	stop	212
6.16.3.76	syncRequest	212
6.16.3.77	transportInterrupted	212
6.16.3.78	transportResumed	213
6.16.3.79	waitForTransportInterruptionProcessingToComplete	213
6.17	activemq::core::ActiveMQConnectionFactory Class Reference	213
6.17.1	Constructor & Destructor Documentation	216
6.17.1.1	ActiveMQConnectionFactory	216
6.17.1.2	ActiveMQConnectionFactory	216
6.17.1.3	ActiveMQConnectionFactory	216
6.17.1.4	~ActiveMQConnectionFactory	216
6.17.2	Member Function Documentation	216
6.17.2.1	createActiveMQConnection	217
6.17.2.2	createConnection	217
6.17.2.3	createConnection	217
6.17.2.4	createConnection	218
6.17.2.5	createConnection	219
6.17.2.6	getBrokerURI	219
6.17.2.7	getClientId	219

6.17.2.8	getCloseTimeout	219
6.17.2.9	getCompressionLevel	220
6.17.2.10	getExceptionListener	220
6.17.2.11	getPassword	220
6.17.2.12	getPrefetchPolicy	220
6.17.2.13	getProducerWindowSize	220
6.17.2.14	getRedeliveryPolicy	221
6.17.2.15	getSendTimeout	221
6.17.2.16	getUsername	221
6.17.2.17	isAlwaysSyncSend	221
6.17.2.18	isDispatchAsync	222
6.17.2.19	isMessagePrioritySupported	222
6.17.2.20	isUseAsyncSend	222
6.17.2.21	isUseCompression	222
6.17.2.22	setAlwaysSyncSend	222
6.17.2.23	setBrokerURI	223
6.17.2.24	setBrokerURI	223
6.17.2.25	setClientId	223
6.17.2.26	setCloseTimeout	223
6.17.2.27	setCompressionLevel	223
6.17.2.28	setDispatchAsync	224
6.17.2.29	setExceptionListener	224
6.17.2.30	setMessagePrioritySupported	224
6.17.2.31	setPassword	224
6.17.2.32	setPrefetchPolicy	225
6.17.2.33	setProducerWindowSize	225
6.17.2.34	setRedeliveryPolicy	225
6.17.2.35	setSendTimeout	226
6.17.2.36	setUseAsyncSend	226
6.17.2.37	setUseCompression	226
6.17.2.38	setUsername	226
6.17.3	Field Documentation	226
6.17.3.1	DEFAULT_URI	226
6.18	activemq::core::ActiveMQConnectionMetaData Class Reference	227

6.18.1	Detailed Description	227
6.18.2	Constructor & Destructor Documentation	228
6.18.2.1	ActiveMQConnectionMetaData	228
6.18.2.2	~ActiveMQConnectionMetaData	228
6.18.3	Member Function Documentation	228
6.18.3.1	getCMSMajorVersion	228
6.18.3.2	getCMSMinorVersion	228
6.18.3.3	getCMSProviderName	228
6.18.3.4	getCMSVersion	229
6.18.3.5	getCMSXPropertyNames	229
6.18.3.6	getProviderMajorVersion	230
6.18.3.7	getProviderMinorVersion	230
6.18.3.8	getProviderVersion	230
6.19	activemq::core::ActiveMQConstants Class Reference	231
6.19.1	Detailed Description	232
6.19.2	Member Enumeration Documentation	232
6.19.2.1	AckType	232
6.19.2.2	DestinationActions	232
6.19.2.3	DestinationOption	232
6.19.2.4	TransactionState	233
6.19.2.5	URIParam	233
6.19.3	Member Function Documentation	233
6.19.3.1	toDestinationOption	233
6.19.3.2	toString	233
6.19.3.3	toString	233
6.19.3.4	toURIOption	234
6.20	activemq::core::ActiveMQConsumer Class Reference	234
6.20.1	Constructor & Destructor Documentation	236
6.20.1.1	ActiveMQConsumer	236
6.20.1.2	~ActiveMQConsumer	236
6.20.2	Member Function Documentation	236
6.20.2.1	acknowledge	236
6.20.2.2	acknowledge	236
6.20.2.3	afterMessageIsConsumed	237

6.20.2.4	beforeMessagesConsumed	237
6.20.2.5	clearMessagesInProgress	237
6.20.2.6	close	237
6.20.2.7	commit	237
6.20.2.8	deliverAcks	238
6.20.2.9	dequeue	238
6.20.2.10	dispatch	238
6.20.2.11	dispose	238
6.20.2.12	doClose	239
6.20.2.13	getConsumerId	239
6.20.2.14	getConsumerInfo	239
6.20.2.15	getFailureError	239
6.20.2.16	getLastDeliveredSequenceId	240
6.20.2.17	getMessageAvailableCount	240
6.20.2.18	getMessageListener	240
6.20.2.19	getMessageSelector	240
6.20.2.20	getRedeliveryPolicy	241
6.20.2.21	inProgressClearRequired	241
6.20.2.22	isClosed	241
6.20.2.23	isSynchronizationRegistered	241
6.20.2.24	iterate	241
6.20.2.25	receive	241
6.20.2.26	receive	242
6.20.2.27	receiveNoWait	242
6.20.2.28	rollback	242
6.20.2.29	setFailureError	243
6.20.2.30	setLastDeliveredSequenceId	243
6.20.2.31	setMessageListener	243
6.20.2.32	setRedeliveryPolicy	243
6.20.2.33	setSynchronizationRegistered	244
6.20.2.34	start	244
6.20.2.35	stop	244
6.21	activemq::library::ActiveMQCPP Class Reference	244
6.21.1	Constructor & Destructor Documentation	245

6.21.1.1	ActiveMQCPP	245
6.21.1.2	ActiveMQCPP	245
6.21.1.3	~ActiveMQCPP	245
6.21.2	Member Function Documentation	245
6.21.2.1	initializeLibrary	245
6.21.2.2	initializeLibrary	245
6.21.2.3	operator=	246
6.21.2.4	shutdownLibrary	246
6.22	activemq::commands::ActiveMQDestination Class Reference	246
6.22.1	Constructor & Destructor Documentation	249
6.22.1.1	ActiveMQDestination	249
6.22.1.2	ActiveMQDestination	249
6.22.1.3	~ActiveMQDestination	249
6.22.2	Member Function Documentation	249
6.22.2.1	cloneDataStructure	249
6.22.2.2	copyDataStructure	249
6.22.2.3	createDestination	250
6.22.2.4	createTemporaryName	250
6.22.2.5	equals	250
6.22.2.6	getClientId	251
6.22.2.7	getCMSDestination	251
6.22.2.8	getDataStructureType	251
6.22.2.9	getDestinationType	251
6.22.2.10	getOptions	252
6.22.2.11	getOrderedTarget	252
6.22.2.12	getPhysicalName	252
6.22.2.13	getPhysicalName	252
6.22.2.14	isAdvisory	252
6.22.2.15	isComposite	253
6.22.2.16	isConnectionAdvisory	253
6.22.2.17	isConsumerAdvisory	253
6.22.2.18	isExclusive	253
6.22.2.19	isOrdered	253
6.22.2.20	isProducerAdvisory	253

6.22.2.21	isQueue	254
6.22.2.22	isTemporary	254
6.22.2.23	isTopic	254
6.22.2.24	isWildcard	254
6.22.2.25	setAdvisory	254
6.22.2.26	setExclusive	255
6.22.2.27	setOrdered	255
6.22.2.28	setOrderedTarget	255
6.22.2.29	setPhysicalName	255
6.22.2.30	toString	255
6.22.3	Field Documentation	256
6.22.3.1	advisory	256
6.22.3.2	ADVISORY_PREFIX	256
6.22.3.3	COMPOSITE_SEPARATOR	256
6.22.3.4	CONNECTION_ADVISORY_PREFIX	256
6.22.3.5	CONSUMER_ADVISORY_PREFIX	256
6.22.3.6	DEFAULT_ORDERED_TARGET	256
6.22.3.7	exclusive	256
6.22.3.8	ID_ACTIVEMQDESTINATION	256
6.22.3.9	options	257
6.22.3.10	ordered	257
6.22.3.11	orderedTarget	257
6.22.3.12	physicalName	257
6.22.3.13	PRODUCER_ADVISORY_PREFIX	257
6.22.3.14	QUEUE_QUALIFIED_PREFIX	257
6.22.3.15	TEMP_POSTFIX	257
6.22.3.16	TEMP_PREFIX	257
6.22.3.17	TEMP_QUEUE_QUALIFIED_PREFIX	257
6.22.3.18	TEMP_TOPIC_QUALIFIED_PREFIX	257
6.22.3.19	TOPIC_QUALIFIED_PREFIX	257
6.23	activemq::wireformat::openwire::marshal::generated::ActiveMQDestination- Marshaller Class Reference	258
6.23.1	Detailed Description	258
6.23.2	Constructor & Destructor Documentation	258

6.23.2.1	ActiveMQDestinationMarshaller	259
6.23.2.2	~ActiveMQDestinationMarshaller	259
6.23.3	Member Function Documentation	259
6.23.3.1	looseMarshal	259
6.23.3.2	looseUnmarshal	259
6.23.3.3	tightMarshal1	260
6.23.3.4	tightMarshal2	260
6.23.3.5	tightUnmarshal	261
6.24	activemq::exceptions::ActiveMQException Class Reference	262
6.24.1	Constructor & Destructor Documentation	262
6.24.1.1	ActiveMQException	262
6.24.1.2	ActiveMQException	262
6.24.1.3	ActiveMQException	263
6.24.1.4	ActiveMQException	263
6.24.1.5	~ActiveMQException	263
6.24.2	Member Function Documentation	263
6.24.2.1	clone	263
6.24.2.2	convertToCMSException	264
6.25	activemq::commands::ActiveMQMapMessage Class Reference	264
6.25.1	Constructor & Destructor Documentation	270
6.25.1.1	ActiveMQMapMessage	270
6.25.1.2	~ActiveMQMapMessage	271
6.25.2	Member Function Documentation	271
6.25.2.1	beforeMarshal	271
6.25.2.2	checkMapsUnmarshalled	271
6.25.2.3	clearBody	271
6.25.2.4	clone	272
6.25.2.5	cloneDataStructure	272
6.25.2.6	copyDataStructure	272
6.25.2.7	equals	272
6.25.2.8	getBoolean	273
6.25.2.9	getByte	273
6.25.2.10	getBytes	274
6.25.2.11	getChar	274

6.25.2.12	getDataStructureType	274
6.25.2.13	getDouble	275
6.25.2.14	getFloat	275
6.25.2.15	getInt	276
6.25.2.16	getLong	276
6.25.2.17	getMap	276
6.25.2.18	getMap	277
6.25.2.19	getMapNames	277
6.25.2.20	getShort	277
6.25.2.21	getString	278
6.25.2.22	isEmpty	278
6.25.2.23	isMarshalAware	278
6.25.2.24	itemExists	279
6.25.2.25	setBoolean	279
6.25.2.26	setByte	279
6.25.2.27	setBytes	280
6.25.2.28	setChar	280
6.25.2.29	setDouble	281
6.25.2.30	setFloat	281
6.25.2.31	setInt	281
6.25.2.32	setLong	282
6.25.2.33	setShort	282
6.25.2.34	setString	283
6.25.2.35	toString	283
6.25.3	Field Documentation	283
6.25.3.1	ID_ACTIVEMQMAPMESSAGE	284
6.26	activemq::wireformat::openwire::marshal::generated::ActiveMQMap- MessageMarshaller Class Reference	284
6.26.1	Detailed Description	285
6.26.2	Constructor & Destructor Documentation	285
6.26.2.1	ActiveMQMapMessageMarshaller	285
6.26.2.2	~ActiveMQMapMessageMarshaller	285
6.26.3	Member Function Documentation	285
6.26.3.1	createObject	285

6.26.3.2	getDataStructureType	285
6.26.3.3	looseMarshal	286
6.26.3.4	looseUnmarshal	286
6.26.3.5	tightMarshal1	286
6.26.3.6	tightMarshal2	287
6.26.3.7	tightUnmarshal	287
6.27	activemq::commands::ActiveMQMessage Class Reference	288
6.27.1	Constructor & Destructor Documentation	289
6.27.1.1	ActiveMQMessage	289
6.27.1.2	~ActiveMQMessage	289
6.27.2	Member Function Documentation	289
6.27.2.1	clone	289
6.27.2.2	cloneDataStructure	289
6.27.2.3	copyDataStructure	289
6.27.2.4	equals	290
6.27.2.5	getDataStructureType	290
6.27.2.6	toString	290
6.27.3	Field Documentation	290
6.27.3.1	ID_ACTIVEMQMESSAGE	290
6.28	activemq::wireformat::openwire::marshal::generated::ActiveMQMessage- Marshaller Class Reference	291
6.28.1	Detailed Description	291
6.28.2	Constructor & Destructor Documentation	292
6.28.2.1	ActiveMQMessageMarshaller	292
6.28.2.2	~ActiveMQMessageMarshaller	292
6.28.3	Member Function Documentation	292
6.28.3.1	createObject	292
6.28.3.2	getDataStructureType	292
6.28.3.3	looseMarshal	292
6.28.3.4	looseUnmarshal	293
6.28.3.5	tightMarshal1	293
6.28.3.6	tightMarshal2	294
6.28.3.7	tightUnmarshal	294

6.29	activemq::commands::ActiveMQMessageTemplate< T > Class - Template Reference	295
6.29.1	Constructor & Destructor Documentation	296
6.29.1.1	ActiveMQMessageTemplate	296
6.29.1.2	~ActiveMQMessageTemplate	296
6.29.2	Member Function Documentation	296
6.29.2.1	acknowledge	296
6.29.2.2	clearBody	296
6.29.2.3	clearProperties	297
6.29.2.4	equals	297
6.29.2.5	failIfReadOnlyBody	297
6.29.2.6	failIfReadOnlyProperties	297
6.29.2.7	failIfWriteOnlyBody	297
6.29.2.8	getBooleanProperty	297
6.29.2.9	getByteProperty	297
6.29.2.10	getCMSCorrelationID	297
6.29.2.11	getCMSDeliveryMode	298
6.29.2.12	getCMSDestination	298
6.29.2.13	getCMSExpiration	298
6.29.2.14	getCMSMessageID	298
6.29.2.15	getCMSPriority	298
6.29.2.16	getCMSRedelivered	298
6.29.2.17	getCMSReplyTo	298
6.29.2.18	getCMSTimestamp	298
6.29.2.19	getCMSType	298
6.29.2.20	getDoubleProperty	298
6.29.2.21	getFloatProperty	298
6.29.2.22	getIntProperty	298
6.29.2.23	getLongProperty	299
6.29.2.24	getPropertyNames	299
6.29.2.25	getShortProperty	299
6.29.2.26	getStringProperty	299
6.29.2.27	onSend	299
6.29.2.28	propertyExists	299

6.29.2.29	setBooleanProperty	299
6.29.2.30	setByteProperty	299
6.29.2.31	setCMSCorrelationID	299
6.29.2.32	setCMSDeliveryMode	299
6.29.2.33	setCMSDestination	300
6.29.2.34	setCMSExpiration	300
6.29.2.35	setCMSMessageID	300
6.29.2.36	setCMSPriority	300
6.29.2.37	setCMSRedelivered	300
6.29.2.38	setCMSReplyTo	300
6.29.2.39	setCMSTimestamp	300
6.29.2.40	setCMSType	300
6.29.2.41	setDoubleProperty	300
6.29.2.42	setFloatProperty	300
6.29.2.43	setIntProperty	300
6.29.2.44	setLongProperty	300
6.29.2.45	setShortProperty	301
6.29.2.46	setStringProperty	301
6.30	activemq::commands::ActiveMQObjectMessage Class Reference	301
6.30.1	Constructor & Destructor Documentation	302
6.30.1.1	ActiveMQObjectMessage	302
6.30.1.2	~ActiveMQObjectMessage	302
6.30.2	Member Function Documentation	302
6.30.2.1	clone	302
6.30.2.2	cloneDataStructure	302
6.30.2.3	copyDataStructure	302
6.30.2.4	equals	303
6.30.2.5	getDataStructureType	303
6.30.2.6	toString	303
6.30.3	Field Documentation	303
6.30.3.1	ID_ACTIVEMQOBJECTMESSAGE	303
6.31	activemq::wireformat::openwire::marshal::generated::ActiveMQObject- MessageMarshaller Class Reference	304
6.31.1	Detailed Description	304

6.31.2	Constructor & Destructor Documentation	305
6.31.2.1	ActiveMQObjectMessageMarshaller	305
6.31.2.2	~ActiveMQObjectMessageMarshaller	305
6.31.3	Member Function Documentation	305
6.31.3.1	createObject	305
6.31.3.2	getDataStructureType	305
6.31.3.3	looseMarshal	305
6.31.3.4	looseUnmarshal	306
6.31.3.5	tightMarshal1	306
6.31.3.6	tightMarshal2	307
6.31.3.7	tightUnmarshal	307
6.32	activemq::core::ActiveMQProducer Class Reference	308
6.32.1	Constructor & Destructor Documentation	309
6.32.1.1	ActiveMQProducer	309
6.32.1.2	~ActiveMQProducer	309
6.32.2	Member Function Documentation	309
6.32.2.1	close	310
6.32.2.2	dispose	310
6.32.2.3	getDeliveryMode	310
6.32.2.4	getDisableMessageID	310
6.32.2.5	getDisableMessageTimeStamp	310
6.32.2.6	getPriority	311
6.32.2.7	getProducerId	311
6.32.2.8	getProducerInfo	311
6.32.2.9	getSendTimeout	311
6.32.2.10	getTimeToLive	312
6.32.2.11	isClosed	312
6.32.2.12	onProducerAck	312
6.32.2.13	send	312
6.32.2.14	send	313
6.32.2.15	send	313
6.32.2.16	send	314
6.32.2.17	setDeliveryMode	314
6.32.2.18	setDisableMessageID	315

6.32.2.19	setDisableMessageTimeStamp	315
6.32.2.20	setPriority	315
6.32.2.21	setSendTimeout	315
6.32.2.22	setTimeToLive	315
6.33	activemq::util::ActiveMQProperties Class Reference	316
6.33.1	Detailed Description	317
6.33.2	Constructor & Destructor Documentation	317
6.33.2.1	ActiveMQProperties	317
6.33.2.2	~ActiveMQProperties	317
6.33.3	Member Function Documentation	317
6.33.3.1	clear	317
6.33.3.2	clone	317
6.33.3.3	copy	318
6.33.3.4	getProperties	318
6.33.3.5	getProperties	318
6.33.3.6	getProperty	318
6.33.3.7	getProperty	318
6.33.3.8	hasProperty	318
6.33.3.9	isEmpty	319
6.33.3.10	propertyNames	319
6.33.3.11	remove	319
6.33.3.12	setProperties	320
6.33.3.13	setProperty	320
6.33.3.14	size	320
6.33.3.15	toArray	320
6.33.3.16	toString	321
6.34	activemq::commands::ActiveMQQueue Class Reference	321
6.34.1	Constructor & Destructor Documentation	322
6.34.1.1	ActiveMQQueue	322
6.34.1.2	ActiveMQQueue	322
6.34.1.3	~ActiveMQQueue	322
6.34.2	Member Function Documentation	322
6.34.2.1	clone	322
6.34.2.2	cloneDataStructure	322

6.34.2.3	copy	323
6.34.2.4	copyDataStructure	323
6.34.2.5	equals	323
6.34.2.6	equals	323
6.34.2.7	getCMSDestination	324
6.34.2.8	getCMSProperties	324
6.34.2.9	getDataStructureType	324
6.34.2.10	getDestinationType	325
6.34.2.11	getQueueName	325
6.34.2.12	toString	325
6.34.3	Field Documentation	325
6.34.3.1	ID_ACTIVEMQQUEUE	325
6.35	activemq::core::ActiveMQQueueBrowser Class Reference	326
6.35.1	Constructor & Destructor Documentation	326
6.35.1.1	ActiveMQQueueBrowser	326
6.35.1.2	~ActiveMQQueueBrowser	326
6.35.2	Member Function Documentation	327
6.35.2.1	close	327
6.35.2.2	getEnumeration	327
6.35.2.3	getMessageSelector	327
6.35.2.4	getQueue	328
6.35.2.5	hasMoreMessages	328
6.35.2.6	nextMessage	328
6.35.3	Friends And Related Function Documentation	329
6.35.3.1	Browser	329
6.36	activemq::wireformat::openwire::marshal::generated::ActiveMQQueue- Marshaller Class Reference	329
6.36.1	Detailed Description	330
6.36.2	Constructor & Destructor Documentation	330
6.36.2.1	ActiveMQQueueMarshaller	330
6.36.2.2	~ActiveMQQueueMarshaller	330
6.36.3	Member Function Documentation	330
6.36.3.1	createObject	330
6.36.3.2	getDataStructureType	330

6.36.3.3	looseMarshal	331
6.36.3.4	looseUnmarshal	331
6.36.3.5	tightMarshal1	331
6.36.3.6	tightMarshal2	332
6.36.3.7	tightUnmarshal	332
6.37	activemq::core::ActiveMQSession Class Reference	333
6.37.1	Constructor & Destructor Documentation	337
6.37.1.1	ActiveMQSession	337
6.37.1.2	~ActiveMQSession	337
6.37.2	Member Function Documentation	337
6.37.2.1	acknowledge	337
6.37.2.2	addConsumer	337
6.37.2.3	addProducer	338
6.37.2.4	clearMessagesInProgress	338
6.37.2.5	close	338
6.37.2.6	commit	338
6.37.2.7	createBrowser	339
6.37.2.8	createBrowser	339
6.37.2.9	createBytesMessage	340
6.37.2.10	createBytesMessage	340
6.37.2.11	createConsumer	340
6.37.2.12	createConsumer	341
6.37.2.13	createConsumer	341
6.37.2.14	createDurableConsumer	342
6.37.2.15	createMapMessage	342
6.37.2.16	createMessage	343
6.37.2.17	createProducer	343
6.37.2.18	createQueue	343
6.37.2.19	createStreamMessage	344
6.37.2.20	createTemporaryQueue	344
6.37.2.21	createTemporaryTopic	344
6.37.2.22	createTextMessage	345
6.37.2.23	createTextMessage	345
6.37.2.24	createTopic	345

6.37.2.25 deliverAcks	346
6.37.2.26 dispatch	346
6.37.2.27 dispose	346
6.37.2.28 doClose	346
6.37.2.29 doStartTransaction	346
6.37.2.30 fire	347
6.37.2.31 getAcknowledgeMode	347
6.37.2.32 getConnection	347
6.37.2.33 getExceptionListener	347
6.37.2.34 getLastDeliveredSequenceId	347
6.37.2.35 getNextConsumerId	348
6.37.2.36 getNextProducerId	348
6.37.2.37 getScheduler	348
6.37.2.38 getSessionId	348
6.37.2.39 getSessionInfo	348
6.37.2.40 getTransactionContext	349
6.37.2.41 isAutoAcknowledge	349
6.37.2.42 isClientAcknowledge	349
6.37.2.43 isDupsOkAcknowledge	349
6.37.2.44 isIndividualAcknowledge	349
6.37.2.45 isStarted	349
6.37.2.46 isTransacted	349
6.37.2.47 oneway	350
6.37.2.48 recover	350
6.37.2.49 redispach	351
6.37.2.50 removeConsumer	351
6.37.2.51 removeProducer	351
6.37.2.52 rollback	351
6.37.2.53 send	352
6.37.2.54 setLastDeliveredSequenceId	352
6.37.2.55 start	352
6.37.2.56 stop	352
6.37.2.57 syncRequest	353
6.37.2.58 unsubscribe	353

6.37.2.59	wakeup	353
6.37.3	Friends And Related Function Documentation	354
6.37.3.1	ActiveMQSessionExecutor	354
6.37.4	Field Documentation	354
6.37.4.1	ackMode	354
6.37.4.2	closed	354
6.37.4.3	config	354
6.37.4.4	connection	354
6.37.4.5	consumerIds	354
6.37.4.6	consumers	354
6.37.4.7	executor	354
6.37.4.8	lastDeliveredSequenceId	354
6.37.4.9	producerIds	355
6.37.4.10	producerSequenceIds	355
6.37.4.11	sessionInfo	355
6.37.4.12	transaction	355
6.38	activemq::core::ActiveMQSessionExecutor Class Reference	355
6.38.1	Detailed Description	356
6.38.2	Constructor & Destructor Documentation	356
6.38.2.1	ActiveMQSessionExecutor	356
6.38.2.2	~ActiveMQSessionExecutor	356
6.38.3	Member Function Documentation	356
6.38.3.1	clear	356
6.38.3.2	clearMessagesInProgress	357
6.38.3.3	close	357
6.38.3.4	execute	357
6.38.3.5	executeFirst	357
6.38.3.6	getUnconsumedMessages	357
6.38.3.7	hasUnconsumedMessages	358
6.38.3.8	isEmpty	358
6.38.3.9	isRunning	358
6.38.3.10	iterate	358
6.38.3.11	start	358
6.38.3.12	stop	359

6.38.3.13	wakeup	359
6.39	activemq::commands::ActiveMQStreamMessage Class Reference . . .	359
6.39.1	Constructor & Destructor Documentation	361
6.39.1.1	ActiveMQStreamMessage	361
6.39.1.2	~ActiveMQStreamMessage	361
6.39.2	Member Function Documentation	361
6.39.2.1	clearBody	361
6.39.2.2	clone	361
6.39.2.3	cloneDataStructure	362
6.39.2.4	copyDataStructure	362
6.39.2.5	equals	362
6.39.2.6	getDataStructureType	363
6.39.2.7	onSend	363
6.39.2.8	readBoolean	363
6.39.2.9	readByte	363
6.39.2.10	readBytes	364
6.39.2.11	readBytes	365
6.39.2.12	readChar	365
6.39.2.13	readDouble	366
6.39.2.14	readFloat	366
6.39.2.15	readInt	367
6.39.2.16	readLong	367
6.39.2.17	readShort	368
6.39.2.18	readString	368
6.39.2.19	readUnsignedShort	369
6.39.2.20	reset	369
6.39.2.21	toString	369
6.39.2.22	writeBoolean	370
6.39.2.23	writeByte	370
6.39.2.24	writeBytes	370
6.39.2.25	writeBytes	371
6.39.2.26	writeChar	371
6.39.2.27	writeDouble	372
6.39.2.28	writeFloat	372

6.39.2.29	writeInt	372
6.39.2.30	writeLong	373
6.39.2.31	writeShort	373
6.39.2.32	writeString	374
6.39.2.33	writeUnsignedShort	374
6.39.3	Field Documentation	374
6.39.3.1	ID_ACTIVEMQSTREAMMESSAGE	374
6.40	activemq::wireformat::openwire::marshal::generated::ActiveMQStream- MessageMarshaller Class Reference	375
6.40.1	Detailed Description	375
6.40.2	Constructor & Destructor Documentation	376
6.40.2.1	ActiveMQStreamMessageMarshaller	376
6.40.2.2	~ActiveMQStreamMessageMarshaller	376
6.40.3	Member Function Documentation	376
6.40.3.1	createObject	376
6.40.3.2	getDataStructureType	376
6.40.3.3	looseMarshal	376
6.40.3.4	looseUnmarshal	377
6.40.3.5	tightMarshal1	377
6.40.3.6	tightMarshal2	378
6.40.3.7	tightUnmarshal	378
6.41	activemq::commands::ActiveMQTempDestination Class Reference	379
6.41.1	Constructor & Destructor Documentation	380
6.41.1.1	ActiveMQTempDestination	380
6.41.1.2	ActiveMQTempDestination	380
6.41.1.3	~ActiveMQTempDestination	380
6.41.2	Member Function Documentation	380
6.41.2.1	cloneDataStructure	380
6.41.2.2	close	380
6.41.2.3	copyDataStructure	380
6.41.2.4	equals	381
6.41.2.5	getDataStructureType	381
6.41.2.6	setConnection	381
6.41.2.7	toString	382

6.41.3	Field Documentation	382
6.41.3.1	connection	382
6.41.3.2	ID_ACTIVEMQTEMPDESTINATION	382
6.42	activemq::wireformat::openwire::marshal::generated::ActiveMQTemp- DestinationMarshaller Class Reference	382
6.42.1	Detailed Description	383
6.42.2	Constructor & Destructor Documentation	383
6.42.2.1	ActiveMQTempDestinationMarshaller	383
6.42.2.2	~ActiveMQTempDestinationMarshaller	383
6.42.3	Member Function Documentation	383
6.42.3.1	looseMarshal	384
6.42.3.2	looseUnmarshal	384
6.42.3.3	tightMarshal1	385
6.42.3.4	tightMarshal2	385
6.42.3.5	tightUnmarshal	386
6.43	activemq::commands::ActiveMQTempQueue Class Reference	386
6.43.1	Constructor & Destructor Documentation	387
6.43.1.1	ActiveMQTempQueue	387
6.43.1.2	ActiveMQTempQueue	387
6.43.1.3	~ActiveMQTempQueue	387
6.43.2	Member Function Documentation	387
6.43.2.1	clone	388
6.43.2.2	cloneDataStructure	388
6.43.2.3	copy	388
6.43.2.4	copyDataStructure	388
6.43.2.5	destroy	389
6.43.2.6	equals	389
6.43.2.7	equals	389
6.43.2.8	getCMSDestination	389
6.43.2.9	getCMSProperties	390
6.43.2.10	getDataStructureType	390
6.43.2.11	getDestinationType	390
6.43.2.12	getQueueName	391
6.43.2.13	toString	391

6.43.3	Field Documentation	391
6.43.3.1	ID_ACTIVEMQTEMPQUEUE	391
6.44	activemq::wireformat::openwire::marshal::generated::ActiveMQTempQueueMarshaller Class Reference	391
6.44.1	Detailed Description	392
6.44.2	Constructor & Destructor Documentation	392
6.44.2.1	ActiveMQTempQueueMarshaller	392
6.44.2.2	~ActiveMQTempQueueMarshaller	393
6.44.3	Member Function Documentation	393
6.44.3.1	createObject	393
6.44.3.2	getDataStructureType	393
6.44.3.3	looseMarshal	393
6.44.3.4	looseUnmarshal	394
6.44.3.5	tightMarshal1	394
6.44.3.6	tightMarshal2	395
6.44.3.7	tightUnmarshal	395
6.45	activemq::commands::ActiveMQTempTopic Class Reference	396
6.45.1	Constructor & Destructor Documentation	397
6.45.1.1	ActiveMQTempTopic	397
6.45.1.2	ActiveMQTempTopic	397
6.45.1.3	~ActiveMQTempTopic	397
6.45.2	Member Function Documentation	397
6.45.2.1	clone	397
6.45.2.2	cloneDataStructure	397
6.45.2.3	copy	397
6.45.2.4	copyDataStructure	398
6.45.2.5	destroy	398
6.45.2.6	equals	398
6.45.2.7	equals	399
6.45.2.8	getCMSDestination	399
6.45.2.9	getCMSProperties	399
6.45.2.10	getDataStructureType	399
6.45.2.11	getDestinationType	400
6.45.2.12	getTopicName	400

6.45.2.13	toString	400
6.45.3	Field Documentation	400
6.45.3.1	ID_ACTIVEMQTEMPTOPIC	401
6.46	activemq::wireformat::openwire::marshal::generated::ActiveMQTemp- TopicMarshaller Class Reference	401
6.46.1	Detailed Description	402
6.46.2	Constructor & Destructor Documentation	402
6.46.2.1	ActiveMQTempTopicMarshaller	402
6.46.2.2	~ActiveMQTempTopicMarshaller	402
6.46.3	Member Function Documentation	402
6.46.3.1	createObject	402
6.46.3.2	getDataStructureType	402
6.46.3.3	looseMarshal	403
6.46.3.4	looseUnmarshal	403
6.46.3.5	tightMarshal1	403
6.46.3.6	tightMarshal2	404
6.46.3.7	tightUnmarshal	404
6.47	activemq::commands::ActiveMQTextMessage Class Reference	405
6.47.1	Constructor & Destructor Documentation	406
6.47.1.1	ActiveMQTextMessage	406
6.47.1.2	~ActiveMQTextMessage	406
6.47.2	Member Function Documentation	406
6.47.2.1	beforeMarshal	406
6.47.2.2	clearBody	406
6.47.2.3	clone	407
6.47.2.4	cloneDataStructure	407
6.47.2.5	copyDataStructure	407
6.47.2.6	equals	408
6.47.2.7	getDataStructureType	408
6.47.2.8	getSize	408
6.47.2.9	getText	408
6.47.2.10	setText	409
6.47.2.11	setText	409
6.47.2.12	toString	409

6.47.3	Field Documentation	410
6.47.3.1	ID_ACTIVEMQTEXTMESSAGE	410
6.47.3.2	text	410
6.48	activemq::wireformat::openwire::marshal::generated::ActiveMQText- MessageMarshaller Class Reference	410
6.48.1	Detailed Description	411
6.48.2	Constructor & Destructor Documentation	411
6.48.2.1	ActiveMQTextMessageMarshaller	411
6.48.2.2	~ActiveMQTextMessageMarshaller	411
6.48.3	Member Function Documentation	411
6.48.3.1	createObject	411
6.48.3.2	getDataStructureType	412
6.48.3.3	looseMarshal	412
6.48.3.4	looseUnmarshal	412
6.48.3.5	tightMarshal1	413
6.48.3.6	tightMarshal2	413
6.48.3.7	tightUnmarshal	414
6.49	activemq::commands::ActiveMQTopic Class Reference	414
6.49.1	Constructor & Destructor Documentation	415
6.49.1.1	ActiveMQTopic	415
6.49.1.2	ActiveMQTopic	415
6.49.1.3	~ActiveMQTopic	415
6.49.2	Member Function Documentation	415
6.49.2.1	clone	415
6.49.2.2	cloneDataStructure	416
6.49.2.3	copy	416
6.49.2.4	copyDataStructure	416
6.49.2.5	equals	416
6.49.2.6	equals	417
6.49.2.7	getCMSDestination	417
6.49.2.8	getCMSProperties	417
6.49.2.9	getDataStructureType	417
6.49.2.10	getDestinationType	418
6.49.2.11	getTopicName	418

6.49.2.12	toString	418
6.49.3	Field Documentation	418
6.49.3.1	ID_ACTIVEMQTOPIC	418
6.50	activemq::wireformat::openwire::marshal::generated::ActiveMQTopic- Marshaller Class Reference	419
6.50.1	Detailed Description	420
6.50.2	Constructor & Destructor Documentation	420
6.50.2.1	ActiveMQTopicMarshaller	420
6.50.2.2	~ActiveMQTopicMarshaller	420
6.50.3	Member Function Documentation	420
6.50.3.1	createObject	420
6.50.3.2	getDataStructureType	420
6.50.3.3	looseMarshal	421
6.50.3.4	looseUnmarshal	421
6.50.3.5	tightMarshal1	421
6.50.3.6	tightMarshal2	422
6.50.3.7	tightUnmarshal	422
6.51	activemq::core::ActiveMQTransactionContext Class Reference	423
6.51.1	Detailed Description	424
6.51.2	Constructor & Destructor Documentation	425
6.51.2.1	ActiveMQTransactionContext	425
6.51.2.2	~ActiveMQTransactionContext	425
6.51.3	Member Function Documentation	425
6.51.3.1	addSynchronization	425
6.51.3.2	begin	425
6.51.3.3	commit	425
6.51.3.4	commit	426
6.51.3.5	end	426
6.51.3.6	forget	427
6.51.3.7	getTransactionId	427
6.51.3.8	getTransactionTimeout	427
6.51.3.9	isInLocalTransaction	428
6.51.3.10	isInTransaction	428
6.51.3.11	isInXATransaction	428

6.51.3.12	isSameRM	428
6.51.3.13	prepare	429
6.51.3.14	recover	429
6.51.3.15	removeSynchronization	430
6.51.3.16	rollback	430
6.51.3.17	rollback	430
6.51.3.18	setTransactionTimeout	431
6.51.3.19	start	431
6.52	activemq::core::ActiveMQXAConnection Class Reference	432
6.52.1	Constructor & Destructor Documentation	432
6.52.1.1	ActiveMQXAConnection	432
6.52.1.2	~ActiveMQXAConnection	432
6.52.2	Member Function Documentation	433
6.52.2.1	createSession	433
6.52.2.2	createXASession	433
6.53	activemq::core::ActiveMQXAConnectionFactory Class Reference	433
6.53.1	Constructor & Destructor Documentation	434
6.53.1.1	ActiveMQXAConnectionFactory	434
6.53.1.2	ActiveMQXAConnectionFactory	434
6.53.1.3	ActiveMQXAConnectionFactory	435
6.53.1.4	~ActiveMQXAConnectionFactory	435
6.53.2	Member Function Documentation	435
6.53.2.1	createActiveMQConnection	435
6.53.2.2	createXAConnection	435
6.53.2.3	createXAConnection	436
6.54	activemq::core::ActiveMQXASession Class Reference	436
6.54.1	Constructor & Destructor Documentation	437
6.54.1.1	ActiveMQXASession	437
6.54.1.2	~ActiveMQXASession	437
6.54.2	Member Function Documentation	437
6.54.2.1	commit	437
6.54.2.2	doStartTransaction	437
6.54.2.3	getXAResource	438
6.54.2.4	isAutoAcknowledge	438

6.54.2.5	isTransacted	438
6.54.2.6	rollback	439
6.55	decaf::util::zip::Adler32 Class Reference	439
6.55.1	Detailed Description	440
6.55.2	Constructor & Destructor Documentation	440
6.55.2.1	Adler32	440
6.55.2.2	~Adler32	440
6.55.3	Member Function Documentation	440
6.55.3.1	getValue	440
6.55.3.2	reset	440
6.55.3.3	update	440
6.55.3.4	update	440
6.55.3.5	update	441
6.55.3.6	update	441
6.56	decaf::lang::Appendable Class Reference	442
6.56.1	Detailed Description	442
6.56.2	Constructor & Destructor Documentation	442
6.56.2.1	~Appendable	442
6.56.3	Member Function Documentation	442
6.56.3.1	append	443
6.56.3.2	append	443
6.56.3.3	append	443
6.57	decaf::internal::AprPool Class Reference	444
6.57.1	Detailed Description	445
6.57.2	Constructor & Destructor Documentation	445
6.57.2.1	AprPool	445
6.57.2.2	~AprPool	445
6.57.3	Member Function Documentation	445
6.57.3.1	cleanup	445
6.57.3.2	getAprPool	445
6.57.3.3	getGlobalPool	445
6.58	decaf::util::ArrayList< E > Class Template Reference	445
6.58.1	Constructor & Destructor Documentation	448
6.58.1.1	ArrayList	448

6.58.1.2	ArrayList	448
6.58.1.3	ArrayList	448
6.58.1.4	ArrayList	448
6.58.1.5	~ArrayList	448
6.58.2	Member Function Documentation	448
6.58.2.1	add	448
6.58.2.2	add	449
6.58.2.3	addAll	450
6.58.2.4	addAll	451
6.58.2.5	clear	452
6.58.2.6	contains	452
6.58.2.7	ensureCapacity	453
6.58.2.8	get	453
6.58.2.9	indexOf	453
6.58.2.10	isEmpty	454
6.58.2.11	lastIndexOf	454
6.58.2.12	operator=	455
6.58.2.13	operator=	455
6.58.2.14	remove	455
6.58.2.15	removeAt	456
6.58.2.16	set	456
6.58.2.17	size	457
6.58.2.18	toArray	457
6.58.2.19	toString	458
6.58.2.20	trimToSize	458
6.59	decaf::util::concurrent::CopyOnWriteArrayList< E >::ArrayListIterator - Class Reference	458
6.59.1	Constructor & Destructor Documentation	459
6.59.1.1	ArrayListIterator	459
6.59.1.2	~ArrayListIterator	459
6.59.2	Member Function Documentation	459
6.59.2.1	add	459
6.59.2.2	hasNext	459
6.59.2.3	hasPrevious	460

6.59.2.4	next	460
6.59.2.5	nextIndex	460
6.59.2.6	previous	461
6.59.2.7	previousIndex	461
6.59.2.8	remove	461
6.59.2.9	set	462
6.60	decaf::lang::ArrayPointer< T, REFCOUNTER > Class Template - Reference	462
6.60.1	Detailed Description	464
6.60.2	Member Typedef Documentation	464
6.60.2.1	ConstReferenceType	464
6.60.2.2	CounterType	464
6.60.2.3	PointerType	464
6.60.2.4	ReferenceType	464
6.60.3	Constructor & Destructor Documentation	464
6.60.3.1	ArrayPointer	464
6.60.3.2	ArrayPointer	465
6.60.3.3	ArrayPointer	465
6.60.3.4	ArrayPointer	465
6.60.3.5	ArrayPointer	466
6.60.3.6	~ArrayPointer	466
6.60.4	Member Function Documentation	466
6.60.4.1	clone	466
6.60.4.2	get	466
6.60.4.3	length	467
6.60.4.4	operator!	467
6.60.4.5	operator!=	467
6.60.4.6	operator=	467
6.60.4.7	operator=	468
6.60.4.8	operator==	468
6.60.4.9	operator[]	468
6.60.4.10	operator[]	468
6.60.4.11	release	468
6.60.4.12	reset	469

6.60.4.13	swap	469
6.60.5	Friends And Related Function Documentation	469
6.60.5.1	operator!=	469
6.60.5.2	operator!=	469
6.60.5.3	operator==	470
6.60.5.4	operator==	470
6.61	decaf::lang::ArrayPointerComparator< T, R > Class Template Reference	470
6.61.1	Detailed Description	470
6.61.2	Constructor & Destructor Documentation	471
6.61.2.1	~ArrayPointerComparator	471
6.61.3	Member Function Documentation	471
6.61.3.1	compare	471
6.61.3.2	operator()	471
6.62	decaf::util::Arrays Class Reference	471
6.62.1	Constructor & Destructor Documentation	472
6.62.1.1	~Arrays	472
6.62.2	Member Function Documentation	472
6.62.2.1	fill	472
6.62.2.2	fill	472
6.63	decaf::util::concurrent::atomic::AtomicBoolean Class Reference	473
6.63.1	Detailed Description	473
6.63.2	Constructor & Destructor Documentation	473
6.63.2.1	AtomicBoolean	473
6.63.2.2	AtomicBoolean	474
6.63.2.3	~AtomicBoolean	474
6.63.3	Member Function Documentation	474
6.63.3.1	compareAndSet	474
6.63.3.2	get	474
6.63.3.3	getAndSet	474
6.63.3.4	set	475
6.63.3.5	toString	475
6.64	decaf::util::concurrent::atomic::AtomicInteger Class Reference	475
6.64.1	Detailed Description	476
6.64.2	Constructor & Destructor Documentation	477

6.64.2.1	AtomicInteger	477
6.64.2.2	AtomicInteger	477
6.64.2.3	~AtomicInteger	477
6.64.3	Member Function Documentation	477
6.64.3.1	addAndGet	477
6.64.3.2	compareAndSet	477
6.64.3.3	decrementAndGet	478
6.64.3.4	doubleValue	478
6.64.3.5	floatValue	478
6.64.3.6	get	478
6.64.3.7	getAndAdd	479
6.64.3.8	getAndDecrement	479
6.64.3.9	getAndIncrement	479
6.64.3.10	getAndSet	480
6.64.3.11	incrementAndGet	480
6.64.3.12	intValue	480
6.64.3.13	longValue	480
6.64.3.14	set	481
6.64.3.15	toString	481
6.65	decaf::util::concurrent::atomic::AtomicRefCounter Class Reference	481
6.65.1	Constructor & Destructor Documentation	482
6.65.1.1	AtomicRefCounter	482
6.65.1.2	AtomicRefCounter	482
6.65.1.3	~AtomicRefCounter	482
6.65.2	Member Function Documentation	483
6.65.2.1	release	483
6.65.2.2	swap	484
6.66	decaf::util::concurrent::atomic::AtomicReference< T > Class Template Reference	484
6.66.1	Detailed Description	484
6.66.2	Constructor & Destructor Documentation	485
6.66.2.1	AtomicReference	485
6.66.2.2	AtomicReference	485
6.66.2.3	~AtomicReference	485

6.66.3	Member Function Documentation	485
6.66.3.1	compareAndSet	485
6.66.3.2	get	485
6.66.3.3	getAndSet	485
6.66.3.4	set	486
6.66.3.5	toString	486
6.67	activemq::transport::failover::BackupTransport Class Reference	486
6.67.1	Constructor & Destructor Documentation	487
6.67.1.1	BackupTransport	487
6.67.1.2	~BackupTransport	487
6.67.2	Member Function Documentation	487
6.67.2.1	getTransport	487
6.67.2.2	getUri	487
6.67.2.3	isClosed	488
6.67.2.4	onException	488
6.67.2.5	setClosed	488
6.67.2.6	setTransport	488
6.67.2.7	setUri	488
6.68	activemq::transport::failover::BackupTransportPool Class Reference	489
6.68.1	Constructor & Destructor Documentation	490
6.68.1.1	BackupTransportPool	490
6.68.1.2	BackupTransportPool	490
6.68.1.3	~BackupTransportPool	490
6.68.2	Member Function Documentation	490
6.68.2.1	getBackup	490
6.68.2.2	getBackupPoolSize	490
6.68.2.3	isEnabled	490
6.68.2.4	isPending	491
6.68.2.5	iterate	491
6.68.2.6	setBackupPoolSize	491
6.68.2.7	setEnabled	491
6.68.3	Friends And Related Function Documentation	491
6.68.3.1	BackupTransport	491
6.69	activemq::commands::BaseCommand Class Reference	491

6.69.1	Constructor & Destructor Documentation	493
6.69.1.1	BaseCommand	493
6.69.1.2	~BaseCommand	493
6.69.2	Member Function Documentation	493
6.69.2.1	copyDataStructure	493
6.69.2.2	equals	494
6.69.2.3	getCommandId	495
6.69.2.4	isBrokerInfo	495
6.69.2.5	isConnectionControl	495
6.69.2.6	isConnectionInfo	495
6.69.2.7	isConsumerInfo	495
6.69.2.8	isKeepAliveInfo	495
6.69.2.9	isMessage	496
6.69.2.10	isMessageAck	496
6.69.2.11	isMessageDispatch	496
6.69.2.12	isMessageDispatchNotification	496
6.69.2.13	isProducerAck	496
6.69.2.14	isProducerInfo	496
6.69.2.15	isRemoveInfo	496
6.69.2.16	isRemoveSubscriptionInfo	497
6.69.2.17	isResponse	497
6.69.2.18	isResponseRequired	497
6.69.2.19	isShutdownInfo	497
6.69.2.20	isTransactionInfo	497
6.69.2.21	isWireFormatInfo	497
6.69.2.22	setCommandId	498
6.69.2.23	setResponseRequired	498
6.69.2.24	toString	498
6.70	activemq::wireformat::openwire::marshal::generated::BaseCommand- Marshaller Class Reference	499
6.70.1	Detailed Description	500
6.70.2	Constructor & Destructor Documentation	500
6.70.2.1	BaseCommandMarshaller	500
6.70.2.2	~BaseCommandMarshaller	500

6.70.3	Member Function Documentation	500
6.70.3.1	looseMarshal	500
6.70.3.2	looseUnmarshal	501
6.70.3.3	tightMarshal1	503
6.70.3.4	tightMarshal2	504
6.70.3.5	tightUnmarshal	505
6.71	activemq::wireformat::openwire::marshal::BaseDataStreamMarshaller - Class Reference	507
6.71.1	Detailed Description	510
6.71.2	Constructor & Destructor Documentation	511
6.71.2.1	~BaseDataStreamMarshaller	511
6.71.3	Member Function Documentation	511
6.71.3.1	looseMarshal	511
6.71.3.2	looseMarshalBrokerError	511
6.71.3.3	looseMarshalCachedObject	512
6.71.3.4	looseMarshalLong	512
6.71.3.5	looseMarshalNestedObject	512
6.71.3.6	looseMarshalObjectArray	513
6.71.3.7	looseMarshalString	513
6.71.3.8	looseUnmarshal	514
6.71.3.9	looseUnmarshalBrokerError	514
6.71.3.10	looseUnmarshalByteArray	514
6.71.3.11	looseUnmarshalCachedObject	515
6.71.3.12	looseUnmarshalConstByteArray	515
6.71.3.13	looseUnmarshalLong	516
6.71.3.14	looseUnmarshalNestedObject	516
6.71.3.15	looseUnmarshalString	517
6.71.3.16	readAsciiString	517
6.71.3.17	tightMarshal1	517
6.71.3.18	tightMarshal2	518
6.71.3.19	tightMarshalBrokerError1	518
6.71.3.20	tightMarshalBrokerError2	519
6.71.3.21	tightMarshalCachedObject1	519
6.71.3.22	tightMarshalCachedObject2	520

6.71.3.23	tightMarshalLong1	520
6.71.3.24	tightMarshalLong2	520
6.71.3.25	tightMarshalNestedObject1	521
6.71.3.26	tightMarshalNestedObject2	521
6.71.3.27	tightMarshalObjectArray1	522
6.71.3.28	tightMarshalObjectArray2	522
6.71.3.29	tightMarshalString1	523
6.71.3.30	tightMarshalString2	523
6.71.3.31	tightUnmarshal	524
6.71.3.32	tightUnmarshalBrokerError	524
6.71.3.33	tightUnmarshalByteArray	524
6.71.3.34	tightUnmarshalCachedObject	525
6.71.3.35	tightUnmarshalConstByteArray	525
6.71.3.36	tightUnmarshalLong	526
6.71.3.37	tightUnmarshalNestedObject	526
6.71.3.38	tightUnmarshalString	527
6.71.3.39	toHexFromBytes	527
6.71.3.40	toString	527
6.71.3.41	toString	528
6.71.3.42	toString	528
6.72	activemq::commands::BaseDataStructure Class Reference	528
6.72.1	Constructor & Destructor Documentation	529
6.72.1.1	~BaseDataStructure	529
6.72.2	Member Function Documentation	529
6.72.2.1	afterMarshal	529
6.72.2.2	afterUnmarshal	529
6.72.2.3	beforeMarshal	530
6.72.2.4	beforeUnmarshal	530
6.72.2.5	copyDataStructure	530
6.72.2.6	equals	530
6.72.2.7	getMarshaledForm	530
6.72.2.8	isMarshalAware	530
6.72.2.9	setMarshaledForm	531
6.72.2.10	toString	531

6.73	decaf::net::BindException Class Reference	532
6.73.1	Constructor & Destructor Documentation	533
6.73.1.1	BindException	533
6.73.1.2	BindException	533
6.73.1.3	BindException	533
6.73.1.4	BindException	533
6.73.1.5	BindException	533
6.73.1.6	BindException	534
6.73.1.7	~BindException	534
6.73.2	Member Function Documentation	534
6.73.2.1	clone	534
6.74	decaf::io::BlockingByteArrayInputStream Class Reference	534
6.74.1	Detailed Description	536
6.74.2	Constructor & Destructor Documentation	536
6.74.2.1	BlockingByteArrayInputStream	536
6.74.2.2	BlockingByteArrayInputStream	536
6.74.2.3	~BlockingByteArrayInputStream	536
6.74.3	Member Function Documentation	536
6.74.3.1	available	536
6.74.3.2	close	537
6.74.3.3	doReadArrayBounded	537
6.74.3.4	doReadByte	537
6.74.3.5	setByteArray	537
6.74.3.6	skip	537
6.75	decaf::util::concurrent::BlockingQueue< E > Class Template Reference	538
6.75.1	Detailed Description	539
6.75.2	Constructor & Destructor Documentation	541
6.75.2.1	~BlockingQueue	541
6.75.3	Member Function Documentation	541
6.75.3.1	drainTo	541
6.75.3.2	drainTo	542
6.75.3.3	offer	543
6.75.3.4	poll	543
6.75.3.5	put	544

6.75.3.6	remainingCapacity	544
6.75.3.7	take	545
6.76	decaf::lang::Boolean Class Reference	545
6.76.1	Constructor & Destructor Documentation	546
6.76.1.1	Boolean	546
6.76.1.2	Boolean	546
6.76.1.3	~Boolean	546
6.76.2	Member Function Documentation	546
6.76.2.1	booleanValue	546
6.76.2.2	compareTo	547
6.76.2.3	compareTo	547
6.76.2.4	equals	547
6.76.2.5	equals	548
6.76.2.6	operator<	548
6.76.2.7	operator<	548
6.76.2.8	operator==	548
6.76.2.9	operator==	549
6.76.2.10	parseBoolean	549
6.76.2.11	toString	549
6.76.2.12	toString	549
6.76.2.13	valueOf	550
6.76.2.14	valueOf	550
6.76.3	Field Documentation	550
6.76.3.1	_FALSE	550
6.76.3.2	_TRUE	550
6.77	activemq::commands::BooleanExpression Class Reference	551
6.77.1	Constructor & Destructor Documentation	551
6.77.1.1	BooleanExpression	551
6.77.1.2	~BooleanExpression	551
6.77.2	Member Function Documentation	551
6.77.2.1	cloneDataStructure	551
6.77.2.2	copyDataStructure	552
6.77.2.3	equals	552
6.77.2.4	toString	552

6.78	activemq::wireformat::openwire::utils::BooleanStream Class Reference	552
6.78.1	Detailed Description	553
6.78.2	Constructor & Destructor Documentation	553
6.78.2.1	BooleanStream	553
6.78.2.2	~BooleanStream	553
6.78.3	Member Function Documentation	554
6.78.3.1	clear	554
6.78.3.2	marshal	554
6.78.3.3	marshal	554
6.78.3.4	marshalledSize	554
6.78.3.5	readBoolean	554
6.78.3.6	unmarshal	555
6.78.3.7	writeBoolean	555
6.79	decaf::util::concurrent::BrokenBarrierException Class Reference	556
6.79.1	Constructor & Destructor Documentation	556
6.79.1.1	BrokenBarrierException	556
6.79.1.2	BrokenBarrierException	556
6.79.1.3	BrokenBarrierException	557
6.79.1.4	BrokenBarrierException	557
6.79.1.5	BrokenBarrierException	557
6.79.1.6	BrokenBarrierException	557
6.79.1.7	~BrokenBarrierException	558
6.79.2	Member Function Documentation	558
6.79.2.1	clone	558
6.80	activemq::commands::BrokerError Class Reference	558
6.80.1	Detailed Description	559
6.80.2	Constructor & Destructor Documentation	560
6.80.2.1	BrokerError	560
6.80.2.2	~BrokerError	560
6.80.3	Member Function Documentation	560
6.80.3.1	cloneDataStructure	560
6.80.3.2	copyDataStructure	560
6.80.3.3	getCause	560
6.80.3.4	getDataStructureType	561

6.80.3.5	getExceptionClass	561
6.80.3.6	getMessage	561
6.80.3.7	getStackTraceElements	561
6.80.3.8	setCause	561
6.80.3.9	setExceptionClass	562
6.80.3.10	setMessage	562
6.80.3.11	setStackTraceElements	562
6.80.3.12	visit	562
6.81	activemq::exceptions::BrokerException Class Reference	563
6.81.1	Constructor & Destructor Documentation	563
6.81.1.1	BrokerException	563
6.81.1.2	BrokerException	563
6.81.1.3	BrokerException	563
6.81.1.4	BrokerException	563
6.81.1.5	BrokerException	564
6.81.1.6	~BrokerException	564
6.81.2	Member Function Documentation	564
6.81.2.1	clone	564
6.82	activemq::commands::BrokerId Class Reference	564
6.82.1	Member Typedef Documentation	565
6.82.1.1	COMPARATOR	565
6.82.2	Constructor & Destructor Documentation	565
6.82.2.1	BrokerId	565
6.82.2.2	BrokerId	565
6.82.2.3	~BrokerId	565
6.82.3	Member Function Documentation	565
6.82.3.1	cloneDataStructure	565
6.82.3.2	compareTo	566
6.82.3.3	copyDataStructure	566
6.82.3.4	equals	566
6.82.3.5	equals	566
6.82.3.6	getDataStructureType	566
6.82.3.7	getValue	567
6.82.3.8	getValue	567

6.82.3.9	operator<	567
6.82.3.10	operator=	567
6.82.3.11	operator==	567
6.82.3.12	setValue	567
6.82.3.13	toString	567
6.82.4	Field Documentation	567
6.82.4.1	ID_BROKERID	567
6.82.4.2	value	567
6.83	activemq::wireformat::openwire::marshal::generated::BrokerIdMarshaller	
	Class Reference	567
6.83.1	Detailed Description	568
6.83.2	Constructor & Destructor Documentation	568
6.83.2.1	BrokerIdMarshaller	568
6.83.2.2	~BrokerIdMarshaller	569
6.83.3	Member Function Documentation	569
6.83.3.1	createObject	569
6.83.3.2	getDataStructureType	569
6.83.3.3	looseMarshal	569
6.83.3.4	looseUnmarshal	570
6.83.3.5	tightMarshal1	570
6.83.3.6	tightMarshal2	571
6.83.3.7	tightUnmarshal	571
6.84	activemq::commands::BrokerInfo Class Reference	572
6.84.1	Constructor & Destructor Documentation	573
6.84.1.1	BrokerInfo	573
6.84.1.2	~BrokerInfo	573
6.84.2	Member Function Documentation	574
6.84.2.1	cloneDataStructure	574
6.84.2.2	copyDataStructure	574
6.84.2.3	equals	574
6.84.2.4	getBrokerId	574
6.84.2.5	getBrokerId	574
6.84.2.6	getBrokerName	575
6.84.2.7	getBrokerName	575

6.84.2.8	getBrokerUploadUrl	575
6.84.2.9	getBrokerUploadUrl	575
6.84.2.10	getBrokerURL	575
6.84.2.11	getBrokerURL	575
6.84.2.12	getConnectionId	575
6.84.2.13	getDataStructureType	575
6.84.2.14	getNetworkProperties	575
6.84.2.15	getNetworkProperties	575
6.84.2.16	getPeerBrokerInfos	575
6.84.2.17	getPeerBrokerInfos	575
6.84.2.18	isBrokerInfo	575
6.84.2.19	isDuplexConnection	576
6.84.2.20	isFaultTolerantConfiguration	576
6.84.2.21	isMasterBroker	576
6.84.2.22	isNetworkConnection	576
6.84.2.23	isSlaveBroker	576
6.84.2.24	setBrokerId	576
6.84.2.25	setBrokerName	576
6.84.2.26	setBrokerUploadUrl	576
6.84.2.27	setBrokerURL	576
6.84.2.28	setConnectionId	576
6.84.2.29	setDuplexConnection	576
6.84.2.30	setFaultTolerantConfiguration	576
6.84.2.31	setMasterBroker	576
6.84.2.32	setNetworkConnection	576
6.84.2.33	setNetworkProperties	577
6.84.2.34	setPeerBrokerInfos	577
6.84.2.35	setSlaveBroker	577
6.84.2.36	toString	577
6.84.2.37	visit	577
6.84.3	Field Documentation	577
6.84.3.1	brokerId	577
6.84.3.2	brokerName	577
6.84.3.3	brokerUploadUrl	577

6.84.3.4	brokerURL	577
6.84.3.5	connectionId	578
6.84.3.6	duplexConnection	578
6.84.3.7	faultTolerantConfiguration	578
6.84.3.8	ID_BROKERINFO	578
6.84.3.9	masterBroker	578
6.84.3.10	networkConnection	578
6.84.3.11	networkProperties	578
6.84.3.12	peerBrokerInfos	578
6.84.3.13	slaveBroker	578
6.85	activemq::wireformat::openwire::marshal::generated::BrokerInfo- Marshaller Class Reference	578
6.85.1	Detailed Description	579
6.85.2	Constructor & Destructor Documentation	579
6.85.2.1	BrokerInfoMarshaller	579
6.85.2.2	~BrokerInfoMarshaller	579
6.85.3	Member Function Documentation	579
6.85.3.1	createObject	579
6.85.3.2	getDataStructureType	580
6.85.3.3	looseMarshal	580
6.85.3.4	looseUnmarshal	580
6.85.3.5	tightMarshal1	581
6.85.3.6	tightMarshal2	581
6.85.3.7	tightUnmarshal	582
6.86	decaf::nio::Buffer Class Reference	582
6.86.1	Detailed Description	583
6.86.2	Constructor & Destructor Documentation	585
6.86.2.1	Buffer	585
6.86.2.2	Buffer	585
6.86.2.3	~Buffer	585
6.86.3	Member Function Documentation	585
6.86.3.1	capacity	585
6.86.3.2	clear	585
6.86.3.3	flip	585

6.86.3.4	hasRemaining	586
6.86.3.5	isReadOnly	586
6.86.3.6	limit	586
6.86.3.7	limit	587
6.86.3.8	mark	587
6.86.3.9	position	587
6.86.3.10	position	587
6.86.3.11	remaining	588
6.86.3.12	reset	588
6.86.3.13	rewind	588
6.86.4	Field Documentation	589
6.86.4.1	_capacity	589
6.86.4.2	_limit	589
6.86.4.3	_mark	589
6.86.4.4	_markSet	589
6.86.4.5	_position	589
6.87	decaf::io::BufferedInputStream Class Reference	589
6.87.1	Detailed Description	591
6.87.2	Constructor & Destructor Documentation	591
6.87.2.1	BufferedInputStream	591
6.87.2.2	BufferedInputStream	591
6.87.2.3	~BufferedInputStream	592
6.87.3	Member Function Documentation	592
6.87.3.1	available	592
6.87.3.2	close	592
6.87.3.3	doReadArrayBounded	593
6.87.3.4	doReadByte	593
6.87.3.5	mark	593
6.87.3.6	markSupported	593
6.87.3.7	reset	594
6.87.3.8	skip	594
6.88	decaf::io::BufferedOutputStream Class Reference	595
6.88.1	Detailed Description	595
6.88.2	Constructor & Destructor Documentation	596

6.88.2.1	BufferedOutputStream	596
6.88.2.2	BufferedOutputStream	596
6.88.2.3	~BufferedOutputStream	596
6.88.3	Member Function Documentation	596
6.88.3.1	doWriteArray	596
6.88.3.2	doWriteArrayBounded	596
6.88.3.3	doWriteByte	597
6.88.3.4	flush	597
6.89	decaf::internal::nio::BufferFactory Class Reference	597
6.89.1	Detailed Description	599
6.89.2	Constructor & Destructor Documentation	599
6.89.2.1	~BufferFactory	599
6.89.3	Member Function Documentation	599
6.89.3.1	createByteBuffer	599
6.89.3.2	createByteBuffer	599
6.89.3.3	createByteBuffer	600
6.89.3.4	createCharBuffer	600
6.89.3.5	createCharBuffer	601
6.89.3.6	createCharBuffer	601
6.89.3.7	createDoubleBuffer	602
6.89.3.8	createDoubleBuffer	602
6.89.3.9	createDoubleBuffer	603
6.89.3.10	createFloatBuffer	603
6.89.3.11	createFloatBuffer	604
6.89.3.12	createFloatBuffer	604
6.89.3.13	createIntBuffer	605
6.89.3.14	createIntBuffer	605
6.89.3.15	createIntBuffer	606
6.89.3.16	createLongBuffer	606
6.89.3.17	createLongBuffer	606
6.89.3.18	createLongBuffer	607
6.89.3.19	createShortBuffer	607
6.89.3.20	createShortBuffer	608
6.89.3.21	createShortBuffer	608

6.90	decaf::nio::BufferOverflowException Class Reference	609
6.90.1	Constructor & Destructor Documentation	610
6.90.1.1	BufferOverflowException	610
6.90.1.2	BufferOverflowException	610
6.90.1.3	BufferOverflowException	610
6.90.1.4	BufferOverflowException	610
6.90.1.5	BufferOverflowException	610
6.90.1.6	BufferOverflowException	611
6.90.1.7	~BufferOverflowException	611
6.90.2	Member Function Documentation	611
6.90.2.1	clone	611
6.91	decaf::nio::BufferUnderflowException Class Reference	611
6.91.1	Constructor & Destructor Documentation	612
6.91.1.1	BufferUnderflowException	612
6.91.1.2	BufferUnderflowException	612
6.91.1.3	BufferUnderflowException	612
6.91.1.4	BufferUnderflowException	613
6.91.1.5	BufferUnderflowException	613
6.91.1.6	BufferUnderflowException	613
6.91.1.7	~BufferUnderflowException	613
6.91.2	Member Function Documentation	613
6.91.2.1	clone	614
6.92	decaf::lang::Byte Class Reference	614
6.92.1	Constructor & Destructor Documentation	616
6.92.1.1	Byte	616
6.92.1.2	Byte	616
6.92.1.3	~Byte	616
6.92.2	Member Function Documentation	616
6.92.2.1	byteValue	616
6.92.2.2	compareTo	616
6.92.2.3	compareTo	617
6.92.2.4	decode	617
6.92.2.5	doubleValue	618
6.92.2.6	equals	618

6.92.2.7	equals	618
6.92.2.8	floatValue	618
6.92.2.9	intValue	618
6.92.2.10	longValue	619
6.92.2.11	operator<	619
6.92.2.12	operator<	619
6.92.2.13	operator==	620
6.92.2.14	operator==	620
6.92.2.15	parseByte	620
6.92.2.16	parseByte	621
6.92.2.17	shortValue	621
6.92.2.18	toString	622
6.92.2.19	toString	622
6.92.2.20	valueOf	622
6.92.2.21	valueOf	622
6.92.2.22	valueOf	623
6.92.3	Field Documentation	623
6.92.3.1	MAX_VALUE	623
6.92.3.2	MIN_VALUE	623
6.92.3.3	SIZE	623
6.93	decaf::internal::util::ByteArrayAdapter Class Reference	624
6.93.1	Detailed Description	626
6.93.2	Constructor & Destructor Documentation	627
6.93.2.1	ByteArrayAdapter	627
6.93.2.2	ByteArrayAdapter	627
6.93.2.3	ByteArrayAdapter	627
6.93.2.4	ByteArrayAdapter	628
6.93.2.5	ByteArrayAdapter	628
6.93.2.6	ByteArrayAdapter	629
6.93.2.7	ByteArrayAdapter	629
6.93.2.8	ByteArrayAdapter	629
6.93.2.9	~ByteArrayAdapter	630
6.93.3	Member Function Documentation	630
6.93.3.1	clear	630

6.93.3.2	get	630
6.93.3.3	getByteArray	630
6.93.3.4	getCapacity	631
6.93.3.5	getChar	631
6.93.3.6	getCharArray	631
6.93.3.7	getCharCapacity	632
6.93.3.8	getDouble	632
6.93.3.9	getDoubleArray	632
6.93.3.10	getDoubleAt	632
6.93.3.11	getDoubleCapacity	633
6.93.3.12	getFloat	633
6.93.3.13	getFloatArray	634
6.93.3.14	getFloatAt	634
6.93.3.15	getFloatCapacity	634
6.93.3.16	getInt	634
6.93.3.17	getIntArray	635
6.93.3.18	getIntAt	635
6.93.3.19	getIntCapacity	636
6.93.3.20	getLong	636
6.93.3.21	getLongArray	636
6.93.3.22	getLongAt	636
6.93.3.23	getLongCapacity	637
6.93.3.24	getShort	637
6.93.3.25	getShortArray	638
6.93.3.26	getShortAt	638
6.93.3.27	getShortCapacity	638
6.93.3.28	operator[]	638
6.93.3.29	operator[]	639
6.93.3.30	put	639
6.93.3.31	putChar	639
6.93.3.32	putDouble	640
6.93.3.33	putDoubleAt	640
6.93.3.34	putFloat	641
6.93.3.35	putFloatAt	641

6.93.3.36	putInt	642
6.93.3.37	putIntAt	642
6.93.3.38	putLong	643
6.93.3.39	putLongAt	643
6.93.3.40	putShort	644
6.93.3.41	putShortAt	644
6.93.3.42	read	645
6.93.3.43	resize	645
6.93.3.44	write	646
6.94	decaf::internal::nio::ByteBuffer Class Reference	646
6.94.1	Detailed Description	658
6.94.2	Constructor & Destructor Documentation	659
6.94.2.1	ByteBuffer	659
6.94.2.2	ByteBuffer	659
6.94.2.3	ByteBuffer	660
6.94.2.4	ByteBuffer	660
6.94.2.5	~ByteBuffer	660
6.94.3	Member Function Documentation	660
6.94.3.1	array	661
6.94.3.2	arrayOffset	661
6.94.3.3	asCharBuffer	662
6.94.3.4	asDoubleBuffer	662
6.94.3.5	asFloatBuffer	662
6.94.3.6	asIntBuffer	663
6.94.3.7	asLongBuffer	663
6.94.3.8	asReadOnlyBuffer	663
6.94.3.9	asShortBuffer	664
6.94.3.10	compact	664
6.94.3.11	duplicate	665
6.94.3.12	get	665
6.94.3.13	get	666
6.94.3.14	getChar	666
6.94.3.15	getChar	666
6.94.3.16	getDouble	667

6.94.3.17	getDouble	667
6.94.3.18	getFloat	668
6.94.3.19	getFloat	668
6.94.3.20	getInt	669
6.94.3.21	getInt	669
6.94.3.22	getLong	669
6.94.3.23	getLong	670
6.94.3.24	getShort	670
6.94.3.25	getShort	671
6.94.3.26	hasArray	671
6.94.3.27	isReadOnly	671
6.94.3.28	put	672
6.94.3.29	put	672
6.94.3.30	putChar	673
6.94.3.31	putChar	673
6.94.3.32	putDouble	674
6.94.3.33	putDouble	674
6.94.3.34	putFloat	675
6.94.3.35	putFloat	675
6.94.3.36	putInt	676
6.94.3.37	putInt	676
6.94.3.38	putLong	677
6.94.3.39	putLong	677
6.94.3.40	putShort	678
6.94.3.41	putShort	678
6.94.3.42	setReadOnly	679
6.94.3.43	slice	679
6.95	decaf::io::ByteArrayInputStream Class Reference	679
6.95.1	Detailed Description	682
6.95.2	Constructor & Destructor Documentation	682
6.95.2.1	ByteArrayInputStream	682
6.95.2.2	ByteArrayInputStream	682
6.95.2.3	ByteArrayInputStream	682
6.95.2.4	ByteArrayInputStream	683

6.95.2.5	~ByteArrayInputStream	683
6.95.3	Member Function Documentation	683
6.95.3.1	available	683
6.95.3.2	doReadArrayBounded	684
6.95.3.3	doReadByte	684
6.95.3.4	mark	684
6.95.3.5	markSupported	685
6.95.3.6	reset	685
6.95.3.7	setByteArray	685
6.95.3.8	setByteArray	686
6.95.3.9	setByteArray	686
6.95.3.10	skip	687
6.96	decaf::io::ByteArrayOutputStream Class Reference	687
6.96.1	Constructor & Destructor Documentation	688
6.96.1.1	ByteArrayOutputStream	688
6.96.1.2	ByteArrayOutputStream	688
6.96.1.3	~ByteArrayOutputStream	689
6.96.2	Member Function Documentation	689
6.96.2.1	doWriteArrayBounded	689
6.96.2.2	doWriteByte	689
6.96.2.3	reset	689
6.96.2.4	size	689
6.96.2.5	toByteArray	689
6.96.2.6	toString	690
6.96.2.7	writeTo	690
6.97	decaf::nio::ByteBuffer Class Reference	690
6.97.1	Detailed Description	693
6.97.2	Constructor & Destructor Documentation	694
6.97.2.1	ByteBuffer	694
6.97.2.2	~ByteBuffer	695
6.97.3	Member Function Documentation	695
6.97.3.1	allocate	695
6.97.3.2	array	695
6.97.3.3	arrayOffset	696

6.97.3.4	asCharBuffer	696
6.97.3.5	asDoubleBuffer	697
6.97.3.6	asFloatBuffer	697
6.97.3.7	asIntBuffer	697
6.97.3.8	asLongBuffer	698
6.97.3.9	asReadOnlyBuffer	698
6.97.3.10	asShortBuffer	698
6.97.3.11	compact	699
6.97.3.12	compareTo	699
6.97.3.13	duplicate	699
6.97.3.14	equals	700
6.97.3.15	get	700
6.97.3.16	get	700
6.97.3.17	get	701
6.97.3.18	get	701
6.97.3.19	getChar	702
6.97.3.20	getChar	702
6.97.3.21	getDouble	703
6.97.3.22	getDouble	703
6.97.3.23	getFloat	703
6.97.3.24	getFloat	704
6.97.3.25	getInt	704
6.97.3.26	getInt	705
6.97.3.27	getLong	705
6.97.3.28	getLong	705
6.97.3.29	getShort	706
6.97.3.30	getShort	706
6.97.3.31	hasArray	707
6.97.3.32	isReadOnly	707
6.97.3.33	operator<	707
6.97.3.34	operator==	707
6.97.3.35	put	707
6.97.3.36	put	708
6.97.3.37	put	709

6.97.3.38 put	709
6.97.3.39 put	710
6.97.3.40 putChar	710
6.97.3.41 putChar	711
6.97.3.42 putDouble	711
6.97.3.43 putDouble	712
6.97.3.44 putFloat	712
6.97.3.45 putFloat	713
6.97.3.46 putInt	713
6.97.3.47 putInt	714
6.97.3.48 putLong	714
6.97.3.49 putLong	715
6.97.3.50 putShort	715
6.97.3.51 putShort	716
6.97.3.52 slice	716
6.97.3.53 toString	716
6.97.3.54 wrap	717
6.97.3.55 wrap	717
6.98 cms::BytesMessage Class Reference	718
6.98.1 Detailed Description	719
6.98.2 Constructor & Destructor Documentation	720
6.98.2.1 ~BytesMessage	720
6.98.3 Member Function Documentation	720
6.98.3.1 clone	720
6.98.3.2 getBodyBytes	721
6.98.3.3 getBodyLength	721
6.98.3.4 readBoolean	722
6.98.3.5 readByte	722
6.98.3.6 readBytes	723
6.98.3.7 readBytes	723
6.98.3.8 readChar	724
6.98.3.9 readDouble	725
6.98.3.10 readFloat	725
6.98.3.11 readInt	725

6.98.3.12 readLong	726
6.98.3.13 readShort	726
6.98.3.14 readString	727
6.98.3.15 readUnsignedShort	727
6.98.3.16 readUTF	728
6.98.3.17 reset	728
6.98.3.18 setBodyBytes	729
6.98.3.19 writeBoolean	729
6.98.3.20 writeByte	729
6.98.3.21 writeBytes	730
6.98.3.22 writeBytes	730
6.98.3.23 writeChar	731
6.98.3.24 writeDouble	731
6.98.3.25 writeFloat	732
6.98.3.26 writeInt	732
6.98.3.27 writeLong	732
6.98.3.28 writeShort	733
6.98.3.29 writeString	733
6.98.3.30 writeUnsignedShort	734
6.98.3.31 writeUTF	734
6.99 activemq::cmsutil::CachedConsumer Class Reference	734
6.99.1 Detailed Description	735
6.99.2 Constructor & Destructor Documentation	735
6.99.2.1 CachedConsumer	735
6.99.2.2 ~CachedConsumer	735
6.99.3 Member Function Documentation	735
6.99.3.1 close	736
6.99.3.2 getMessageListener	736
6.99.3.3 getMessageSelector	736
6.99.3.4 receive	736
6.99.3.5 receive	737
6.99.3.6 receiveNoWait	737
6.99.3.7 setMessageListener	737
6.99.3.8 start	738

6.99.3.9 stop	738
6.100activemq::cmsutil::CachedProducer Class Reference	738
6.100.1 Detailed Description	739
6.100.2 Constructor & Destructor Documentation	740
6.100.2.1 CachedProducer	740
6.100.2.2 ~CachedProducer	740
6.100.3 Member Function Documentation	740
6.100.3.1 close	740
6.100.3.2 getDeliveryMode	740
6.100.3.3 getDisableMessageID	740
6.100.3.4 getDisableMessageTimeStamp	741
6.100.3.5 getPriority	741
6.100.3.6 getTimeToLive	741
6.100.3.7 send	742
6.100.3.8 send	742
6.100.3.9 send	743
6.100.3.10send	743
6.100.3.11setDeliveryMode	744
6.100.3.12setDisableMessageID	744
6.100.3.13setDisableMessageTimeStamp	744
6.100.3.14setPriority	745
6.100.3.15setTimeToLive	745
6.101decaf::util::concurrent::Callable< V > Class Template Reference	746
6.101.1 Detailed Description	746
6.101.2 Constructor & Destructor Documentation	746
6.101.2.1 ~Callable	746
6.101.3 Member Function Documentation	746
6.101.3.1 call	746
6.102decaf::util::concurrent::ThreadPoolExecutor::CallerRunsPolicy Class Reference	747
6.102.1 Detailed Description	747
6.102.2 Constructor & Destructor Documentation	748
6.102.2.1 CallerRunsPolicy	748
6.102.2.2 ~CallerRunsPolicy	748

6.102.3 Member Function Documentation	748
6.102.3.1 rejectedExecution	748
6.103decaf::util::concurrent::CancellationException Class Reference	748
6.103.1 Constructor & Destructor Documentation	749
6.103.1.1 CancellationException	749
6.103.1.2 CancellationException	749
6.103.1.3 CancellationException	749
6.103.1.4 CancellationException	749
6.103.1.5 CancellationException	750
6.103.1.6 CancellationException	750
6.103.1.7 ~CancellationException	750
6.103.2 Member Function Documentation	750
6.103.2.1 clone	750
6.104decaf::security::cert::Certificate Class Reference	751
6.104.1 Detailed Description	751
6.104.2 Constructor & Destructor Documentation	752
6.104.2.1 ~Certificate	752
6.104.3 Member Function Documentation	752
6.104.3.1 equals	752
6.104.3.2 getEncoded	752
6.104.3.3 getPublicKey	752
6.104.3.4 getPublicKey	753
6.104.3.5 getType	753
6.104.3.6 toString	753
6.104.3.7 verify	753
6.104.3.8 verify	754
6.105decaf::security::cert::CertificateEncodingException Class Reference	755
6.105.1 Constructor & Destructor Documentation	755
6.105.1.1 CertificateEncodingException	755
6.105.1.2 CertificateEncodingException	755
6.105.1.3 CertificateEncodingException	756
6.105.1.4 CertificateEncodingException	756
6.105.1.5 ~CertificateEncodingException	756
6.105.2 Member Function Documentation	756

6.105.2.1 clone	756
6.106decaf::security::cert::CertificateException Class Reference	757
6.106.1 Constructor & Destructor Documentation	757
6.106.1.1 CertificateException	757
6.106.1.2 CertificateException	757
6.106.1.3 CertificateException	757
6.106.1.4 CertificateException	758
6.106.1.5 ~CertificateException	758
6.106.2 Member Function Documentation	758
6.106.2.1 clone	758
6.107decaf::security::cert::CertificateExpiredException Class Reference	759
6.107.1 Constructor & Destructor Documentation	759
6.107.1.1 CertificateExpiredException	759
6.107.1.2 CertificateExpiredException	759
6.107.1.3 CertificateExpiredException	760
6.107.1.4 CertificateExpiredException	760
6.107.1.5 ~CertificateExpiredException	760
6.107.2 Member Function Documentation	760
6.107.2.1 clone	760
6.108decaf::security::cert::CertificateNotYetValidException Class Reference	761
6.108.1 Constructor & Destructor Documentation	761
6.108.1.1 CertificateNotYetValidException	761
6.108.1.2 CertificateNotYetValidException	761
6.108.1.3 CertificateNotYetValidException	762
6.108.1.4 CertificateNotYetValidException	762
6.108.1.5 ~CertificateNotYetValidException	762
6.108.2 Member Function Documentation	762
6.108.2.1 clone	762
6.109decaf::security::cert::CertificateParsingException Class Reference	763
6.109.1 Constructor & Destructor Documentation	763
6.109.1.1 CertificateParsingException	763
6.109.1.2 CertificateParsingException	763
6.109.1.3 CertificateParsingException	764
6.109.1.4 CertificateParsingException	764

6.109.1.5 ~CertificateParsingException	764
6.109.2 Member Function Documentation	764
6.109.2.1 clone	764
6.110decaf::lang::Character Class Reference	765
6.110.1 Constructor & Destructor Documentation	766
6.110.1.1 Character	766
6.110.2 Member Function Documentation	767
6.110.2.1 byteValue	767
6.110.2.2 compareTo	767
6.110.2.3 compareTo	767
6.110.2.4 digit	768
6.110.2.5 doubleValue	768
6.110.2.6 equals	768
6.110.2.7 equals	768
6.110.2.8 floatValue	769
6.110.2.9 intValue	769
6.110.2.10sDigit	769
6.110.2.11isISOControl	769
6.110.2.12sLetter	769
6.110.2.13sLetterOrDigit	770
6.110.2.14sLowerCase	770
6.110.2.15sUpperCase	770
6.110.2.16sWhitespace	770
6.110.2.17longValue	770
6.110.2.18operator<	770
6.110.2.19operator<	771
6.110.2.20operator==	771
6.110.2.21operator==	771
6.110.2.22shortValue	772
6.110.2.23toString	772
6.110.2.24valueOf	772
6.110.3 Field Documentation	772
6.110.3.1 MAX_RADIX	772
6.110.3.2 MAX_VALUE	772

6.110.3.3 MIN_RADIX	773
6.110.3.4 MIN_VALUE	773
6.110.3.5 SIZE	773
6.111decaf::internal::nio::CharArrayBuffer Class Reference	773
6.111.1 Constructor & Destructor Documentation	777
6.111.1.1 CharArrayBuffer	777
6.111.1.2 CharArrayBuffer	778
6.111.1.3 CharArrayBuffer	778
6.111.1.4 CharArrayBuffer	779
6.111.1.5 ~CharArrayBuffer	779
6.111.2 Member Function Documentation	779
6.111.2.1 array	779
6.111.2.2 arrayOffset	779
6.111.2.3 asReadOnlyBuffer	780
6.111.2.4 compact	780
6.111.2.5 duplicate	781
6.111.2.6 get	781
6.111.2.7 get	782
6.111.2.8 hasArray	782
6.111.2.9 isReadOnly	782
6.111.2.10put	783
6.111.2.11put	783
6.111.2.12setReadOnly	784
6.111.2.13slice	784
6.111.2.14subSequence	784
6.111.3 Field Documentation	785
6.111.3.1 _array	785
6.111.3.2 length	785
6.111.3.3 offset	785
6.111.3.4 readOnly	785
6.112decaf::nio::CharBuffer Class Reference	785
6.112.1 Detailed Description	787
6.112.2 Constructor & Destructor Documentation	788
6.112.2.1 CharBuffer	788

6.112.2.2 ~CharBuffer	788
6.112.3 Member Function Documentation	788
6.112.3.1 allocate	788
6.112.3.2 append	789
6.112.3.3 append	789
6.112.3.4 append	790
6.112.3.5 array	790
6.112.3.6 arrayOffset	791
6.112.3.7 asReadOnlyBuffer	791
6.112.3.8 charAt	792
6.112.3.9 compact	792
6.112.3.10compareTo	793
6.112.3.11duplicate	793
6.112.3.12equals	793
6.112.3.13get	793
6.112.3.14get	794
6.112.3.15get	794
6.112.3.16get	794
6.112.3.17hasArray	795
6.112.3.18length	795
6.112.3.19operator<	796
6.112.3.20operator==	796
6.112.3.21put	796
6.112.3.22put	796
6.112.3.23put	797
6.112.3.24put	798
6.112.3.25put	798
6.112.3.26put	799
6.112.3.27put	800
6.112.3.28read	800
6.112.3.29slice	801
6.112.3.30subSequence	801
6.112.3.31toString	802
6.112.3.32wrap	802

6.112.3.33wrap	802
6.113decaf::lang::CharSequence Class Reference	803
6.113.1 Detailed Description	803
6.113.2 Constructor & Destructor Documentation	804
6.113.2.1 ~CharSequence	804
6.113.3 Member Function Documentation	804
6.113.3.1 charAt	804
6.113.3.2 length	804
6.113.3.3 subSequence	804
6.113.3.4 toString	805
6.114decaf::util::zip::CheckedInputStream Class Reference	805
6.114.1 Detailed Description	806
6.114.2 Constructor & Destructor Documentation	806
6.114.2.1 CheckedInputStream	806
6.114.2.2 ~CheckedInputStream	807
6.114.3 Member Function Documentation	807
6.114.3.1 doReadArrayBounded	807
6.114.3.2 doReadByte	807
6.114.3.3 getChecksum	807
6.114.3.4 skip	807
6.115decaf::util::zip::CheckedOutputStream Class Reference	808
6.115.1 Detailed Description	809
6.115.2 Constructor & Destructor Documentation	809
6.115.2.1 CheckedOutputStream	809
6.115.2.2 ~CheckedOutputStream	809
6.115.3 Member Function Documentation	809
6.115.3.1 doWriteArrayBounded	809
6.115.3.2 doWriteByte	810
6.115.3.3 getChecksum	810
6.116decaf::util::zip::Checksum Class Reference	810
6.116.1 Detailed Description	811
6.116.2 Constructor & Destructor Documentation	811
6.116.2.1 ~Checksum	811
6.116.3 Member Function Documentation	811

6.116.3.1	getValue	811
6.116.3.2	reset	811
6.116.3.3	update	811
6.116.3.4	update	811
6.116.3.5	update	812
6.116.3.6	update	812
6.117	decaf::lang::exceptions::ClassCastException Class Reference	813
6.117.1	Constructor & Destructor Documentation	813
6.117.1.1	ClassCastException	813
6.117.1.2	ClassCastException	813
6.117.1.3	ClassCastException	814
6.117.1.4	ClassCastException	814
6.117.1.5	ClassCastException	814
6.117.1.6	ClassCastException	814
6.117.1.7	~ClassCastException	815
6.117.2	Member Function Documentation	815
6.117.2.1	clone	815
6.118	cms::Closeable Class Reference	815
6.118.1	Detailed Description	816
6.118.2	Constructor & Destructor Documentation	816
6.118.2.1	~Closeable	816
6.118.3	Member Function Documentation	816
6.118.3.1	close	816
6.119	decaf::io::Closeable Class Reference	816
6.119.1	Detailed Description	817
6.119.2	Constructor & Destructor Documentation	817
6.119.2.1	~Closeable	817
6.119.3	Member Function Documentation	817
6.119.3.1	close	817
6.120	activemq::transport::failover::CloseTransportsTask Class Reference	818
6.120.1	Constructor & Destructor Documentation	818
6.120.1.1	CloseTransportsTask	818
6.120.1.2	~CloseTransportsTask	818
6.120.2	Member Function Documentation	818

6.120.2.1 add	818
6.120.2.2 isPending	818
6.120.2.3 iterate	819
6.121activemq::cmsutil::CmsAccessor Class Reference	819
6.121.1 Detailed Description	820
6.121.2 Constructor & Destructor Documentation	820
6.121.2.1 CmsAccessor	820
6.121.2.2 CmsAccessor	820
6.121.2.3 ~CmsAccessor	820
6.121.3 Member Function Documentation	820
6.121.3.1 checkConnectionFactory	820
6.121.3.2 createConnection	821
6.121.3.3 createSession	821
6.121.3.4 destroy	821
6.121.3.5 getConnectionFactory	822
6.121.3.6 getConnectionFactory	822
6.121.3.7 getResourceLifecycleManager	822
6.121.3.8 getResourceLifecycleManager	822
6.121.3.9 getSessionAcknowledgeMode	822
6.121.3.10init	822
6.121.3.11operator=	823
6.121.3.12setConnectionFactory	823
6.121.3.13setSessionAcknowledgeMode	823
6.122activemq::cmsutil::CmsDestinationAccessor Class Reference	823
6.122.1 Detailed Description	824
6.122.2 Constructor & Destructor Documentation	824
6.122.2.1 CmsDestinationAccessor	824
6.122.2.2 ~CmsDestinationAccessor	824
6.122.3 Member Function Documentation	824
6.122.3.1 checkDestinationResolver	824
6.122.3.2 destroy	824
6.122.3.3 getDestinationResolver	825
6.122.3.4 getDestinationResolver	825
6.122.3.5 init	825

6.122.3.6 isPubSubDomain	825
6.122.3.7 resolveDestinationName	825
6.122.3.8 setDestinationResolver	825
6.122.3.9 setPubSubDomain	825
6.123cms::CMSException Class Reference	826
6.123.1 Detailed Description	826
6.123.2 Constructor & Destructor Documentation	827
6.123.2.1 CMSException	827
6.123.2.2 CMSException	827
6.123.2.3 CMSException	827
6.123.2.4 CMSException	827
6.123.2.5 CMSException	827
6.123.2.6 ~CMSException	827
6.123.3 Member Function Documentation	827
6.123.3.1 getCause	827
6.123.3.2 getMessage	827
6.123.3.3 getStackTrace	828
6.123.3.4 getStackTraceString	828
6.123.3.5 printStackTrace	828
6.123.3.6 printStackTrace	828
6.123.3.7 setMark	828
6.123.3.8 what	829
6.124activemq::util::CMSExceptionSupport Class Reference	829
6.124.1 Constructor & Destructor Documentation	829
6.124.1.1 ~CMSExceptionSupport	829
6.124.2 Member Function Documentation	829
6.124.2.1 create	829
6.124.2.2 create	829
6.124.2.3 createMessageEOFException	830
6.124.2.4 createMessageFormatException	830
6.125cms::CMSProperties Class Reference	830
6.125.1 Detailed Description	831
6.125.2 Constructor & Destructor Documentation	831
6.125.2.1 ~CMSProperties	831

6.125.3 Member Function Documentation	831
6.125.3.1 clear	831
6.125.3.2 clone	831
6.125.3.3 copy	832
6.125.3.4 getProperty	832
6.125.3.5 getProperty	832
6.125.3.6 hasProperty	833
6.125.3.7 isEmpty	833
6.125.3.8 propertyNames	833
6.125.3.9 remove	833
6.125.3.10 setProperty	834
6.125.3.11 size	834
6.125.3.12 toArray	834
6.125.3.13 toString	834
6.126 cms::CMSSecurityException Class Reference	835
6.126.1 Detailed Description	835
6.126.2 Constructor & Destructor Documentation	835
6.126.2.1 CMSSecurityException	835
6.126.2.2 CMSSecurityException	836
6.126.2.3 CMSSecurityException	836
6.126.2.4 CMSSecurityException	836
6.126.2.5 CMSSecurityException	836
6.126.2.6 ~CMSSecurityException	836
6.127 activemq::cmsutil::CmsTemplate Class Reference	836
6.127.1 Detailed Description	839
6.127.2 Constructor & Destructor Documentation	839
6.127.2.1 CmsTemplate	839
6.127.2.2 CmsTemplate	839
6.127.2.3 ~CmsTemplate	839
6.127.3 Member Function Documentation	840
6.127.3.1 destroy	840
6.127.3.2 execute	840
6.127.3.3 execute	840
6.127.3.4 execute	841

6.127.3.5 execute	841
6.127.3.6 getDefaultDestination	841
6.127.3.7 getDefaultDestination	842
6.127.3.8 getDefaultDestinationName	842
6.127.3.9 getDeliveryMode	842
6.127.3.10getPriority	842
6.127.3.11getReceiveTimeout	842
6.127.3.12getTimeToLive	842
6.127.3.13nit	843
6.127.3.14sExplicitQosEnabled	843
6.127.3.15sMessageIdEnabled	843
6.127.3.16sMessageTimestampEnabled	843
6.127.3.17sNoLocal	843
6.127.3.18receive	843
6.127.3.19receive	844
6.127.3.20receive	844
6.127.3.21receiveSelected	845
6.127.3.22receiveSelected	845
6.127.3.23receiveSelected	846
6.127.3.24send	846
6.127.3.25send	846
6.127.3.26send	847
6.127.3.27setDefaultDestination	847
6.127.3.28setDefaultDestinationName	848
6.127.3.29setDeliveryMode	848
6.127.3.30setDeliveryPersistent	848
6.127.3.31setExplicitQosEnabled	849
6.127.3.32setMessageIdEnabled	849
6.127.3.33setMessageTimestampEnabled	849
6.127.3.34setNoLocal	849
6.127.3.35setPriority	849
6.127.3.36setPubSubDomain	849
6.127.3.37setReceiveTimeout	850
6.127.3.38setTimeToLive	850

6.127.4 Friends And Related Function Documentation	850
6.127.4.1 ProducerExecutor	850
6.127.4.2 ReceiveExecutor	850
6.127.4.3 ResolveProducerExecutor	850
6.127.4.4 ResolveReceiveExecutor	850
6.127.4.5 SendExecutor	850
6.127.5 Field Documentation	850
6.127.5.1 DEFAULT_PRIORITY	850
6.127.5.2 DEFAULT_TIME_TO_LIVE	850
6.127.5.3 RECEIVE_TIMEOUT_INDEFINITE_WAIT	850
6.127.5.4 RECEIVE_TIMEOUT_NO_WAIT	851
6.128code Struct Reference	851
6.128.1 Field Documentation	851
6.128.1.1 bits	851
6.128.1.2 op	851
6.128.1.3 val	851
6.129decaf::util::Collection< E > Class Template Reference	851
6.129.1 Detailed Description	852
6.129.2 Constructor & Destructor Documentation	853
6.129.2.1 ~Collection	853
6.129.3 Member Function Documentation	853
6.129.3.1 add	853
6.129.3.2 addAll	855
6.129.3.3 clear	856
6.129.3.4 contains	857
6.129.3.5 containsAll	858
6.129.3.6 copy	859
6.129.3.7 equals	860
6.129.3.8 isEmpty	860
6.129.3.9 remove	861
6.129.3.10removeAll	862
6.129.3.11retainAll	863
6.129.3.12size	864
6.129.3.13toArray	865

6.130	activemq::commands::Command Class Reference	866
6.130.1	Constructor & Destructor Documentation	867
6.130.1.1	~Command	867
6.130.2	Member Function Documentation	867
6.130.2.1	getCommandId	867
6.130.2.2	isBrokerInfo	867
6.130.2.3	isConnectionControl	867
6.130.2.4	isConnectionInfo	867
6.130.2.5	isConsumerInfo	868
6.130.2.6	isKeepAliveInfo	868
6.130.2.7	isMessage	868
6.130.2.8	isMessageAck	868
6.130.2.9	isMessageDispatch	868
6.130.2.10	isMessageDispatchNotification	868
6.130.2.11	isProducerAck	868
6.130.2.12	isProducerInfo	869
6.130.2.13	isRemoveInfo	869
6.130.2.14	isRemoveSubscriptionInfo	869
6.130.2.15	isResponse	869
6.130.2.16	isResponseRequired	869
6.130.2.17	isShutdownInfo	869
6.130.2.18	isTransactionInfo	869
6.130.2.19	isWireFormatInfo	870
6.130.2.20	setCommandId	870
6.130.2.21	setResponseRequired	870
6.130.2.22	toString	870
6.130.2.23	visit	871
6.131	activemq::state::CommandVisitor Class Reference	872
6.131.1	Detailed Description	874
6.131.2	Constructor & Destructor Documentation	874
6.131.2.1	~CommandVisitor	874
6.131.3	Member Function Documentation	874
6.131.3.1	processBeginTransaction	874
6.131.3.2	processBrokerError	874

6.131.3.3 processBrokerInfo	874
6.131.3.4 processCommitTransactionOnePhase	874
6.131.3.5 processCommitTransactionTwoPhase	874
6.131.3.6 processConnectionControl	875
6.131.3.7 processConnectionError	875
6.131.3.8 processConnectionInfo	875
6.131.3.9 processConsumerControl	875
6.131.3.10 processConsumerInfo	875
6.131.3.11 processControlCommand	875
6.131.3.12 processDestinationInfo	875
6.131.3.13 processEndTransaction	875
6.131.3.14 processFlushCommand	875
6.131.3.15 processForgetTransaction	876
6.131.3.16 processKeepAliveInfo	876
6.131.3.17 processMessage	876
6.131.3.18 processMessageAck	876
6.131.3.19 processMessageDispatch	876
6.131.3.20 processMessageDispatchNotification	876
6.131.3.21 processMessagePull	876
6.131.3.22 processPrepareTransaction	876
6.131.3.23 processProducerAck	876
6.131.3.24 processProducerInfo	877
6.131.3.25 processRecoverTransactions	877
6.131.3.26 processRemoveConnection	877
6.131.3.27 processRemoveConsumer	877
6.131.3.28 processRemoveDestination	877
6.131.3.29 processRemoveInfo	877
6.131.3.30 processRemoveProducer	877
6.131.3.31 processRemoveSession	877
6.131.3.32 processRemoveSubscriptionInfo	878
6.131.3.33 processReplayCommand	878
6.131.3.34 processResponse	878
6.131.3.35 processRollbackTransaction	878
6.131.3.36 processSessionInfo	878

6.131.3.37	processShutdownInfo	878
6.131.3.38	processTransactionInfo	878
6.131.3.39	processWireFormat	878
6.132	activemq::state::CommandVisitorAdapter Class Reference	878
6.132.1	Detailed Description	881
6.132.2	Constructor & Destructor Documentation	881
6.132.2.1	~CommandVisitorAdapter	881
6.132.3	Member Function Documentation	881
6.132.3.1	processBeginTransaction	881
6.132.3.2	processBrokerError	881
6.132.3.3	processBrokerInfo	881
6.132.3.4	processCommitTransactionOnePhase	881
6.132.3.5	processCommitTransactionTwoPhase	881
6.132.3.6	processConnectionControl	881
6.132.3.7	processConnectionError	882
6.132.3.8	processConnectionInfo	882
6.132.3.9	processConsumerControl	882
6.132.3.10	processConsumerInfo	882
6.132.3.11	processControlCommand	882
6.132.3.12	processDestinationInfo	882
6.132.3.13	processEndTransaction	882
6.132.3.14	processFlushCommand	882
6.132.3.15	processForgetTransaction	882
6.132.3.16	processKeepAliveInfo	883
6.132.3.17	processMessage	883
6.132.3.18	processMessageAck	883
6.132.3.19	processMessageDispatch	883
6.132.3.20	processMessageDispatchNotification	883
6.132.3.21	processMessagePull	883
6.132.3.22	processPrepareTransaction	883
6.132.3.23	processProducerAck	883
6.132.3.24	processProducerInfo	883
6.132.3.25	processRecoverTransactions	883
6.132.3.26	processRemoveConnection	884

6.132.3.27	processRemoveConsumer	884
6.132.3.28	processRemoveDestination	884
6.132.3.29	processRemoveInfo	884
6.132.3.30	processRemoveProducer	884
6.132.3.31	processRemoveSession	884
6.132.3.32	processRemoveSubscriptionInfo	884
6.132.3.33	processReplayCommand	884
6.132.3.34	processResponse	884
6.132.3.35	processRollbackTransaction	885
6.132.3.36	processSessionInfo	885
6.132.3.37	processShutdownInfo	885
6.132.3.38	processTransactionInfo	885
6.132.3.39	processWireFormat	885
6.133	decaf::lang::Comparable< T > Class Template Reference	885
6.133.1	Detailed Description	886
6.133.2	Constructor & Destructor Documentation	886
6.133.2.1	~Comparable	886
6.133.3	Member Function Documentation	886
6.133.3.1	compareTo	886
6.133.3.2	equals	887
6.133.3.3	operator<	887
6.133.3.4	operator==	888
6.134	decaf::util::Comparator< T > Class Template Reference	888
6.134.1	Detailed Description	889
6.134.2	Constructor & Destructor Documentation	889
6.134.2.1	~Comparator	889
6.134.3	Member Function Documentation	889
6.134.3.1	compare	889
6.134.3.2	operator()	890
6.135	activemq::util::CompositeData Class Reference	890
6.135.1	Detailed Description	891
6.135.2	Constructor & Destructor Documentation	891
6.135.2.1	CompositeData	891
6.135.2.2	~CompositeData	891

6.135.3 Member Function Documentation	891
6.135.3.1 getComponents	891
6.135.3.2 getComponents	891
6.135.3.3 getFragment	891
6.135.3.4 getHost	891
6.135.3.5 getParameters	891
6.135.3.6 getPath	891
6.135.3.7 getScheme	891
6.135.3.8 setComponents	891
6.135.3.9 setFragment	891
6.135.3.10 setHost	892
6.135.3.11 setParameters	892
6.135.3.12 setPath	892
6.135.3.13 setScheme	892
6.135.3.14 toURI	892
6.136 activemq::threads::CompositeTask Class Reference	892
6.136.1 Detailed Description	892
6.136.2 Constructor & Destructor Documentation	893
6.136.2.1 ~CompositeTask	893
6.136.3 Member Function Documentation	893
6.136.3.1 isPending	893
6.137 activemq::threads::CompositeTaskRunner Class Reference	893
6.137.1 Detailed Description	894
6.137.2 Constructor & Destructor Documentation	894
6.137.2.1 CompositeTaskRunner	894
6.137.2.2 ~CompositeTaskRunner	894
6.137.3 Member Function Documentation	894
6.137.3.1 addTask	894
6.137.3.2 iterate	895
6.137.3.3 removeTask	895
6.137.3.4 run	895
6.137.3.5 shutdown	895
6.137.3.6 shutdown	896
6.137.3.7 wakeup	896

6.138activemq::transport::CompositeTransport Class Reference	896
6.138.1 Detailed Description	896
6.138.2 Constructor & Destructor Documentation	897
6.138.2.1 ~CompositeTransport	897
6.138.3 Member Function Documentation	897
6.138.3.1 addURI	897
6.138.3.2 removeURI	897
6.139decaf::util::concurrent::ConcurrentMap< K, V, COMPARATOR > Class Template Reference	898
6.139.1 Detailed Description	898
6.139.2 Constructor & Destructor Documentation	898
6.139.2.1 ~ConcurrentMap	898
6.139.3 Member Function Documentation	898
6.139.3.1 putIfAbsent	899
6.139.3.2 remove	900
6.139.3.3 replace	900
6.139.3.4 replace	901
6.140decaf::util::ConcurrentModificationException Class Reference	902
6.140.1 Constructor & Destructor Documentation	903
6.140.1.1 ConcurrentModificationException	903
6.140.1.2 ConcurrentModificationException	903
6.140.1.3 ConcurrentModificationException	903
6.140.1.4 ConcurrentModificationException	903
6.140.1.5 ConcurrentModificationException	904
6.140.1.6 ConcurrentModificationException	904
6.140.1.7 ~ConcurrentModificationException	904
6.140.2 Member Function Documentation	904
6.140.2.1 clone	904
6.141decaf::util::concurrent::ConcurrentStlMap< K, V, COMPARATOR > - Class Template Reference	905
6.141.1 Detailed Description	908
6.141.2 Constructor & Destructor Documentation	909
6.141.2.1 ConcurrentStlMap	909
6.141.2.2 ConcurrentStlMap	909

6.141.2.3 ConcurrentStlMap	909
6.141.2.4 ~ConcurrentStlMap	909
6.141.3 Member Function Documentation	909
6.141.3.1 clear	910
6.141.3.2 containsKey	910
6.141.3.3 containsValue	910
6.141.3.4 copy	911
6.141.3.5 copy	911
6.141.3.6 equals	911
6.141.3.7 equals	911
6.141.3.8 get	912
6.141.3.9 get	912
6.141.3.10 isEmpty	913
6.141.3.11 keySet	913
6.141.3.12 lock	913
6.141.3.13 notify	914
6.141.3.14 notifyAll	914
6.141.3.15 put	914
6.141.3.16 putAll	915
6.141.3.17 putAll	915
6.141.3.18 putIfAbsent	915
6.141.3.19 remove	916
6.141.3.20 remove	917
6.141.3.21 replace	917
6.141.3.22 replace	918
6.141.3.23 size	919
6.141.3.24 tryLock	919
6.141.3.25 unlock	919
6.141.3.26 values	919
6.141.3.27 wait	920
6.141.3.28 wait	920
6.141.3.29 wait	920
6.142 decaf::util::concurrent::locks::Condition Class Reference	921
6.142.1 Detailed Description	922

6.142.2 Constructor & Destructor Documentation	923
6.142.2.1 ~Condition	923
6.142.3 Member Function Documentation	923
6.142.3.1 await	923
6.142.3.2 await	924
6.142.3.3 awaitNanos	925
6.142.3.4 awaitUninterruptibly	926
6.142.3.5 awaitUntil	927
6.142.3.6 signal	927
6.142.3.7 signalAll	927
6.143decaf::util::concurrent::ConditionHandle Class Reference	928
6.143.1 Constructor & Destructor Documentation	928
6.143.1.1 ConditionHandle	928
6.143.1.2 ~ConditionHandle	928
6.143.1.3 ConditionHandle	928
6.143.1.4 ~ConditionHandle	928
6.143.2 Field Documentation	928
6.143.2.1 condition	928
6.143.2.2 criticalSection	928
6.143.2.3 generation	928
6.143.2.4 mutex	928
6.143.2.5 numWaiting	929
6.143.2.6 numWake	929
6.143.2.7 semaphore	929
6.144decaf::internal::util::concurrent::ConditionImpl Class Reference	929
6.144.1 Member Function Documentation	929
6.144.1.1 create	929
6.144.1.2 destroy	930
6.144.1.3 notify	930
6.144.1.4 notifyAll	930
6.144.1.5 wait	930
6.144.1.6 wait	931
6.145decaf::net::ConnectException Class Reference	931
6.145.1 Constructor & Destructor Documentation	932

6.145.1.1	ConnectException	932
6.145.1.2	ConnectException	932
6.145.1.3	ConnectException	932
6.145.1.4	ConnectException	932
6.145.1.5	ConnectException	932
6.145.1.6	ConnectException	933
6.145.1.7	~ConnectException	933
6.145.2	Member Function Documentation	933
6.145.2.1	clone	933
6.146	cms::Connection Class Reference	933
6.146.1	Detailed Description	934
6.146.2	Constructor & Destructor Documentation	935
6.146.2.1	~Connection	935
6.146.3	Member Function Documentation	935
6.146.3.1	close	935
6.146.3.2	createSession	935
6.146.3.3	createSession	936
6.146.3.4	getClientID	936
6.146.3.5	getExceptionListener	936
6.146.3.6	getMetaData	937
6.146.3.7	setClientID	937
6.146.3.8	setExceptionListener	938
6.147	activemq::commands::ConnectionControl Class Reference	938
6.147.1	Constructor & Destructor Documentation	940
6.147.1.1	ConnectionControl	940
6.147.1.2	~ConnectionControl	940
6.147.2	Member Function Documentation	940
6.147.2.1	cloneDataStructure	940
6.147.2.2	copyDataStructure	940
6.147.2.3	equals	940
6.147.2.4	getConnectedBrokers	941
6.147.2.5	getConnectedBrokers	941
6.147.2.6	getDataStructureType	941
6.147.2.7	getReconnectTo	941

6.147.2.8	getReconnectTo	941
6.147.2.9	isClose	941
6.147.2.10	sConnectionControl	941
6.147.2.11	isExit	941
6.147.2.12	sFaultTolerant	941
6.147.2.13	sRebalanceConnection	941
6.147.2.14	sResume	941
6.147.2.15	sSuspend	942
6.147.2.16	setClose	942
6.147.2.17	setConnectedBrokers	942
6.147.2.18	setExit	942
6.147.2.19	setFaultTolerant	942
6.147.2.20	setRebalanceConnection	942
6.147.2.21	setReconnectTo	942
6.147.2.22	setResume	942
6.147.2.23	setSuspend	942
6.147.2.24	toString	942
6.147.2.25	visit	942
6.147.3	Field Documentation	943
6.147.3.1	close	943
6.147.3.2	connectedBrokers	943
6.147.3.3	exit	943
6.147.3.4	faultTolerant	943
6.147.3.5	ID_CONNECTIONCONTROL	943
6.147.3.6	rebalanceConnection	943
6.147.3.7	reconnectTo	943
6.147.3.8	resume	943
6.147.3.9	suspend	943
6.148	activemq::wireformat::openwire::marshal::generated::Connection- ControlMarshaller Class Reference	943
6.148.1	Detailed Description	944
6.148.2	Constructor & Destructor Documentation	944
6.148.2.1	ConnectionControlMarshaller	944
6.148.2.2	~ConnectionControlMarshaller	945

6.148.3 Member Function Documentation	945
6.148.3.1 createObject	945
6.148.3.2 getDataStructureType	945
6.148.3.3 looseMarshal	945
6.148.3.4 looseUnmarshal	946
6.148.3.5 tightMarshal1	946
6.148.3.6 tightMarshal2	947
6.148.3.7 tightUnmarshal	947
6.149activemq::commands::ConnectionError Class Reference	948
6.149.1 Constructor & Destructor Documentation	949
6.149.1.1 ConnectionError	949
6.149.1.2 ~ConnectionError	949
6.149.2 Member Function Documentation	949
6.149.2.1 cloneDataStructure	949
6.149.2.2 copyDataStructure	949
6.149.2.3 equals	949
6.149.2.4 getConnectionId	950
6.149.2.5 getConnectionId	950
6.149.2.6 getDataStructureType	950
6.149.2.7 getException	950
6.149.2.8 getException	950
6.149.2.9 setConnectionId	950
6.149.2.10setException	950
6.149.2.11toString	950
6.149.2.12visit	950
6.149.3 Field Documentation	951
6.149.3.1 connectionId	951
6.149.3.2 exception	951
6.149.3.3 ID_CONNECTIONERROR	951
6.150activemq::wireformat::openwire::marshal::generated::ConnectionError-	
Marshaller Class Reference	951
6.150.1 Detailed Description	952
6.150.2 Constructor & Destructor Documentation	952
6.150.2.1 ConnectionErrorMarshaller	952

6.150.2.2 ~ConnectionErrorMarshaller	952
6.150.3 Member Function Documentation	952
6.150.3.1 createObject	952
6.150.3.2 getDataStructureType	953
6.150.3.3 looseMarshal	953
6.150.3.4 looseUnmarshal	953
6.150.3.5 tightMarshal1	954
6.150.3.6 tightMarshal2	954
6.150.3.7 tightUnmarshal	955
6.151cms::ConnectionFactory Class Reference	955
6.151.1 Detailed Description	956
6.151.2 Constructor & Destructor Documentation	956
6.151.2.1 ~ConnectionFactory	956
6.151.3 Member Function Documentation	956
6.151.3.1 createCMSConnectionFactory	956
6.151.3.2 createConnection	957
6.151.3.3 createConnection	957
6.151.3.4 createConnection	958
6.152activemq::exceptions::ConnectionFailedException Class Reference	958
6.152.1 Constructor & Destructor Documentation	959
6.152.1.1 ConnectionFailedException	959
6.152.1.2 ConnectionFailedException	959
6.152.1.3 ConnectionFailedException	959
6.152.1.4 ConnectionFailedException	959
6.152.1.5 ~ConnectionFailedException	959
6.152.2 Member Function Documentation	959
6.152.2.1 clone	959
6.153activemq::commands::ConnectionId Class Reference	960
6.153.1 Member Typedef Documentation	961
6.153.1.1 COMPARATOR	961
6.153.2 Constructor & Destructor Documentation	961
6.153.2.1 ConnectionId	961
6.153.2.2 ConnectionId	961
6.153.2.3 ConnectionId	961

6.153.2.4	ConnectionId	961
6.153.2.5	ConnectionId	961
6.153.2.6	~ConnectionId	961
6.153.3	Member Function Documentation	961
6.153.3.1	cloneDataStructure	961
6.153.3.2	compareTo	962
6.153.3.3	copyDataStructure	962
6.153.3.4	equals	962
6.153.3.5	equals	962
6.153.3.6	getDataStructureType	962
6.153.3.7	getValue	962
6.153.3.8	getValue	963
6.153.3.9	operator<	963
6.153.3.10	operator=	963
6.153.3.11	operator==	963
6.153.3.12	setValue	963
6.153.3.13	toString	963
6.153.4	Field Documentation	963
6.153.4.1	ID_CONNECTIONID	963
6.153.4.2	value	963
6.154	activemq::wireformat::openwire::marshal::generated::ConnectionId- Marshaller Class Reference	963
6.154.1	Detailed Description	964
6.154.2	Constructor & Destructor Documentation	964
6.154.2.1	ConnectionIdMarshaller	964
6.154.2.2	~ConnectionIdMarshaller	965
6.154.3	Member Function Documentation	965
6.154.3.1	createObject	965
6.154.3.2	getDataStructureType	965
6.154.3.3	looseMarshal	965
6.154.3.4	looseUnmarshal	966
6.154.3.5	tightMarshal1	966
6.154.3.6	tightMarshal2	967
6.154.3.7	tightUnmarshal	967

6.155activemq::commands::ConnectionInfo Class Reference	968
6.155.1 Constructor & Destructor Documentation	969
6.155.1.1 ConnectionInfo	969
6.155.1.2 ~ConnectionInfo	969
6.155.2 Member Function Documentation	969
6.155.2.1 cloneDataStructure	969
6.155.2.2 copyDataStructure	970
6.155.2.3 createRemoveCommand	970
6.155.2.4 equals	970
6.155.2.5 getBrokerPath	970
6.155.2.6 getBrokerPath	970
6.155.2.7 getClientId	970
6.155.2.8 getClientId	970
6.155.2.9 getConnectionId	971
6.155.2.10getConnectionId	971
6.155.2.11getDataStructureType	971
6.155.2.12getPassword	971
6.155.2.13getPassword	971
6.155.2.14getUserName	971
6.155.2.15getUserName	971
6.155.2.16sBrokerMasterConnector	971
6.155.2.17sClientMaster	971
6.155.2.18sConnectionInfo	971
6.155.2.19sFailoverReconnect	971
6.155.2.20sFaultTolerant	972
6.155.2.21isManageable	972
6.155.2.22setBrokerMasterConnector	972
6.155.2.23setBrokerPath	972
6.155.2.24setClientId	972
6.155.2.25setClientMaster	972
6.155.2.26setConnectionId	972
6.155.2.27setFailoverReconnect	972
6.155.2.28setFaultTolerant	972
6.155.2.29setManageable	972

6.155.2.30	setPassword	972
6.155.2.31	setUserName	972
6.155.2.32	toString	972
6.155.2.33	visit	973
6.155.3	Field Documentation	973
6.155.3.1	brokerMasterConnector	973
6.155.3.2	brokerPath	973
6.155.3.3	clientId	973
6.155.3.4	clientMaster	973
6.155.3.5	connectionId	973
6.155.3.6	failoverReconnect	973
6.155.3.7	faultTolerant	973
6.155.3.8	ID_CONNECTIONINFO	973
6.155.3.9	manageable	973
6.155.3.10	password	973
6.155.3.11	userName	974
6.156	activemq::wireformat::openwire::marshal::generated::ConnectionInfo- Marshaller Class Reference	974
6.156.1	Detailed Description	975
6.156.2	Constructor & Destructor Documentation	975
6.156.2.1	ConnectionInfoMarshaller	975
6.156.2.2	~ConnectionInfoMarshaller	975
6.156.3	Member Function Documentation	975
6.156.3.1	createObject	975
6.156.3.2	getDataStructureType	975
6.156.3.3	looseMarshal	976
6.156.3.4	looseUnmarshal	976
6.156.3.5	tightMarshal1	976
6.156.3.6	tightMarshal2	977
6.156.3.7	tightUnmarshal	977
6.157	cms::ConnectionMetaData Class Reference	978
6.157.1	Detailed Description	979
6.157.2	Constructor & Destructor Documentation	979
6.157.2.1	~ConnectionMetaData	979

6.157.3 Member Function Documentation	979
6.157.3.1 getCMSMajorVersion	979
6.157.3.2 getCMSMinorVersion	979
6.157.3.3 getCMSProviderName	980
6.157.3.4 getCMSVersion	980
6.157.3.5 getCMSXPropertyNames	980
6.157.3.6 getProviderMajorVersion	981
6.157.3.7 getProviderMinorVersion	981
6.157.3.8 getProviderVersion	981
6.158activemq::state::ConnectionState Class Reference	982
6.158.1 Constructor & Destructor Documentation	983
6.158.1.1 ConnectionState	983
6.158.1.2 ~ConnectionState	983
6.158.2 Member Function Documentation	983
6.158.2.1 addSession	983
6.158.2.2 addTempDestination	983
6.158.2.3 addTransactionState	983
6.158.2.4 checkShutdown	983
6.158.2.5 getInfo	983
6.158.2.6 getRecoveringPullConsumers	983
6.158.2.7 getSessionState	983
6.158.2.8 getSessionStates	983
6.158.2.9 getTempDesinations	983
6.158.2.10getTransactionState	983
6.158.2.11getTransactionStates	984
6.158.2.12sConnectionInterruptProcessingComplete	984
6.158.2.13removeSession	984
6.158.2.14removeTempDestination	984
6.158.2.15removeTransactionState	984
6.158.2.16reset	984
6.158.2.17setConnectionInterruptProcessingComplete	984
6.158.2.18shutdown	984
6.158.2.19toString	984
6.159activemq::state::ConnectionStateTracker Class Reference	984

6.159.1 Constructor & Destructor Documentation	986
6.159.1.1 ConnectionStateTracker	986
6.159.1.2 ~ConnectionStateTracker	986
6.159.2 Member Function Documentation	986
6.159.2.1 connectionInterruptProcessingComplete	986
6.159.2.2 getMaxCacheSize	986
6.159.2.3 isRestoreConsumers	986
6.159.2.4 isRestoreProducers	986
6.159.2.5 isRestoreSessions	986
6.159.2.6 isRestoreTransaction	986
6.159.2.7 isTrackMessages	986
6.159.2.8 isTrackTransactionProducers	986
6.159.2.9 isTrackTransactions	986
6.159.2.10 processBeginTransaction	987
6.159.2.11 processCommitTransactionOnePhase	987
6.159.2.12 processCommitTransactionTwoPhase	987
6.159.2.13 processConnectionInfo	987
6.159.2.14 processConsumerInfo	987
6.159.2.15 processDestinationInfo	987
6.159.2.16 processEndTransaction	987
6.159.2.17 processMessage	988
6.159.2.18 processMessageAck	988
6.159.2.19 processPrepareTransaction	988
6.159.2.20 processProducerInfo	988
6.159.2.21 processRemoveConnection	988
6.159.2.22 processRemoveConsumer	988
6.159.2.23 processRemoveDestination	988
6.159.2.24 processRemoveProducer	989
6.159.2.25 processRemoveSession	989
6.159.2.26 processRollbackTransaction	989
6.159.2.27 processSessionInfo	989
6.159.2.28 restore	989
6.159.2.29 setMaxCacheSize	989
6.159.2.30 setRestoreConsumers	989

6.159.2.31	setRestoreProducers	989
6.159.2.32	setRestoreSessions	989
6.159.2.33	setRestoreTransaction	989
6.159.2.34	setTrackMessages	989
6.159.2.35	setTrackTransactionProducers	990
6.159.2.36	setTrackTransactions	990
6.159.2.37	track	990
6.159.2.38	trackBack	990
6.159.2.39	transportInterrupted	990
6.159.3	Friends And Related Function Documentation	990
6.159.3.1	RemoveTransactionAction	990
6.160	decaf::util::logging::ConsoleHandler Class Reference	990
6.160.1	Detailed Description	990
6.160.2	Constructor & Destructor Documentation	991
6.160.2.1	ConsoleHandler	991
6.160.2.2	~ConsoleHandler	991
6.160.3	Member Function Documentation	991
6.160.3.1	close	991
6.160.3.2	publish	991
6.161	activemq::commands::ConsumerControl Class Reference	992
6.161.1	Constructor & Destructor Documentation	993
6.161.1.1	ConsumerControl	993
6.161.1.2	~ConsumerControl	993
6.161.2	Member Function Documentation	993
6.161.2.1	cloneDataStructure	993
6.161.2.2	copyDataStructure	993
6.161.2.3	equals	994
6.161.2.4	getConsumerId	994
6.161.2.5	getConsumerId	994
6.161.2.6	getDataStructureType	994
6.161.2.7	getDestination	994
6.161.2.8	getDestination	994
6.161.2.9	getPrefetch	994
6.161.2.10	isClose	994

6.161.2.11	isFlush	994
6.161.2.12	sStart	995
6.161.2.13	sStop	995
6.161.2.14	setClose	995
6.161.2.15	setConsumerId	995
6.161.2.16	setDestination	995
6.161.2.17	setFlush	995
6.161.2.18	setPrefetch	995
6.161.2.19	setStart	995
6.161.2.20	setStop	995
6.161.2.21	toString	995
6.161.2.22	visit	995
6.161.3	Field Documentation	996
6.161.3.1	close	996
6.161.3.2	consumerId	996
6.161.3.3	destination	996
6.161.3.4	flush	996
6.161.3.5	ID_CONSUMERCONTROL	996
6.161.3.6	prefetch	996
6.161.3.7	start	996
6.161.3.8	stop	996
6.162	activemq::wireformat::openwire::marshal::generated::Consumer- ControlMarshaller Class Reference	996
6.162.1	Detailed Description	997
6.162.2	Constructor & Destructor Documentation	997
6.162.2.1	ConsumerControlMarshaller	997
6.162.2.2	~ConsumerControlMarshaller	997
6.162.3	Member Function Documentation	997
6.162.3.1	createObject	997
6.162.3.2	getDataStructureType	998
6.162.3.3	looseMarshal	998
6.162.3.4	looseUnmarshal	998
6.162.3.5	tightMarshal1	999
6.162.3.6	tightMarshal2	999

6.162.3.7 tightUnmarshal	1000
6.163activemq::commands::ConsumerId Class Reference	1000
6.163.1 Member Typedef Documentation	1002
6.163.1.1 COMPARATOR	1002
6.163.2 Constructor & Destructor Documentation	1002
6.163.2.1 ConsumerId	1002
6.163.2.2 ConsumerId	1002
6.163.2.3 ConsumerId	1002
6.163.2.4 ~ConsumerId	1002
6.163.3 Member Function Documentation	1002
6.163.3.1 cloneDataStructure	1002
6.163.3.2 compareTo	1002
6.163.3.3 copyDataStructure	1002
6.163.3.4 equals	1003
6.163.3.5 equals	1003
6.163.3.6 getConnectionId	1003
6.163.3.7 getConnectionId	1003
6.163.3.8 getDataStructureType	1003
6.163.3.9 getParentId	1003
6.163.3.10getSessionId	1003
6.163.3.11getValue	1003
6.163.3.12operator<	1003
6.163.3.13operator=	1003
6.163.3.14operator==	1004
6.163.3.15setConnectionId	1004
6.163.3.16setSessionId	1004
6.163.3.17setValue	1004
6.163.3.18toString	1004
6.163.4 Field Documentation	1004
6.163.4.1 connectionId	1004
6.163.4.2 ID_CONSUMERID	1004
6.163.4.3 sessionId	1004
6.163.4.4 value	1004

6.164activemq::wireformat::openwire::marshal::generated::ConsumerId-	
Marshaller Class Reference	1005
6.164.1 Detailed Description	1005
6.164.2 Constructor & Destructor Documentation	1006
6.164.2.1 ConsumerIdMarshaller	1006
6.164.2.2 ~ConsumerIdMarshaller	1006
6.164.3 Member Function Documentation	1006
6.164.3.1 createObject	1006
6.164.3.2 getDataStructureType	1006
6.164.3.3 looseMarshal	1006
6.164.3.4 looseUnmarshal	1007
6.164.3.5 tightMarshal1	1007
6.164.3.6 tightMarshal2	1008
6.164.3.7 tightUnmarshal	1008
6.165activemq::commands::ConsumerInfo Class Reference	1009
6.165.1 Constructor & Destructor Documentation	1011
6.165.1.1 ConsumerInfo	1011
6.165.1.2 ~ConsumerInfo	1011
6.165.2 Member Function Documentation	1011
6.165.2.1 cloneDataStructure	1011
6.165.2.2 copyDataStructure	1012
6.165.2.3 createRemoveCommand	1012
6.165.2.4 equals	1012
6.165.2.5 getAdditionalPredicate	1012
6.165.2.6 getAdditionalPredicate	1012
6.165.2.7 getBrokerPath	1012
6.165.2.8 getBrokerPath	1012
6.165.2.9 getConsumerId	1012
6.165.2.10getConsumerId	1013
6.165.2.11getDataStructureType	1013
6.165.2.12getDestination	1013
6.165.2.13getDestination	1013
6.165.2.14getMaximumPendingMessageLimit	1013
6.165.2.15getNetworkConsumerPath	1013

6.165.2.16getNetworkConsumerPath	1013
6.165.2.17getPrefetchSize	1013
6.165.2.18getPriority	1013
6.165.2.19getSelector	1013
6.165.2.20getSelector	1013
6.165.2.21getSubscriptionName	1014
6.165.2.22getSubscriptionName	1014
6.165.2.23sBrowser	1014
6.165.2.24sConsumerInfo	1014
6.165.2.25sDispatchAsync	1014
6.165.2.26sExclusive	1014
6.165.2.27sNetworkSubscription	1014
6.165.2.28sNoLocal	1014
6.165.2.29sNoRangeAcks	1014
6.165.2.30sOptimizedAcknowledge	1014
6.165.2.31isRetroactive	1014
6.165.2.32setAdditionalPredicate	1014
6.165.2.33setBrokerPath	1014
6.165.2.34setBrowser	1015
6.165.2.35setConsumerId	1015
6.165.2.36setDestination	1015
6.165.2.37setDispatchAsync	1015
6.165.2.38setExclusive	1015
6.165.2.39setMaximumPendingMessageLimit	1015
6.165.2.40setNetworkConsumerPath	1015
6.165.2.41setNetworkSubscription	1015
6.165.2.42setNoLocal	1015
6.165.2.43setNoRangeAcks	1015
6.165.2.44setOptimizedAcknowledge	1015
6.165.2.45setPrefetchSize	1015
6.165.2.46setPriority	1015
6.165.2.47setRetroactive	1015
6.165.2.48setSelector	1015
6.165.2.49setSubscriptionName	1016

6.165.2.50toString	1016
6.165.2.51visit	1016
6.165.3 Field Documentation	1016
6.165.3.1 additionalPredicate	1016
6.165.3.2 brokerPath	1016
6.165.3.3 browser	1016
6.165.3.4 consumerId	1016
6.165.3.5 destination	1016
6.165.3.6 dispatchAsync	1016
6.165.3.7 exclusive	1017
6.165.3.8 ID_CONSUMERINFO	1017
6.165.3.9 maximumPendingMessageLimit	1017
6.165.3.10networkConsumerPath	1017
6.165.3.11networkSubscription	1017
6.165.3.12noLocal	1017
6.165.3.13noRangeAcks	1017
6.165.3.14optimizedAcknowledge	1017
6.165.3.15prefetchSize	1017
6.165.3.16priority	1017
6.165.3.17retroactive	1017
6.165.3.18selector	1017
6.165.3.19subscriptionName	1017
6.166activemq::wireformat::openwire::marshal::generated::ConsumerInfo- Marshaller Class Reference	1017
6.166.1 Detailed Description	1018
6.166.2 Constructor & Destructor Documentation	1018
6.166.2.1 ConsumerInfoMarshaller	1018
6.166.2.2 ~ConsumerInfoMarshaller	1019
6.166.3 Member Function Documentation	1019
6.166.3.1 createObject	1019
6.166.3.2 getDataStructureType	1019
6.166.3.3 looseMarshal	1019
6.166.3.4 looseUnmarshal	1020
6.166.3.5 tightMarshal1	1020

6.166.3.6	tightMarshal2	1021
6.166.3.7	tightUnmarshal	1021
6.167	activemq::state::ConsumerState Class Reference	1022
6.167.1	Constructor & Destructor Documentation	1022
6.167.1.1	ConsumerState	1022
6.167.1.2	~ConsumerState	1022
6.167.2	Member Function Documentation	1022
6.167.2.1	getInfo	1022
6.167.2.2	toString	1022
6.168	activemq::commands::ControlCommand Class Reference	1022
6.168.1	Constructor & Destructor Documentation	1023
6.168.1.1	ControlCommand	1023
6.168.1.2	~ControlCommand	1023
6.168.2	Member Function Documentation	1023
6.168.2.1	cloneDataStructure	1023
6.168.2.2	copyDataStructure	1024
6.168.2.3	equals	1024
6.168.2.4	getCommand	1024
6.168.2.5	getCommand	1024
6.168.2.6	getDataStructureType	1024
6.168.2.7	setCommand	1025
6.168.2.8	toString	1025
6.168.2.9	visit	1025
6.168.3	Field Documentation	1025
6.168.3.1	command	1025
6.168.3.2	ID_CONTROLCOMMAND	1025
6.169	activemq::wireformat::openwire::marshal::generated::ControlCommand- Marshaller Class Reference	1025
6.169.1	Detailed Description	1026
6.169.2	Constructor & Destructor Documentation	1026
6.169.2.1	ControlCommandMarshaller	1026
6.169.2.2	~ControlCommandMarshaller	1027
6.169.3	Member Function Documentation	1027
6.169.3.1	createObject	1027

6.169.3.2	getDataStructureType	1027
6.169.3.3	looseMarshal	1027
6.169.3.4	looseUnmarshal	1028
6.169.3.5	tightMarshal1	1028
6.169.3.6	tightMarshal2	1029
6.169.3.7	tightUnmarshal	1029
6.170	decaf::util::concurrent::CopyOnWriteArrayList< E > Class Template - Reference	1030
6.170.1	Constructor & Destructor Documentation	1032
6.170.1.1	CopyOnWriteArrayList	1032
6.170.1.2	CopyOnWriteArrayList	1032
6.170.1.3	CopyOnWriteArrayList	1032
6.170.1.4	CopyOnWriteArrayList	1032
6.170.1.5	~CopyOnWriteArrayList	1032
6.170.2	Member Function Documentation	1032
6.170.2.1	add	1033
6.170.2.2	add	1034
6.170.2.3	addAll	1034
6.170.2.4	addAll	1035
6.170.2.5	addAllAbsent	1036
6.170.2.6	addIfAbsent	1036
6.170.2.7	clear	1036
6.170.2.8	contains	1037
6.170.2.9	containsAll	1037
6.170.2.10	copy	1038
6.170.2.11	equals	1038
6.170.2.12	get	1038
6.170.2.13	indexOf	1039
6.170.2.14	indexOf	1039
6.170.2.15	isEmpty	1040
6.170.2.16	iterator	1040
6.170.2.17	iterator	1040
6.170.2.18	lastIndexOf	1040
6.170.2.19	lastIndexOf	1041

6.170.2.20	istlIterator	1041
6.170.2.21	istlIterator	1042
6.170.2.22	istlIterator	1042
6.170.2.23	istlIterator	1042
6.170.2.24	lock	1042
6.170.2.25	notify	1043
6.170.2.26	notifyAll	1043
6.170.2.27	operator=	1043
6.170.2.28	operator=	1043
6.170.2.29	remove	1043
6.170.2.30	removeAll	1044
6.170.2.31	removeAt	1045
6.170.2.32	retainAll	1045
6.170.2.33	set	1046
6.170.2.34	size	1046
6.170.2.35	toArray	1047
6.170.2.36	toString	1047
6.170.2.37	tryLock	1047
6.170.2.38	unlock	1048
6.170.2.39	wait	1048
6.170.2.40	wait	1048
6.170.2.41	wait	1049
6.170.3	Friends And Related Function Documentation	1049
6.170.3.1	CopyOnWriteArraySet	1049
6.171	decaf::util::concurrent::CopyOnWriteArraySet< E > Class Template - Reference	1050
6.171.1	Detailed Description	1053
6.171.2	Constructor & Destructor Documentation	1053
6.171.2.1	CopyOnWriteArraySet	1053
6.171.2.2	CopyOnWriteArraySet	1053
6.171.2.3	CopyOnWriteArraySet	1054
6.171.2.4	~CopyOnWriteArraySet	1054
6.171.3	Member Function Documentation	1054
6.171.3.1	add	1054

6.171.3.2	addAll	1055
6.171.3.3	clear	1056
6.171.3.4	contains	1056
6.171.3.5	containsAll	1057
6.171.3.6	copy	1057
6.171.3.7	equals	1058
6.171.3.8	isEmpty	1058
6.171.3.9	iterator	1058
6.171.3.10	iterator	1058
6.171.3.11	remove	1059
6.171.3.12	removeAll	1059
6.171.3.13	retainAll	1061
6.171.3.14	size	1061
6.171.3.15	toArray	1062
6.172	decaf::util::concurrent::CountDownLatch Class Reference	1062
6.172.1	Constructor & Destructor Documentation	1063
6.172.1.1	CountDownLatch	1063
6.172.1.2	~CountDownLatch	1063
6.172.2	Member Function Documentation	1063
6.172.2.1	await	1063
6.172.2.2	await	1064
6.172.2.3	await	1064
6.172.2.4	countDown	1065
6.172.2.5	getCount	1065
6.173	decaf::util::zip::CRC32 Class Reference	1065
6.173.1	Detailed Description	1066
6.173.2	Constructor & Destructor Documentation	1066
6.173.2.1	CRC32	1066
6.173.2.2	~CRC32	1066
6.173.3	Member Function Documentation	1066
6.173.3.1	getValue	1066
6.173.3.2	reset	1066
6.173.3.3	update	1067
6.173.3.4	update	1067

6.173.3.5 update	1067
6.173.3.6 update	1068
6.174ct_data_s Struct Reference	1068
6.174.1 Field Documentation	1068
6.174.1.1 code	1068
6.174.1.2 dad	1068
6.174.1.3 dl	1068
6.174.1.4 fc	1068
6.174.1.5 freq	1068
6.174.1.6 len	1068
6.175activemq::commands::DataArrayResponse Class Reference	1069
6.175.1 Constructor & Destructor Documentation	1070
6.175.1.1 DataArrayResponse	1070
6.175.1.2 ~DataArrayResponse	1070
6.175.2 Member Function Documentation	1070
6.175.2.1 cloneDataStructure	1070
6.175.2.2 copyDataStructure	1070
6.175.2.3 equals	1070
6.175.2.4 getData	1071
6.175.2.5 getData	1071
6.175.2.6 getDataStructureType	1071
6.175.2.7 setData	1071
6.175.2.8 toString	1071
6.175.3 Field Documentation	1071
6.175.3.1 data	1071
6.175.3.2 ID_DATAARRAYRESPONSE	1071
6.176activemq::wireformat::openwire::marshal::generated::DataArray- ResponseMarshaller Class Reference	1072
6.176.1 Detailed Description	1072
6.176.2 Constructor & Destructor Documentation	1073
6.176.2.1 DataArrayResponseMarshaller	1073
6.176.2.2 ~DataArrayResponseMarshaller	1073
6.176.3 Member Function Documentation	1073
6.176.3.1 createObject	1073

6.176.3.2	getDataStructureType	1073
6.176.3.3	looseMarshal	1073
6.176.3.4	looseUnmarshal	1074
6.176.3.5	tightMarshal1	1074
6.176.3.6	tightMarshal2	1075
6.176.3.7	tightUnmarshal	1075
6.177	decaf::util::zip::DataFormatException Class Reference	1076
6.177.1	Constructor & Destructor Documentation	1076
6.177.1.1	DataFormatException	1076
6.177.1.2	DataFormatException	1076
6.177.1.3	DataFormatException	1077
6.177.1.4	DataFormatException	1077
6.177.1.5	DataFormatException	1077
6.177.1.6	DataFormatException	1077
6.177.1.7	~DataFormatException	1078
6.177.2	Member Function Documentation	1078
6.177.2.1	clone	1078
6.178	decaf::net::DatagramPacket Class Reference	1078
6.178.1	Detailed Description	1079
6.178.2	Constructor & Destructor Documentation	1080
6.178.2.1	DatagramPacket	1080
6.178.2.2	DatagramPacket	1080
6.178.2.3	DatagramPacket	1080
6.178.2.4	DatagramPacket	1081
6.178.2.5	DatagramPacket	1081
6.178.2.6	DatagramPacket	1082
6.178.2.7	~DatagramPacket	1082
6.178.3	Member Function Documentation	1082
6.178.3.1	getAddress	1082
6.178.3.2	getData	1082
6.178.3.3	getLength	1082
6.178.3.4	getOffset	1083
6.178.3.5	getPort	1083
6.178.3.6	getSize	1083

6.178.3.7	getSocketAddress	1083
6.178.3.8	setAddress	1083
6.178.3.9	setData	1083
6.178.3.10	setData	1084
6.178.3.11	setLength	1084
6.178.3.12	setOffset	1085
6.178.3.13	setPort	1085
6.178.3.14	setSocketAddress	1085
6.179	decaf::io::DataInput Class Reference	1086
6.179.1	Detailed Description	1087
6.179.2	Constructor & Destructor Documentation	1087
6.179.2.1	~DataInput	1087
6.179.3	Member Function Documentation	1087
6.179.3.1	readBoolean	1087
6.179.3.2	readByte	1088
6.179.3.3	readChar	1088
6.179.3.4	readDouble	1088
6.179.3.5	readFloat	1089
6.179.3.6	readFully	1089
6.179.3.7	readFully	1090
6.179.3.8	readInt	1090
6.179.3.9	readLine	1091
6.179.3.10	readLong	1091
6.179.3.11	readShort	1092
6.179.3.12	readString	1092
6.179.3.13	readUnsignedByte	1092
6.179.3.14	readUnsignedShort	1093
6.179.3.15	readUTF	1093
6.179.3.16	skipBytes	1093
6.180	decaf::io::DataInputStream Class Reference	1094
6.180.1	Detailed Description	1095
6.180.2	Constructor & Destructor Documentation	1096
6.180.2.1	DataInputStream	1096
6.180.2.2	~DataInputStream	1096

6.180.3 Member Function Documentation	1096
6.180.3.1 readBoolean	1096
6.180.3.2 readByte	1096
6.180.3.3 readChar	1097
6.180.3.4 readDouble	1097
6.180.3.5 readFloat	1097
6.180.3.6 readFully	1098
6.180.3.7 readFully	1098
6.180.3.8 readInt	1099
6.180.3.9 readLine	1099
6.180.3.10 readLong	1100
6.180.3.11 readShort	1100
6.180.3.12 readString	1101
6.180.3.13 readUnsignedByte	1101
6.180.3.14 readUnsignedShort	1102
6.180.3.15 readUTF	1102
6.180.3.16 skipBytes	1102
6.181 decaf::io::DataOutput Class Reference	1103
6.181.1 Detailed Description	1104
6.181.2 Constructor & Destructor Documentation	1104
6.181.2.1 ~DataOutput	1104
6.181.3 Member Function Documentation	1104
6.181.3.1 writeBoolean	1104
6.181.3.2 writeByte	1105
6.181.3.3 writeBytes	1105
6.181.3.4 writeChar	1105
6.181.3.5 writeChars	1106
6.181.3.6 writeDouble	1106
6.181.3.7 writeFloat	1106
6.181.3.8 writeInt	1107
6.181.3.9 writeLong	1107
6.181.3.10 writeShort	1107
6.181.3.11 writeUnsignedShort	1108
6.181.3.12 writeUTF	1108

6.182decaf::io::DataOutputStream Class Reference	1109
6.182.1 Detailed Description	1110
6.182.2 Constructor & Destructor Documentation	1110
6.182.2.1 DataOutputStream	1110
6.182.2.2 ~DataOutputStream	1110
6.182.3 Member Function Documentation	1110
6.182.3.1 doWriteArrayBounded	1110
6.182.3.2 doWriteByte	1111
6.182.3.3 size	1111
6.182.3.4 writeBoolean	1111
6.182.3.5 writeByte	1111
6.182.3.6 writeBytes	1111
6.182.3.7 writeChar	1111
6.182.3.8 writeChars	1111
6.182.3.9 writeDouble	1111
6.182.3.10writeFloat	1111
6.182.3.11writeInt	1111
6.182.3.12writeLong	1111
6.182.3.13writeShort	1112
6.182.3.14writeUnsignedShort	1112
6.182.3.15writeUTF	1112
6.182.4 Field Documentation	1112
6.182.4.1 buffer	1112
6.182.4.2 written	1112
6.183activemq::commands::DataResponse Class Reference	1112
6.183.1 Constructor & Destructor Documentation	1113
6.183.1.1 DataResponse	1113
6.183.1.2 ~DataResponse	1113
6.183.2 Member Function Documentation	1113
6.183.2.1 cloneDataStructure	1113
6.183.2.2 copyDataStructure	1113
6.183.2.3 equals	1114
6.183.2.4 getData	1114
6.183.2.5 getData	1114

6.183.2.6	getDataStructureType	1114
6.183.2.7	setData	1114
6.183.2.8	toString	1114
6.183.3	Field Documentation	1115
6.183.3.1	data	1115
6.183.3.2	ID_DATARESPONSE	1115
6.184	activemq::wireformat::openwire::marshal::generated::DataResponse- Marshaller Class Reference	1115
6.184.1	Detailed Description	1116
6.184.2	Constructor & Destructor Documentation	1116
6.184.2.1	DataResponseMarshaller	1116
6.184.2.2	~DataResponseMarshaller	1116
6.184.3	Member Function Documentation	1116
6.184.3.1	createObject	1116
6.184.3.2	getDataStructureType	1117
6.184.3.3	looseMarshal	1117
6.184.3.4	looseUnmarshal	1117
6.184.3.5	tightMarshal1	1118
6.184.3.6	tightMarshal2	1118
6.184.3.7	tightUnmarshal	1119
6.185	activemq::wireformat::openwire::marshal::DataStreamMarshaller Class Reference	1119
6.185.1	Detailed Description	1120
6.185.2	Constructor & Destructor Documentation	1120
6.185.2.1	~DataStreamMarshaller	1120
6.185.3	Member Function Documentation	1120
6.185.3.1	createObject	1120
6.185.3.2	getDataStructureType	1122
6.185.3.3	looseMarshal	1123
6.185.3.4	looseUnmarshal	1125
6.185.3.5	tightMarshal1	1127
6.185.3.6	tightMarshal2	1129
6.185.3.7	tightUnmarshal	1131
6.186	activemq::commands::DataStructure Class Reference	1133

6.186.1 Constructor & Destructor Documentation	1133
6.186.1.1 ~DataStructure	1133
6.186.2 Member Function Documentation	1133
6.186.2.1 cloneDataStructure	1133
6.186.2.2 copyDataStructure	1134
6.186.2.3 equals	1135
6.186.2.4 getDataStructureType	1137
6.186.2.5 toString	1138
6.187decaf::util::Date Class Reference	1139
6.187.1 Detailed Description	1140
6.187.2 Constructor & Destructor Documentation	1140
6.187.2.1 Date	1140
6.187.2.2 Date	1140
6.187.2.3 Date	1140
6.187.2.4 ~Date	1140
6.187.3 Member Function Documentation	1140
6.187.3.1 after	1140
6.187.3.2 before	1141
6.187.3.3 compareTo	1141
6.187.3.4 equals	1141
6.187.3.5 getTime	1141
6.187.3.6 operator<	1141
6.187.3.7 operator=	1141
6.187.3.8 operator==	1142
6.187.3.9 setTime	1142
6.187.3.10toString	1142
6.188decaf::internal::DecafRuntime Class Reference	1143
6.188.1 Detailed Description	1143
6.188.2 Constructor & Destructor Documentation	1143
6.188.2.1 DecafRuntime	1143
6.188.2.2 ~DecafRuntime	1143
6.188.3 Member Function Documentation	1143
6.188.3.1 getGlobalLock	1143
6.188.3.2 getGlobalPool	1144

6.189	activemq::threads::DedicatedTaskRunner Class Reference	1144
6.189.1	Constructor & Destructor Documentation	1145
6.189.1.1	DedicatedTaskRunner	1145
6.189.1.2	~DedicatedTaskRunner	1145
6.189.2	Member Function Documentation	1145
6.189.2.1	run	1145
6.189.2.2	shutdown	1145
6.189.2.3	shutdown	1145
6.189.2.4	wakeup	1145
6.190	activemq::core::policies::DefaultPrefetchPolicy Class Reference	1146
6.190.1	Constructor & Destructor Documentation	1147
6.190.1.1	DefaultPrefetchPolicy	1147
6.190.1.2	~DefaultPrefetchPolicy	1147
6.190.2	Member Function Documentation	1147
6.190.2.1	clone	1147
6.190.2.2	getDurableTopicPrefetch	1147
6.190.2.3	getMaxPrefetchLimit	1147
6.190.2.4	getQueueBrowserPrefetch	1148
6.190.2.5	getQueuePrefetch	1148
6.190.2.6	getTopicPrefetch	1148
6.190.2.7	setDurableTopicPrefetch	1148
6.190.2.8	setQueueBrowserPrefetch	1149
6.190.2.9	setQueuePrefetch	1149
6.190.2.10	setTopicPrefetch	1149
6.190.3	Field Documentation	1149
6.190.3.1	DEFAULT_DURABLE_TOPIC_PREFETCH	1149
6.190.3.2	DEFAULT_QUEUE_BROWSER_PREFETCH	1149
6.190.3.3	DEFAULT_QUEUE_PREFETCH	1149
6.190.3.4	DEFAULT_TOPIC_PREFETCH	1150
6.190.3.5	MAX_PREFETCH_SIZE	1150
6.191	activemq::core::policies::DefaultRedeliveryPolicy Class Reference	1150
6.191.1	Constructor & Destructor Documentation	1151
6.191.1.1	DefaultRedeliveryPolicy	1151
6.191.1.2	~DefaultRedeliveryPolicy	1151

6.191.2 Member Function Documentation	1151
6.191.2.1 clone	1151
6.191.2.2 getBackOffMultiplier	1151
6.191.2.3 getCollisionAvoidancePercent	1151
6.191.2.4 getInitialRedeliveryDelay	1152
6.191.2.5 getMaximumRedeliveries	1152
6.191.2.6 getNextRedeliveryDelay	1152
6.191.2.7 getRedeliveryDelay	1152
6.191.2.8 isUseCollisionAvoidance	1153
6.191.2.9 isUseExponentialBackOff	1153
6.191.2.10setBackOffMultiplier	1153
6.191.2.11setCollisionAvoidancePercent	1153
6.191.2.12setInitialRedeliveryDelay	1154
6.191.2.13setMaximumRedeliveries	1154
6.191.2.14setRedeliveryDelay	1154
6.191.2.15setUseCollisionAvoidance	1154
6.191.2.16setUseExponentialBackOff	1155
6.192decaf::internal::net::DefaultServerSocketFactory Class Reference	1155
6.192.1 Detailed Description	1156
6.192.2 Constructor & Destructor Documentation	1156
6.192.2.1 DefaultServerSocketFactory	1157
6.192.2.2 ~DefaultServerSocketFactory	1157
6.192.3 Member Function Documentation	1157
6.192.3.1 createServerSocket	1157
6.192.3.2 createServerSocket	1157
6.192.3.3 createServerSocket	1158
6.192.3.4 createServerSocket	1158
6.193decaf::internal::net::DefaultSocketFactory Class Reference	1159
6.193.1 Detailed Description	1161
6.193.2 Constructor & Destructor Documentation	1161
6.193.2.1 DefaultSocketFactory	1161
6.193.2.2 ~DefaultSocketFactory	1161
6.193.3 Member Function Documentation	1161
6.193.3.1 createSocket	1161

6.193.3.2 createSocket	1161
6.193.3.3 createSocket	1162
6.193.3.4 createSocket	1163
6.193.3.5 createSocket	1163
6.194decaf::internal::net::ssl::DefaultSSLContext Class Reference	1164
6.194.1 Detailed Description	1164
6.194.2 Constructor & Destructor Documentation	1164
6.194.2.1 DefaultSSLContext	1164
6.194.2.2 ~DefaultSSLContext	1164
6.194.3 Member Function Documentation	1165
6.194.3.1 getContext	1165
6.195decaf::internal::net::ssl::DefaultSSLServerSocketFactory Class Reference	1165
6.195.1 Detailed Description	1167
6.195.2 Constructor & Destructor Documentation	1167
6.195.2.1 DefaultSSLServerSocketFactory	1167
6.195.2.2 ~DefaultSSLServerSocketFactory	1167
6.195.3 Member Function Documentation	1167
6.195.3.1 createServerSocket	1167
6.195.3.2 createServerSocket	1168
6.195.3.3 createServerSocket	1168
6.195.3.4 createServerSocket	1169
6.195.3.5 getDefaultCipherSuites	1169
6.195.3.6 getSupportedCipherSuites	1170
6.196decaf::internal::net::ssl::DefaultSSLSocketFactory Class Reference . . .	1170
6.196.1 Detailed Description	1173
6.196.2 Constructor & Destructor Documentation	1173
6.196.2.1 DefaultSSLSocketFactory	1173
6.196.2.2 ~DefaultSSLSocketFactory	1173
6.196.3 Member Function Documentation	1173
6.196.3.1 createSocket	1173
6.196.3.2 createSocket	1174
6.196.3.3 createSocket	1174
6.196.3.4 createSocket	1175
6.196.3.5 createSocket	1176

6.196.3.6	createSocket	1176
6.196.3.7	getDefaultCipherSuites	1177
6.196.3.8	getSupportedCipherSuites	1177
6.197	activemq::transport::DefaultTransportListener Class Reference	1178
6.197.1	Detailed Description	1178
6.197.2	Constructor & Destructor Documentation	1178
6.197.2.1	~DefaultTransportListener	1179
6.197.3	Member Function Documentation	1179
6.197.3.1	onCommand	1179
6.197.3.2	onException	1179
6.197.3.3	transportInterrupted	1179
6.197.3.4	transportResumed	1179
6.198	decaf::util::zip::Deflater Class Reference	1180
6.198.1	Detailed Description	1181
6.198.2	Constructor & Destructor Documentation	1182
6.198.2.1	Deflater	1182
6.198.2.2	Deflater	1182
6.198.2.3	~Deflater	1182
6.198.3	Member Function Documentation	1182
6.198.3.1	deflate	1182
6.198.3.2	deflate	1183
6.198.3.3	deflate	1183
6.198.3.4	end	1184
6.198.3.5	finish	1184
6.198.3.6	finished	1184
6.198.3.7	getAdler	1184
6.198.3.8	getBytesRead	1184
6.198.3.9	getBytesWritten	1184
6.198.3.10	needsInput	1185
6.198.3.11	reset	1185
6.198.3.12	setDictionary	1185
6.198.3.13	setDictionary	1186
6.198.3.14	setDictionary	1186
6.198.3.15	setInput	1186

6.198.3.16	setInput	1187
6.198.3.17	setInput	1187
6.198.3.18	setLevel	1188
6.198.3.19	setStrategy	1188
6.198.4	Field Documentation	1188
6.198.4.1	BEST_COMPRESSION	1188
6.198.4.2	BEST_SPEED	1188
6.198.4.3	DEFAULT_COMPRESSION	1189
6.198.4.4	DEFAULT_STRATEGY	1189
6.198.4.5	DEFLATED	1189
6.198.4.6	FILTERED	1189
6.198.4.7	HUFFMAN_ONLY	1189
6.198.4.8	NO_COMPRESSION	1189
6.199	decaf::util::zip::DeflaterOutputStream Class Reference	1189
6.199.1	Detailed Description	1191
6.199.2	Constructor & Destructor Documentation	1191
6.199.2.1	DeflaterOutputStream	1191
6.199.2.2	DeflaterOutputStream	1191
6.199.2.3	DeflaterOutputStream	1192
6.199.2.4	~DeflaterOutputStream	1192
6.199.3	Member Function Documentation	1192
6.199.3.1	close	1192
6.199.3.2	deflate	1193
6.199.3.3	doWriteArrayBounded	1193
6.199.3.4	doWriteByte	1193
6.199.3.5	finish	1193
6.199.4	Field Documentation	1193
6.199.4.1	buf	1193
6.199.4.2	DEFAULT_BUFFER_SIZE	1193
6.199.4.3	deflater	1194
6.199.4.4	isDone	1194
6.199.4.5	ownDeflater	1194
6.200	decaf::util::concurrent::Delayed Class Reference	1194
6.200.1	Detailed Description	1194

6.200.2 Constructor & Destructor Documentation	1194
6.200.2.1 ~Delayed	1194
6.200.3 Member Function Documentation	1194
6.200.3.1 getDelay	1195
6.201 cms::DeliveryMode Class Reference	1195
6.201.1 Detailed Description	1195
6.201.2 Member Enumeration Documentation	1196
6.201.2.1 DELIVERY_MODE	1196
6.201.3 Constructor & Destructor Documentation	1196
6.201.3.1 ~DeliveryMode	1196
6.202 decaf::util::Deque< E > Class Template Reference	1196
6.202.1 Detailed Description	1198
6.202.2 Constructor & Destructor Documentation	1198
6.202.2.1 ~Deque	1198
6.202.3 Member Function Documentation	1198
6.202.3.1 addFirst	1198
6.202.3.2 addLast	1199
6.202.3.3 descendingIterator	1199
6.202.3.4 descendingIterator	1200
6.202.3.5 getFirst	1200
6.202.3.6 getFirst	1201
6.202.3.7 getLast	1201
6.202.3.8 getLast	1202
6.202.3.9 offerFirst	1202
6.202.3.10 offerLast	1203
6.202.3.11 peekFirst	1204
6.202.3.12 peekLast	1204
6.202.3.13 pollFirst	1205
6.202.3.14 pollLast	1205
6.202.3.15 pop	1206
6.202.3.16 push	1206
6.202.3.17 removeFirst	1207
6.202.3.18 removeFirstOccurrence	1208
6.202.3.19 removeLast	1208

6.202.3.20removeLastOccurrence	1209
6.203cms::Destination Class Reference	1210
6.203.1 Detailed Description	1210
6.203.2 Member Enumeration Documentation	1211
6.203.2.1 DestinationType	1211
6.203.3 Constructor & Destructor Documentation	1211
6.203.3.1 ~Destination	1211
6.203.4 Member Function Documentation	1211
6.203.4.1 clone	1211
6.203.4.2 copy	1211
6.203.4.3 equals	1212
6.203.4.4 getCMSProperties	1212
6.203.4.5 getDestinationType	1212
6.204activemq::commands::ActiveMQDestination::DestinationFilter Struct Reference	1213
6.204.1 Field Documentation	1213
6.204.1.1 ANY_CHILD	1213
6.204.1.2 ANY_DESCENDENT	1213
6.205activemq::commands::DestinationInfo Class Reference	1213
6.205.1 Constructor & Destructor Documentation	1215
6.205.1.1 DestinationInfo	1215
6.205.1.2 ~DestinationInfo	1215
6.205.2 Member Function Documentation	1215
6.205.2.1 cloneDataStructure	1215
6.205.2.2 copyDataStructure	1215
6.205.2.3 equals	1215
6.205.2.4 getBrokerPath	1216
6.205.2.5 getBrokerPath	1216
6.205.2.6 getConnectionId	1216
6.205.2.7 getConnectionId	1216
6.205.2.8 getDataStructureType	1216
6.205.2.9 getDestination	1216
6.205.2.10getDestination	1216
6.205.2.11getOperationType	1216

6.205.2.12	getTimeout	1216
6.205.2.13	setBrokerPath	1216
6.205.2.14	setConnectionId	1216
6.205.2.15	setDestination	1217
6.205.2.16	setOperationType	1217
6.205.2.17	setTimeout	1217
6.205.2.18	toString	1217
6.205.2.19	visit	1217
6.205.3	Field Documentation	1217
6.205.3.1	brokerPath	1217
6.205.3.2	connectionId	1217
6.205.3.3	destination	1217
6.205.3.4	ID_DESTINATIONINFO	1217
6.205.3.5	operationType	1218
6.205.3.6	timeout	1218
6.206	activemq::wireformat::openwire::marshal::generated::DestinationInfo- Marshaller Class Reference	1218
6.206.1	Detailed Description	1219
6.206.2	Constructor & Destructor Documentation	1219
6.206.2.1	DestinationInfoMarshaller	1219
6.206.2.2	~DestinationInfoMarshaller	1219
6.206.3	Member Function Documentation	1219
6.206.3.1	createObject	1219
6.206.3.2	getDataStructureType	1219
6.206.3.3	looseMarshal	1220
6.206.3.4	looseUnmarshal	1220
6.206.3.5	tightMarshal1	1220
6.206.3.6	tightMarshal2	1221
6.206.3.7	tightUnmarshal	1221
6.207	activemq::cmsutil::DestinationResolver Class Reference	1222
6.207.1	Detailed Description	1222
6.207.2	Constructor & Destructor Documentation	1222
6.207.2.1	~DestinationResolver	1222
6.207.3	Member Function Documentation	1222

6.207.3.1	destroy	1223
6.207.3.2	init	1223
6.207.3.3	resolveDestinationName	1223
6.208	decaf::util::concurrent::ThreadPoolExecutor::CallerRunsPolicy::Discard- OldestPolicy Class Reference	1224
6.208.1	Detailed Description	1224
6.208.2	Constructor & Destructor Documentation	1224
6.208.2.1	DiscardOldestPolicy	1224
6.208.2.2	~DiscardOldestPolicy	1224
6.208.3	Member Function Documentation	1225
6.208.3.1	rejectedExecution	1225
6.209	decaf::util::concurrent::ThreadPoolExecutor::CallerRunsPolicy::Discard- Policy Class Reference	1225
6.209.1	Detailed Description	1226
6.209.2	Constructor & Destructor Documentation	1226
6.209.2.1	DiscardPolicy	1226
6.209.2.2	~DiscardPolicy	1226
6.209.3	Member Function Documentation	1226
6.209.3.1	rejectedExecution	1226
6.210	activemq::commands::DiscoveryEvent Class Reference	1226
6.210.1	Constructor & Destructor Documentation	1227
6.210.1.1	DiscoveryEvent	1227
6.210.1.2	~DiscoveryEvent	1227
6.210.2	Member Function Documentation	1227
6.210.2.1	cloneDataStructure	1228
6.210.2.2	copyDataStructure	1228
6.210.2.3	equals	1228
6.210.2.4	getBrokerName	1228
6.210.2.5	getBrokerName	1228
6.210.2.6	getDataStructureType	1228
6.210.2.7	getServiceName	1229
6.210.2.8	getServiceName	1229
6.210.2.9	setBrokerName	1229
6.210.2.10	setServiceName	1229

6.210.2.1 toString	1229
6.210.3 Field Documentation	1229
6.210.3.1 brokerName	1229
6.210.3.2 ID_DISCOVERYEVENT	1229
6.210.3.3 serviceName	1229
6.211activemq::wireformat::openwire::marshal::generated::DiscoveryEvent- Marshaller Class Reference	1230
6.211.1 Detailed Description	1230
6.211.2 Constructor & Destructor Documentation	1231
6.211.2.1 DiscoveryEventMarshaller	1231
6.211.2.2 ~DiscoveryEventMarshaller	1231
6.211.3 Member Function Documentation	1231
6.211.3.1 createObject	1231
6.211.3.2 getDataStructureType	1231
6.211.3.3 looseMarshal	1231
6.211.3.4 looseUnmarshal	1232
6.211.3.5 tightMarshal1	1232
6.211.3.6 tightMarshal2	1233
6.211.3.7 tightUnmarshal	1233
6.212activemq::core::DispatchData Class Reference	1234
6.212.1 Detailed Description	1234
6.212.2 Constructor & Destructor Documentation	1234
6.212.2.1 DispatchData	1234
6.212.2.2 DispatchData	1234
6.212.3 Member Function Documentation	1234
6.212.3.1 getConsumerId	1234
6.212.3.2 getMessage	1234
6.213activemq::core::Dispatcher Class Reference	1234
6.213.1 Detailed Description	1235
6.213.2 Constructor & Destructor Documentation	1235
6.213.2.1 ~Dispatcher	1235
6.213.3 Member Function Documentation	1235
6.213.3.1 dispatch	1235
6.214decaf::lang::Double Class Reference	1235

6.214.1 Constructor & Destructor Documentation	1238
6.214.1.1 Double	1238
6.214.1.2 Double	1238
6.214.1.3 ~Double	1238
6.214.2 Member Function Documentation	1238
6.214.2.1 byteValue	1238
6.214.2.2 compare	1238
6.214.2.3 compareTo	1239
6.214.2.4 compareTo	1239
6.214.2.5 doubleToLongBits	1240
6.214.2.6 doubleToRawLongBits	1240
6.214.2.7 doubleValue	1241
6.214.2.8 equals	1241
6.214.2.9 equals	1241
6.214.2.10 floatValue	1242
6.214.2.11 intValue	1242
6.214.2.12 isInfinite	1242
6.214.2.13 isInfinite	1242
6.214.2.14 isNaN	1242
6.214.2.15 isNaN	1242
6.214.2.16 longBitsToDouble	1243
6.214.2.17 longValue	1243
6.214.2.18 operator<	1243
6.214.2.19 operator<	1244
6.214.2.20 operator==	1244
6.214.2.21 operator==	1244
6.214.2.22 parseDouble	1245
6.214.2.23 shortValue	1245
6.214.2.24 toHexString	1245
6.214.2.25 toString	1246
6.214.2.26 toString	1246
6.214.2.27 valueOf	1247
6.214.2.28 valueOf	1247
6.214.3 Field Documentation	1248

6.214.3.1 MAX_VALUE	1248
6.214.3.2 MIN_VALUE	1248
6.214.3.3 NaN	1248
6.214.3.4 NEGATIVE_INFINITY	1248
6.214.3.5 POSITIVE_INFINITY	1248
6.214.3.6 SIZE	1248
6.215decaf::internal::nio::DoubleArrayBuffer Class Reference	1248
6.215.1 Constructor & Destructor Documentation	1252
6.215.1.1 DoubleArrayBuffer	1252
6.215.1.2 DoubleArrayBuffer	1252
6.215.1.3 DoubleArrayBuffer	1253
6.215.1.4 DoubleArrayBuffer	1253
6.215.1.5 ~DoubleArrayBuffer	1254
6.215.2 Member Function Documentation	1254
6.215.2.1 array	1254
6.215.2.2 arrayOffset	1254
6.215.2.3 asReadOnlyBuffer	1255
6.215.2.4 compact	1255
6.215.2.5 duplicate	1256
6.215.2.6 get	1256
6.215.2.7 get	1256
6.215.2.8 hasArray	1257
6.215.2.9 isReadOnly	1257
6.215.2.10put	1257
6.215.2.11put	1258
6.215.2.12setReadOnly	1258
6.215.2.13slice	1259
6.216decaf::nio::DoubleBuffer Class Reference	1259
6.216.1 Detailed Description	1261
6.216.2 Constructor & Destructor Documentation	1261
6.216.2.1 DoubleBuffer	1261
6.216.2.2 ~DoubleBuffer	1261
6.216.3 Member Function Documentation	1262
6.216.3.1 allocate	1262

6.216.3.2 array	1262
6.216.3.3 arrayOffset	1263
6.216.3.4 asReadOnlyBuffer	1263
6.216.3.5 compact	1263
6.216.3.6 compareTo	1264
6.216.3.7 duplicate	1264
6.216.3.8 equals	1264
6.216.3.9 get	1264
6.216.3.10get	1265
6.216.3.11get	1265
6.216.3.12get	1266
6.216.3.13hasArray	1266
6.216.3.14operator<	1267
6.216.3.15operator==	1267
6.216.3.16put	1267
6.216.3.17put	1267
6.216.3.18put	1268
6.216.3.19put	1269
6.216.3.20put	1269
6.216.3.21slice	1270
6.216.3.22toString	1270
6.216.3.23wrap	1270
6.216.3.24wrap	1271
6.217decaf::lang::DYNAMIC_CAST_TOKEN Struct Reference	1271
6.218activemq::cmsutil::DynamicDestinationResolver Class Reference	1272
6.218.1 Detailed Description	1272
6.218.2 Constructor & Destructor Documentation	1272
6.218.2.1 DynamicDestinationResolver	1272
6.218.2.2 ~DynamicDestinationResolver	1272
6.218.3 Member Function Documentation	1272
6.218.3.1 destroy	1273
6.218.3.2 init	1273
6.218.3.3 resolveDestinationName	1273
6.219decaf::util::Map< K, V, COMPARATOR >::Entry Class Reference	1274

6.219.1 Constructor & Destructor Documentation	1274
6.219.1.1 Entry	1274
6.219.1.2 ~Entry	1274
6.219.2 Member Function Documentation	1274
6.219.2.1 getKey	1274
6.219.2.2 getValue	1274
6.219.2.3 setValue	1274
6.220decaf::io::EOFException Class Reference	1275
6.220.1 Constructor & Destructor Documentation	1275
6.220.1.1 EOFException	1275
6.220.1.2 EOFException	1275
6.220.1.3 EOFException	1276
6.220.1.4 EOFException	1276
6.220.1.5 EOFException	1276
6.220.1.6 EOFException	1276
6.220.1.7 ~EOFException	1277
6.220.2 Member Function Documentation	1277
6.220.2.1 clone	1277
6.221decaf::util::logging::ErrorManager Class Reference	1277
6.221.1 Detailed Description	1278
6.221.2 Constructor & Destructor Documentation	1278
6.221.2.1 ErrorManager	1278
6.221.2.2 ~ErrorManager	1278
6.221.3 Member Function Documentation	1278
6.221.3.1 error	1278
6.221.4 Field Documentation	1278
6.221.4.1 CLOSE_FAILURE	1279
6.221.4.2 FLUSH_FAILURE	1279
6.221.4.3 FORMAT_FAILURE	1279
6.221.4.4 GENERIC_FAILURE	1279
6.221.4.5 OPEN_FAILURE	1279
6.221.4.6 WRITE_FAILURE	1279
6.222decaf::lang::Exception Class Reference	1279
6.222.1 Constructor & Destructor Documentation	1281

6.222.1.1 Exception	1281
6.222.1.2 Exception	1281
6.222.1.3 Exception	1281
6.222.1.4 Exception	1281
6.222.1.5 Exception	1282
6.222.1.6 ~Exception	1282
6.222.2 Member Function Documentation	1282
6.222.2.1 buildMessage	1282
6.222.2.2 clone	1282
6.222.2.3 getCause	1283
6.222.2.4 getMessage	1284
6.222.2.5 getStackTrace	1284
6.222.2.6 getStackTraceString	1284
6.222.2.7 initCause	1284
6.222.2.8 operator=	1285
6.222.2.9 printStackTrace	1285
6.222.2.10 printStackTrace	1285
6.222.2.11 setMark	1285
6.222.2.12 setMessage	1286
6.222.2.13 setStackTrace	1286
6.222.2.14 what	1286
6.222.3 Field Documentation	1286
6.222.3.1 cause	1286
6.222.3.2 message	1286
6.222.3.3 stackTrace	1286
6.223 cms::ExceptionListener Class Reference	1286
6.223.1 Detailed Description	1287
6.223.2 Constructor & Destructor Documentation	1287
6.223.2.1 ~ExceptionListener	1287
6.223.3 Member Function Documentation	1287
6.223.3.1 onException	1287
6.224 activemq::commands::ExceptionResponse Class Reference	1287
6.224.1 Constructor & Destructor Documentation	1288
6.224.1.1 ExceptionResponse	1288

6.224.1.2 ~ExceptionResponse	1288
6.224.2 Member Function Documentation	1288
6.224.2.1 cloneDataStructure	1289
6.224.2.2 copyDataStructure	1289
6.224.2.3 equals	1289
6.224.2.4 getDataStructureType	1289
6.224.2.5 getException	1290
6.224.2.6 getException	1290
6.224.2.7 setException	1290
6.224.2.8 toString	1290
6.224.3 Field Documentation	1290
6.224.3.1 exception	1290
6.224.3.2 ID_EXCEPTIONRESPONSE	1290
6.225activemq::wireformat::openwire::marshal::generated::ExceptionResponse- Marshaller Class Reference	1290
6.225.1 Detailed Description	1291
6.225.2 Constructor & Destructor Documentation	1291
6.225.2.1 ExceptionResponseMarshaller	1291
6.225.2.2 ~ExceptionResponseMarshaller	1291
6.225.3 Member Function Documentation	1291
6.225.3.1 createObject	1292
6.225.3.2 getDataStructureType	1292
6.225.3.3 looseMarshal	1292
6.225.3.4 looseUnmarshal	1293
6.225.3.5 tightMarshal1	1293
6.225.3.6 tightMarshal2	1293
6.225.3.7 tightUnmarshal	1294
6.226decaf::util::concurrent::ExecutionException Class Reference	1294
6.226.1 Constructor & Destructor Documentation	1295
6.226.1.1 ExecutionException	1295
6.226.1.2 ExecutionException	1295
6.226.1.3 ExecutionException	1295
6.226.1.4 ExecutionException	1296
6.226.1.5 ExecutionException	1296

6.226.1.6	ExecutionException	1296
6.226.1.7	~ExecutionException	1297
6.226.2	Member Function Documentation	1297
6.226.2.1	clone	1297
6.227	decaf::util::concurrent::Executor Class Reference	1297
6.227.1	Detailed Description	1297
6.227.2	Constructor & Destructor Documentation	1299
6.227.2.1	~Executor	1299
6.227.3	Member Function Documentation	1299
6.227.3.1	execute	1299
6.228	decaf::util::concurrent::Executors Class Reference	1299
6.228.1	Detailed Description	1300
6.228.2	Constructor & Destructor Documentation	1300
6.228.2.1	~Executors	1300
6.228.3	Member Function Documentation	1300
6.228.3.1	getDefaultThreadFactory	1300
6.228.3.2	newFixedThreadPool	1301
6.228.3.3	newFixedThreadPool	1301
6.228.4	Friends And Related Function Documentation	1302
6.228.4.1	decaf::lang::Thread	1302
6.229	decaf::util::concurrent::ExecutorService Class Reference	1302
6.229.1	Detailed Description	1303
6.229.2	Constructor & Destructor Documentation	1303
6.229.2.1	~ExecutorService	1303
6.229.3	Member Function Documentation	1303
6.229.3.1	awaitTermination	1303
6.229.3.2	isShutdown	1304
6.229.3.3	isTerminated	1304
6.229.3.4	shutdown	1304
6.229.3.5	shutdownNow	1305
6.230	activemq::transport::failover::FailoverTransport Class Reference	1305
6.230.1	Constructor & Destructor Documentation	1308
6.230.1.1	FailoverTransport	1308
6.230.1.2	~FailoverTransport	1308

6.230.2 Member Function Documentation	1308
6.230.2.1 add	1308
6.230.2.2 addURI	1308
6.230.2.3 close	1308
6.230.2.4 getBackOffMultiplier	1309
6.230.2.5 getBackupPoolSize	1309
6.230.2.6 getInitialReconnectDelay	1309
6.230.2.7 getMaxCacheSize	1309
6.230.2.8 getMaxReconnectAttempts	1309
6.230.2.9 getMaxReconnectDelay	1309
6.230.2.10getReconnectDelay	1309
6.230.2.11getRemoteAddress	1309
6.230.2.12getStartupMaxReconnectAttempts	1309
6.230.2.13getTimeout	1309
6.230.2.14getTransportListener	1309
6.230.2.15getWireFormat	1310
6.230.2.16handleConnectionControl	1310
6.230.2.17handleTransportFailure	1310
6.230.2.18sBackup	1310
6.230.2.19sClosed	1311
6.230.2.20sConnected	1311
6.230.2.21sFaultTolerant	1311
6.230.2.22sInitialized	1311
6.230.2.23sPending	1311
6.230.2.24sRandomize	1312
6.230.2.25sReconnectSupported	1312
6.230.2.26sTrackMessages	1312
6.230.2.27sTrackTransactionProducers	1312
6.230.2.28sUpdateURIsSupported	1312
6.230.2.29sUseExponentialBackOff	1312
6.230.2.30terate	1312
6.230.2.31narrow	1312
6.230.2.32neway	1313
6.230.2.33reconnect	1313

6.230.2.34	reconnect	1313
6.230.2.35	removeURI	1314
6.230.2.36	request	1314
6.230.2.37	request	1315
6.230.2.38	restoreTransport	1315
6.230.2.39	setBackOffMultiplier	1315
6.230.2.40	setBackup	1315
6.230.2.41	setBackupPoolSize	1315
6.230.2.42	setConnectionInterruptProcessingComplete	1316
6.230.2.43	setInitialized	1316
6.230.2.44	setInitialReconnectDelay	1316
6.230.2.45	setMaxCacheSize	1316
6.230.2.46	setMaxReconnectAttempts	1316
6.230.2.47	setMaxReconnectDelay	1316
6.230.2.48	setRandomize	1316
6.230.2.49	setReconnectDelay	1316
6.230.2.50	setReconnectSupported	1316
6.230.2.51	setStartupMaxReconnectAttempts	1316
6.230.2.52	setTimeout	1316
6.230.2.53	setTrackMessages	1316
6.230.2.54	setTrackTransactionProducers	1316
6.230.2.55	setTransportListener	1316
6.230.2.56	setUpdateURIsSupported	1317
6.230.2.57	setUseExponentialBackOff	1317
6.230.2.58	setWireFormat	1317
6.230.2.59	start	1317
6.230.2.60	stop	1317
6.230.2.61	updateURIs	1317
6.230.3	Friends And Related Function Documentation	1318
6.230.3.1	FailoverTransportListener	1318
6.231	activemq::transport::failover::FailoverTransportFactory Class Reference	1318
6.231.1	Detailed Description	1319
6.231.2	Constructor & Destructor Documentation	1319
6.231.2.1	~FailoverTransportFactory	1319

6.231.3 Member Function Documentation	1319
6.231.3.1 create	1319
6.231.3.2 createComposite	1319
6.231.3.3 doCreateComposite	1320
6.232activemq::transport::failover::FailoverTransportListener Class Reference	1320
6.232.1 Detailed Description	1321
6.232.2 Constructor & Destructor Documentation	1321
6.232.2.1 FailoverTransportListener	1321
6.232.2.2 ~FailoverTransportListener	1321
6.232.3 Member Function Documentation	1321
6.232.3.1 onCommand	1321
6.232.3.2 onException	1322
6.232.3.3 transportInterrupted	1322
6.232.3.4 transportResumed	1322
6.233activemq::core::FifoMessageDispatchChannel Class Reference	1322
6.233.1 Constructor & Destructor Documentation	1324
6.233.1.1 FifoMessageDispatchChannel	1324
6.233.1.2 ~FifoMessageDispatchChannel	1324
6.233.2 Member Function Documentation	1324
6.233.2.1 clear	1324
6.233.2.2 close	1324
6.233.2.3 dequeue	1324
6.233.2.4 dequeueNoWait	1325
6.233.2.5 enqueue	1325
6.233.2.6 enqueueFirst	1325
6.233.2.7 isClosed	1325
6.233.2.8 isEmpty	1326
6.233.2.9 isRunning	1326
6.233.2.10lock	1326
6.233.2.11notify	1326
6.233.2.12notifyAll	1327
6.233.2.13peek	1327
6.233.2.14removeAll	1327
6.233.2.15size	1327

6.233.2.16	start	1328
6.233.2.17	stop	1328
6.233.2.18	tryLock	1328
6.233.2.19	unlock	1328
6.233.2.20	wait	1329
6.233.2.21	wait	1329
6.233.2.22	wait	1330
6.234	decaf::io::FileDescriptor Class Reference	1330
6.234.1	Detailed Description	1331
6.234.2	Constructor & Destructor Documentation	1331
6.234.2.1	FileDescriptor	1331
6.234.2.2	FileDescriptor	1331
6.234.2.3	~FileDescriptor	1332
6.234.3	Member Function Documentation	1332
6.234.3.1	sync	1332
6.234.3.2	valid	1332
6.234.4	Field Documentation	1332
6.234.4.1	descriptor	1332
6.234.4.2	err	1332
6.234.4.3	in	1332
6.234.4.4	out	1332
6.234.4.5	readonly	1333
6.235	decaf::util::logging::Filter Class Reference	1333
6.235.1	Detailed Description	1333
6.235.2	Constructor & Destructor Documentation	1333
6.235.2.1	~Filter	1333
6.235.3	Member Function Documentation	1333
6.235.3.1	isLoggable	1333
6.236	decaf::io::FilterInputStream Class Reference	1334
6.236.1	Detailed Description	1336
6.236.2	Constructor & Destructor Documentation	1336
6.236.2.1	FilterInputStream	1336
6.236.2.2	~FilterInputStream	1336
6.236.3	Member Function Documentation	1337

6.236.3.1 available	1337
6.236.3.2 close	1337
6.236.3.3 doReadArray	1337
6.236.3.4 doReadArrayBounded	1338
6.236.3.5 doReadByte	1338
6.236.3.6 isClosed	1338
6.236.3.7 mark	1338
6.236.3.8 markSupported	1339
6.236.3.9 reset	1339
6.236.3.10 skip	1340
6.236.4 Field Documentation	1340
6.236.4.1 closed	1340
6.236.4.2 inputStream	1340
6.236.4.3 own	1340
6.237decaf::io::FilterOutputStream Class Reference	1341
6.237.1 Detailed Description	1342
6.237.2 Constructor & Destructor Documentation	1342
6.237.2.1 FilterOutputStream	1342
6.237.2.2 ~FilterOutputStream	1342
6.237.3 Member Function Documentation	1342
6.237.3.1 close	1342
6.237.3.2 doWriteArray	1343
6.237.3.3 doWriteArrayBounded	1343
6.237.3.4 doWriteByte	1343
6.237.3.5 flush	1343
6.237.3.6 isClosed	1344
6.237.3.7 toString	1344
6.237.4 Field Documentation	1344
6.237.4.1 closed	1344
6.237.4.2 outputStream	1344
6.237.4.3 own	1344
6.238decaf::lang::Float Class Reference	1344
6.238.1 Constructor & Destructor Documentation	1346
6.238.1.1 Float	1346

6.238.1.2 Float	1347
6.238.1.3 Float	1347
6.238.1.4 ~Float	1347
6.238.2 Member Function Documentation	1347
6.238.2.1 byteValue	1347
6.238.2.2 compare	1347
6.238.2.3 compareTo	1348
6.238.2.4 compareTo	1348
6.238.2.5 doubleValue	1348
6.238.2.6 equals	1349
6.238.2.7 equals	1349
6.238.2.8 floatToIntBits	1349
6.238.2.9 floatToRawIntBits	1350
6.238.2.10 floatValue	1350
6.238.2.11 intBitsToFloat	1350
6.238.2.12 intValue	1351
6.238.2.13 isInfinite	1351
6.238.2.14 isInfinite	1351
6.238.2.15 isNaN	1351
6.238.2.16 isNaN	1351
6.238.2.17 longValue	1352
6.238.2.18 operator<	1352
6.238.2.19 operator<	1352
6.238.2.20 operator==	1353
6.238.2.21 operator==	1353
6.238.2.22 parseFloat	1353
6.238.2.23 shortValue	1354
6.238.2.24 toHexString	1354
6.238.2.25 toString	1355
6.238.2.26 toString	1355
6.238.2.27 valueOf	1355
6.238.2.28 valueOf	1356
6.238.3 Field Documentation	1356
6.238.3.1 MAX_VALUE	1356

6.238.3.2 MIN_VALUE	1356
6.238.3.3 NaN	1356
6.238.3.4 NEGATIVE_INFINITY	1356
6.238.3.5 POSITIVE_INFINITY	1357
6.238.3.6 SIZE	1357
6.239decaf::internal::nio::FloatArrayBuffer Class Reference	1357
6.239.1 Constructor & Destructor Documentation	1360
6.239.1.1 FloatArrayBuffer	1361
6.239.1.2 FloatArrayBuffer	1361
6.239.1.3 FloatArrayBuffer	1361
6.239.1.4 FloatArrayBuffer	1362
6.239.1.5 ~FloatArrayBuffer	1362
6.239.2 Member Function Documentation	1362
6.239.2.1 array	1362
6.239.2.2 arrayOffset	1363
6.239.2.3 asReadOnlyBuffer	1363
6.239.2.4 compact	1364
6.239.2.5 duplicate	1364
6.239.2.6 get	1364
6.239.2.7 get	1365
6.239.2.8 hasArray	1365
6.239.2.9 isReadOnly	1366
6.239.2.10put	1366
6.239.2.11put	1366
6.239.2.12setReadOnly	1367
6.239.2.13slice	1367
6.240decaf::nio::FloatBuffer Class Reference	1368
6.240.1 Detailed Description	1369
6.240.2 Constructor & Destructor Documentation	1370
6.240.2.1 FloatBuffer	1370
6.240.2.2 ~FloatBuffer	1370
6.240.3 Member Function Documentation	1370
6.240.3.1 allocate	1370
6.240.3.2 array	1370

6.240.3.3	arrayOffset	1371
6.240.3.4	asReadOnlyBuffer	1371
6.240.3.5	compact	1372
6.240.3.6	compareTo	1372
6.240.3.7	duplicate	1372
6.240.3.8	equals	1373
6.240.3.9	get	1373
6.240.3.10	get	1373
6.240.3.11	get	1374
6.240.3.12	get	1374
6.240.3.13	hasArray	1375
6.240.3.14	operator<	1375
6.240.3.15	operator==	1375
6.240.3.16	put	1375
6.240.3.17	put	1376
6.240.3.18	put	1377
6.240.3.19	put	1377
6.240.3.20	put	1378
6.240.3.21	slice	1378
6.240.3.22	toString	1378
6.240.3.23	wrap	1379
6.240.3.24	wrap	1379
6.241	decaf::io::Flushable Class Reference	1380
6.241.1	Detailed Description	1380
6.241.2	Constructor & Destructor Documentation	1380
6.241.2.1	~Flushable	1380
6.241.3	Member Function Documentation	1380
6.241.3.1	flush	1380
6.242	activemq::commands::FlushCommand Class Reference	1381
6.242.1	Constructor & Destructor Documentation	1381
6.242.1.1	FlushCommand	1381
6.242.1.2	~FlushCommand	1381
6.242.2	Member Function Documentation	1381
6.242.2.1	cloneDataStructure	1382

6.242.2.2 copyDataStructure	1382
6.242.2.3 equals	1382
6.242.2.4 getDataStructureType	1382
6.242.2.5 toString	1383
6.242.2.6 visit	1383
6.242.3 Field Documentation	1383
6.242.3.1 ID_FLUSHCOMMAND	1383
6.243activemq::wireformat::openwire::marshal::generated::FlushCommand- Marshaller Class Reference	1383
6.243.1 Detailed Description	1384
6.243.2 Constructor & Destructor Documentation	1384
6.243.2.1 FlushCommandMarshaller	1384
6.243.2.2 ~FlushCommandMarshaller	1384
6.243.3 Member Function Documentation	1384
6.243.3.1 createObject	1385
6.243.3.2 getDataStructureType	1385
6.243.3.3 looseMarshal	1385
6.243.3.4 looseUnmarshal	1386
6.243.3.5 tightMarshal1	1386
6.243.3.6 tightMarshal2	1386
6.243.3.7 tightUnmarshal	1387
6.244decaf::util::logging::Formatter Class Reference	1387
6.244.1 Detailed Description	1388
6.244.2 Constructor & Destructor Documentation	1388
6.244.2.1 ~Formatter	1388
6.244.3 Member Function Documentation	1388
6.244.3.1 format	1388
6.244.3.2 formatMessage	1389
6.244.3.3 getHead	1389
6.244.3.4 getTail	1389
6.245decaf::util::concurrent::Future< V > Class Template Reference	1390
6.245.1 Detailed Description	1390
6.245.2 Constructor & Destructor Documentation	1390
6.245.2.1 ~Future	1390

6.245.3 Member Function Documentation	1390
6.245.3.1 cancel	1391
6.245.3.2 get	1391
6.245.3.3 get	1392
6.245.3.4 isCancelled	1392
6.245.3.5 isDone	1392
6.246activemq::transport::correlator::FutureResponse Class Reference	1393
6.246.1 Detailed Description	1393
6.246.2 Constructor & Destructor Documentation	1393
6.246.2.1 FutureResponse	1393
6.246.2.2 ~FutureResponse	1393
6.246.3 Member Function Documentation	1393
6.246.3.1 getResponse	1394
6.246.3.2 getResponse	1394
6.246.3.3 getResponse	1394
6.246.3.4 getResponse	1394
6.246.3.5 setResponse	1394
6.247decaf::security::GeneralSecurityException Class Reference	1395
6.247.1 Constructor & Destructor Documentation	1395
6.247.1.1 GeneralSecurityException	1395
6.247.1.2 GeneralSecurityException	1395
6.247.1.3 GeneralSecurityException	1396
6.247.1.4 GeneralSecurityException	1396
6.247.1.5 GeneralSecurityException	1396
6.247.1.6 GeneralSecurityException	1396
6.247.1.7 ~GeneralSecurityException	1397
6.247.2 Member Function Documentation	1397
6.247.2.1 clone	1397
6.248decaf::internal::util::GenericResource< T > Class Template Reference	1397
6.248.1 Detailed Description	1398
6.248.2 Constructor & Destructor Documentation	1398
6.248.2.1 GenericResource	1398
6.248.2.2 ~GenericResource	1398
6.248.3 Member Function Documentation	1398

6.248.3.1 getManaged	1398
6.248.3.2 setManaged	1398
6.249gz_header_s Struct Reference	1398
6.249.1 Field Documentation	1399
6.249.1.1 comm_max	1399
6.249.1.2 comment	1399
6.249.1.3 done	1399
6.249.1.4 extra	1399
6.249.1.5 extra_len	1399
6.249.1.6 extra_max	1399
6.249.1.7 hcrc	1399
6.249.1.8 name	1399
6.249.1.9 name_max	1399
6.249.1.10bs	1399
6.249.1.11text	1399
6.249.1.12ime	1399
6.249.1.13xflags	1399
6.250gz_state Struct Reference	1400
6.250.1 Field Documentation	1400
6.250.1.1 direct	1400
6.250.1.2 eof	1400
6.250.1.3 err	1400
6.250.1.4 fd	1400
6.250.1.5 have	1400
6.250.1.6 how	1400
6.250.1.7 in	1400
6.250.1.8 level	1401
6.250.1.9 mode	1401
6.250.1.10msg	1401
6.250.1.11next	1401
6.250.1.12but	1401
6.250.1.13path	1401
6.250.1.14pos	1401
6.250.1.15raw	1401

6.250.1.16seek	1401
6.250.1.17size	1401
6.250.1.18skip	1401
6.250.1.19start	1401
6.250.1.20strategy	1401
6.250.1.21strm	1401
6.250.1.22want	1401
6.251decaf::util::logging::Handler Class Reference	1401
6.251.1 Detailed Description	1402
6.251.2 Constructor & Destructor Documentation	1403
6.251.2.1 Handler	1403
6.251.2.2 ~Handler	1403
6.251.3 Member Function Documentation	1403
6.251.3.1 flush	1403
6.251.3.2 getErrorManager	1403
6.251.3.3 getFilter	1403
6.251.3.4 getFormatter	1403
6.251.3.5 getLevel	1403
6.251.3.6 isLoggable	1404
6.251.3.7 publish	1404
6.251.3.8 reportError	1404
6.251.3.9 setErrorManager	1404
6.251.3.10setFilter	1405
6.251.3.11setFormatter	1405
6.251.3.12setLevel	1405
6.252decaf::internal::util::HexStringParser Class Reference	1406
6.252.1 Constructor & Destructor Documentation	1406
6.252.1.1 HexStringParser	1406
6.252.1.2 ~HexStringParser	1406
6.252.2 Member Function Documentation	1406
6.252.2.1 parse	1406
6.252.2.2 parseDouble	1407
6.252.2.3 parseFloat	1407
6.253activemq::wireformat::openwire::utils::HexTable Class Reference	1407

6.253.1 Detailed Description	1407
6.253.2 Constructor & Destructor Documentation	1408
6.253.2.1 HexTable	1408
6.253.2.2 ~HexTable	1408
6.253.3 Member Function Documentation	1408
6.253.3.1 operator[]	1408
6.253.3.2 operator[]	1408
6.253.3.3 size	1408
6.254decaf::net::HttpRetryException Class Reference	1408
6.254.1 Constructor & Destructor Documentation	1409
6.254.1.1 HttpRetryException	1409
6.254.1.2 HttpRetryException	1409
6.254.1.3 HttpRetryException	1409
6.254.1.4 HttpRetryException	1410
6.254.1.5 HttpRetryException	1410
6.254.1.6 HttpRetryException	1410
6.254.1.7 ~HttpRetryException	1411
6.254.2 Member Function Documentation	1411
6.254.2.1 clone	1411
6.255activemq::util::IdGenerator Class Reference	1411
6.255.1 Constructor & Destructor Documentation	1412
6.255.1.1 IdGenerator	1412
6.255.1.2 IdGenerator	1412
6.255.1.3 ~IdGenerator	1412
6.255.2 Member Function Documentation	1412
6.255.2.1 compare	1412
6.255.2.2 generateId	1412
6.255.2.3 getHostname	1412
6.255.2.4 getSeedFromId	1413
6.255.2.5 getSequenceFromId	1413
6.255.3 Friends And Related Function Documentation	1413
6.255.3.1 activemq::library::ActiveMQCPP	1413
6.256decaf::lang::exceptions::IllegalArgumentException Class Reference . . .	1413
6.256.1 Constructor & Destructor Documentation	1414

6.256.1.1	IllegalArgumentException	1414
6.256.1.2	IllegalArgumentException	1414
6.256.1.3	IllegalArgumentException	1414
6.256.1.4	IllegalArgumentException	1414
6.256.1.5	IllegalArgumentException	1415
6.256.1.6	IllegalArgumentException	1415
6.256.1.7	~IllegalArgumentException	1415
6.256.2	Member Function Documentation	1415
6.256.2.1	clone	1416
6.257	decaf::lang::exceptions::IllegalMonitorStateException Class Reference .	1416
6.257.1	Constructor & Destructor Documentation	1417
6.257.1.1	IllegalMonitorStateException	1417
6.257.1.2	IllegalMonitorStateException	1417
6.257.1.3	IllegalMonitorStateException	1417
6.257.1.4	IllegalMonitorStateException	1417
6.257.1.5	IllegalMonitorStateException	1418
6.257.1.6	IllegalMonitorStateException	1418
6.257.1.7	~IllegalMonitorStateException	1418
6.257.2	Member Function Documentation	1418
6.257.2.1	clone	1418
6.258	cms::IllegalStateException Class Reference	1419
6.258.1	Detailed Description	1419
6.258.2	Constructor & Destructor Documentation	1419
6.258.2.1	IllegalStateException	1419
6.258.2.2	IllegalStateException	1419
6.258.2.3	IllegalStateException	1419
6.258.2.4	IllegalStateException	1420
6.258.2.5	IllegalStateException	1420
6.258.2.6	~IllegalStateException	1420
6.259	decaf::lang::exceptions::IllegalStateException Class Reference	1420
6.259.1	Constructor & Destructor Documentation	1421
6.259.1.1	IllegalStateException	1421
6.259.1.2	IllegalStateException	1421
6.259.1.3	IllegalStateException	1421

6.259.1.4	IllegalStateException	1421
6.259.1.5	IllegalStateException	1421
6.259.1.6	IllegalStateException	1422
6.259.1.7	~IllegalStateException	1422
6.259.2	Member Function Documentation	1422
6.259.2.1	clone	1422
6.260	decaf::lang::exceptions::IllegalThreadStateException Class Reference	1423
6.260.1	Constructor & Destructor Documentation	1423
6.260.1.1	IllegalThreadStateException	1423
6.260.1.2	IllegalThreadStateException	1423
6.260.1.3	IllegalThreadStateException	1424
6.260.1.4	IllegalThreadStateException	1424
6.260.1.5	IllegalThreadStateException	1424
6.260.1.6	IllegalThreadStateException	1425
6.260.1.7	~IllegalThreadStateException	1425
6.260.2	Member Function Documentation	1425
6.260.2.1	clone	1425
6.261	activemq::transport::inactivity::InactivityMonitor Class Reference	1425
6.261.1	Constructor & Destructor Documentation	1426
6.261.1.1	InactivityMonitor	1426
6.261.1.2	InactivityMonitor	1426
6.261.1.3	~InactivityMonitor	1426
6.261.2	Member Function Documentation	1427
6.261.2.1	close	1427
6.261.2.2	getInitialDelayTime	1427
6.261.2.3	getReadCheckTime	1427
6.261.2.4	getWriteCheckTime	1427
6.261.2.5	isKeepAliveResponseRequired	1427
6.261.2.6	onCommand	1427
6.261.2.7	oneway	1427
6.261.2.8	onException	1428
6.261.2.9	setInitialDelayTime	1428
6.261.2.10	setKeepAliveResponseRequired	1428
6.261.2.11	setReadCheckTime	1428

6.261.2.12 setWriteCheckTime	1428
6.261.3 Friends And Related Function Documentation	1428
6.261.3.1 AsyncSignalReadErrorkTask	1428
6.261.3.2 AsyncWriteTask	1428
6.261.3.3 ReadChecker	1428
6.261.3.4 WriteChecker	1428
6.262decaf::lang::exceptions::IndexOutOfBoundsException Class Reference .	1429
6.262.1 Constructor & Destructor Documentation	1429
6.262.1.1 IndexOutOfBoundsException	1429
6.262.1.2 IndexOutOfBoundsException	1429
6.262.1.3 IndexOutOfBoundsException	1430
6.262.1.4 IndexOutOfBoundsException	1430
6.262.1.5 IndexOutOfBoundsException	1430
6.262.1.6 IndexOutOfBoundsException	1431
6.262.1.7 ~IndexOutOfBoundsException	1431
6.262.2 Member Function Documentation	1431
6.262.2.1 clone	1431
6.263decaf::net::Inet4Address Class Reference	1431
6.263.1 Constructor & Destructor Documentation	1432
6.263.1.1 Inet4Address	1432
6.263.1.2 Inet4Address	1432
6.263.1.3 Inet4Address	1432
6.263.1.4 ~Inet4Address	1433
6.263.2 Member Function Documentation	1433
6.263.2.1 clone	1433
6.263.2.2 isAnyLocalAddress	1433
6.263.2.3 isLinkLocalAddress	1433
6.263.2.4 isLoopbackAddress	1433
6.263.2.5 isMCGlobal	1434
6.263.2.6 isMCLinkLocal	1434
6.263.2.7 isMCNodeLocal	1434
6.263.2.8 isMCOrgLocal	1434
6.263.2.9 isMCSiteLocal	1434
6.263.2.10 isMulticastAddress	1435

6.263.2.1 <code>isSiteLocalAddress</code>	1435
6.263.3 Friends And Related Function Documentation	1435
6.263.3.1 <code>InetAddress</code>	1435
6.264 <code>decaf::net::Inet6Address</code> Class Reference	1435
6.264.1 Constructor & Destructor Documentation	1436
6.264.1.1 <code>Inet6Address</code>	1436
6.264.1.2 <code>Inet6Address</code>	1436
6.264.1.3 <code>Inet6Address</code>	1436
6.264.1.4 <code>~Inet6Address</code>	1436
6.264.2 Member Function Documentation	1436
6.264.2.1 <code>clone</code>	1436
6.264.3 Friends And Related Function Documentation	1437
6.264.3.1 <code>InetAddress</code>	1437
6.265 <code>decaf::net::InetAddress</code> Class Reference	1437
6.265.1 Detailed Description	1439
6.265.2 Constructor & Destructor Documentation	1439
6.265.2.1 <code>InetAddress</code>	1439
6.265.2.2 <code>InetAddress</code>	1439
6.265.2.3 <code>InetAddress</code>	1439
6.265.2.4 <code>~InetAddress</code>	1439
6.265.3 Member Function Documentation	1439
6.265.3.1 <code>bytesToInt</code>	1439
6.265.3.2 <code>clone</code>	1439
6.265.3.3 <code>getAddress</code>	1440
6.265.3.4 <code>getAnyAddress</code>	1440
6.265.3.5 <code>getByAddress</code>	1440
6.265.3.6 <code>getByAddress</code>	1440
6.265.3.7 <code>getHostAddress</code>	1441
6.265.3.8 <code>getHostName</code>	1441
6.265.3.9 <code>getLocalHost</code>	1441
6.265.3.10 <code>getLoopbackAddress</code>	1442
6.265.3.11 <code>isAnyLocalAddress</code>	1442
6.265.3.12 <code>sLinkLocalAddress</code>	1442
6.265.3.13 <code>sLoopbackAddress</code>	1442

6.265.3.14sMCGlobal	1442
6.265.3.15sMCLinkLocal	1443
6.265.3.16sMCNodeLocal	1443
6.265.3.17sMCOrgLocal	1443
6.265.3.18sMCSiteLocal	1443
6.265.3.19sMulticastAddress	1444
6.265.3.20sSiteLocalAddress	1444
6.265.3.21toString	1444
6.265.4 Field Documentation	1444
6.265.4.1 addressBytes	1444
6.265.4.2 anyBytes	1444
6.265.4.3 hostname	1444
6.265.4.4 loopbackBytes	1444
6.265.4.5 reached	1445
6.266decaf::net::InetSocketAddress Class Reference	1445
6.266.1 Constructor & Destructor Documentation	1445
6.266.1.1 InetSocketAddress	1445
6.266.1.2 ~InetSocketAddress	1445
6.267inflate_state Struct Reference	1445
6.267.1 Field Documentation	1446
6.267.1.1 back	1446
6.267.1.2 bits	1446
6.267.1.3 check	1446
6.267.1.4 codes	1446
6.267.1.5 distbits	1446
6.267.1.6 distcode	1446
6.267.1.7 dmax	1446
6.267.1.8 extra	1446
6.267.1.9 flags	1447
6.267.1.10have	1447
6.267.1.11havedict	1447
6.267.1.12head	1447
6.267.1.13hold	1447
6.267.1.14last	1447

6.267.1.15enbits	1447
6.267.1.16encode	1447
6.267.1.17length	1447
6.267.1.18ens	1447
6.267.1.19mode	1447
6.267.1.20ncode	1447
6.267.1.21ndist	1447
6.267.1.22next	1447
6.267.1.23hlen	1447
6.267.1.24offset	1447
6.267.1.25sane	1447
6.267.1.26total	1447
6.267.1.27was	1447
6.267.1.28wbits	1447
6.267.1.29whave	1447
6.267.1.30window	1447
6.267.1.31wnext	1447
6.267.1.32work	1447
6.267.1.33wrap	1448
6.267.1.34wsize	1448
6.268decaf::util::zip::Inflater Class Reference	1448
6.268.1 Detailed Description	1449
6.268.2 Constructor & Destructor Documentation	1449
6.268.2.1 Inflater	1449
6.268.2.2 Inflater	1449
6.268.2.3 ~Inflater	1450
6.268.3 Member Function Documentation	1450
6.268.3.1 end	1450
6.268.3.2 finish	1450
6.268.3.3 finished	1450
6.268.3.4 getAdler	1450
6.268.3.5 getBytesRead	1450
6.268.3.6 getBytesWritten	1451
6.268.3.7 getRemaining	1451

6.268.3.8 inflate	1451
6.268.3.9 inflate	1452
6.268.3.10 inflate	1452
6.268.3.11 needsDictionary	1453
6.268.3.12 needsInput	1453
6.268.3.13 reset	1453
6.268.3.14 setDictionary	1453
6.268.3.15 setDictionary	1454
6.268.3.16 setDictionary	1454
6.268.3.17 setInput	1455
6.268.3.18 setInput	1455
6.268.3.19 setInput	1456
6.269 decaf::util::zip::InflaterInputStream Class Reference	1456
6.269.1 Detailed Description	1459
6.269.2 Constructor & Destructor Documentation	1459
6.269.2.1 InflaterInputStream	1459
6.269.2.2 InflaterInputStream	1459
6.269.2.3 InflaterInputStream	1460
6.269.2.4 ~InflaterInputStream	1460
6.269.3 Member Function Documentation	1460
6.269.3.1 available	1460
6.269.3.2 close	1461
6.269.3.3 doReadArrayBounded	1461
6.269.3.4 doReadByte	1461
6.269.3.5 fill	1461
6.269.3.6 mark	1462
6.269.3.7 markSupported	1462
6.269.3.8 reset	1462
6.269.3.9 skip	1463
6.269.4 Field Documentation	1464
6.269.4.1 atEOF	1464
6.269.4.2 buff	1464
6.269.4.3 DEFAULT_BUFFER_SIZE	1464
6.269.4.4 inflater	1464

6.269.4.5 length	1464
6.269.4.6 ownInflater	1464
6.270decaf::io::InputStream Class Reference	1464
6.270.1 Detailed Description	1466
6.270.2 Constructor & Destructor Documentation	1466
6.270.2.1 InputStream	1466
6.270.2.2 ~InputStream	1466
6.270.3 Member Function Documentation	1466
6.270.3.1 available	1466
6.270.3.2 close	1466
6.270.3.3 doReadArray	1467
6.270.3.4 doReadArrayBounded	1467
6.270.3.5 doReadByte	1467
6.270.3.6 lock	1467
6.270.3.7 mark	1468
6.270.3.8 markSupported	1468
6.270.3.9 notify	1468
6.270.3.10notifyAll	1469
6.270.3.11read	1469
6.270.3.12read	1469
6.270.3.13read	1470
6.270.3.14reset	1471
6.270.3.15skip	1472
6.270.3.16toString	1472
6.270.3.17tryLock	1473
6.270.3.18unlock	1473
6.270.3.19wait	1473
6.270.3.20wait	1474
6.270.3.21wait	1474
6.271decaf::io::InputStreamReader Class Reference	1475
6.271.1 Detailed Description	1475
6.271.2 Constructor & Destructor Documentation	1476
6.271.2.1 InputStreamReader	1476
6.271.2.2 ~InputStreamReader	1476

6.271.3 Member Function Documentation	1476
6.271.3.1 checkClosed	1476
6.271.3.2 close	1476
6.271.3.3 doReadArrayBounded	1476
6.271.3.4 ready	1477
6.272decaf::internal::nio::IntArrayBuffer Class Reference	1477
6.272.1 Constructor & Destructor Documentation	1481
6.272.1.1 IntArrayBuffer	1481
6.272.1.2 IntArrayBuffer	1481
6.272.1.3 IntArrayBuffer	1482
6.272.1.4 IntArrayBuffer	1482
6.272.1.5 ~IntArrayBuffer	1482
6.272.2 Member Function Documentation	1482
6.272.2.1 array	1482
6.272.2.2 arrayOffset	1483
6.272.2.3 asReadOnlyBuffer	1483
6.272.2.4 compact	1484
6.272.2.5 duplicate	1484
6.272.2.6 get	1485
6.272.2.7 get	1485
6.272.2.8 hasArray	1486
6.272.2.9 isReadOnly	1486
6.272.2.10put	1486
6.272.2.11put	1487
6.272.2.12setReadOnly	1487
6.272.2.13slice	1487
6.273decaf::nio::IntBuffer Class Reference	1488
6.273.1 Detailed Description	1489
6.273.2 Constructor & Destructor Documentation	1490
6.273.2.1 IntBuffer	1490
6.273.2.2 ~IntBuffer	1490
6.273.3 Member Function Documentation	1490
6.273.3.1 allocate	1490
6.273.3.2 array	1491

6.273.3.3	arrayOffset	1491
6.273.3.4	asReadOnlyBuffer	1492
6.273.3.5	compact	1492
6.273.3.6	compareTo	1492
6.273.3.7	duplicate	1493
6.273.3.8	equals	1493
6.273.3.9	get	1493
6.273.3.10	get	1493
6.273.3.11	get	1494
6.273.3.12	get	1494
6.273.3.13	hasArray	1495
6.273.3.14	operator<	1495
6.273.3.15	operator==	1495
6.273.3.16	put	1495
6.273.3.17	put	1496
6.273.3.18	put	1497
6.273.3.19	put	1497
6.273.3.20	put	1498
6.273.3.21	slice	1498
6.273.3.22	toString	1498
6.273.3.23	wrap	1499
6.273.3.24	wrap	1499
6.274	decaf::lang::Integer Class Reference	1500
6.274.1	Constructor & Destructor Documentation	1502
6.274.1.1	Integer	1502
6.274.1.2	Integer	1502
6.274.1.3	~Integer	1503
6.274.2	Member Function Documentation	1503
6.274.2.1	bitCount	1503
6.274.2.2	byteValue	1503
6.274.2.3	compareTo	1503
6.274.2.4	compareTo	1504
6.274.2.5	decode	1504
6.274.2.6	doubleValue	1504

6.274.2.7 equals	1505
6.274.2.8 equals	1505
6.274.2.9 floatValue	1505
6.274.2.10highestOneBit	1505
6.274.2.11intValue	1506
6.274.2.12longValue	1506
6.274.2.13lowestOneBit	1506
6.274.2.14numberOfLeadingZeros	1507
6.274.2.15numberOfTrailingZeros	1507
6.274.2.16operator<	1507
6.274.2.17operator<	1508
6.274.2.18operator==	1508
6.274.2.19operator==	1508
6.274.2.20parseInt	1509
6.274.2.21parseInt	1509
6.274.2.22reverse	1510
6.274.2.23reverseBytes	1510
6.274.2.24rotateLeft	1510
6.274.2.25rotateRight	1511
6.274.2.26shortValue	1511
6.274.2.27signum	1512
6.274.2.28toBinaryString	1512
6.274.2.29toHexString	1512
6.274.2.30toOctalString	1513
6.274.2.31toString	1513
6.274.2.32toString	1514
6.274.2.33toString	1514
6.274.2.34valueOf	1514
6.274.2.35valueOf	1515
6.274.2.36valueOf	1515
6.274.3 Field Documentation	1516
6.274.3.1 MAX_VALUE	1516
6.274.3.2 MIN_VALUE	1516
6.274.3.3 SIZE	1516

6.275	activemq::commands::IntegerResponse Class Reference	1516
6.275.1	Constructor & Destructor Documentation	1517
6.275.1.1	IntegerResponse	1517
6.275.1.2	~IntegerResponse	1517
6.275.2	Member Function Documentation	1517
6.275.2.1	cloneDataStructure	1517
6.275.2.2	copyDataStructure	1517
6.275.2.3	equals	1518
6.275.2.4	getDataStructureType	1518
6.275.2.5	getResult	1518
6.275.2.6	setResult	1518
6.275.2.7	toString	1518
6.275.3	Field Documentation	1519
6.275.3.1	ID_INTEGERRESPONSE	1519
6.275.3.2	result	1519
6.276	activemq::wireformat::openwire::marshal::generated::IntegerResponse- Marshaller Class Reference	1519
6.276.1	Detailed Description	1520
6.276.2	Constructor & Destructor Documentation	1520
6.276.2.1	IntegerResponseMarshaller	1520
6.276.2.2	~IntegerResponseMarshaller	1520
6.276.3	Member Function Documentation	1520
6.276.3.1	createObject	1520
6.276.3.2	getDataStructureType	1520
6.276.3.3	looseMarshal	1521
6.276.3.4	looseUnmarshal	1521
6.276.3.5	tightMarshal1	1521
6.276.3.6	tightMarshal2	1522
6.276.3.7	tightUnmarshal	1522
6.277	internal_state Struct Reference	1523
6.277.1	Field Documentation	1524
6.277.1.1	bi_buf	1524
6.277.1.2	bi_valid	1524
6.277.1.3	bl_count	1524

6.277.1.4 bl_desc	1524
6.277.1.5 bl_tree	1524
6.277.1.6 block_start	1524
6.277.1.7 d_buf	1525
6.277.1.8 d_desc	1525
6.277.1.9 depth	1525
6.277.1.10dummy	1525
6.277.1.11dyn_dtree	1525
6.277.1.12dyn_ltree	1525
6.277.1.13good_match	1525
6.277.1.14gzhead	1525
6.277.1.15gzindex	1525
6.277.1.16hash_bits	1525
6.277.1.17hash_mask	1525
6.277.1.18hash_shift	1525
6.277.1.19hash_size	1525
6.277.1.20head	1525
6.277.1.21heap	1525
6.277.1.22heap_len	1525
6.277.1.23heap_max	1525
6.277.1.24high_water	1525
6.277.1.25ns_h	1525
6.277.1.26_buf	1525
6.277.1.27_desc	1525
6.277.1.28ast_eob_len	1525
6.277.1.29ast_flush	1525
6.277.1.30ast_lit	1525
6.277.1.31level	1526
6.277.1.32lit_bufsize	1526
6.277.1.33lookahead	1526
6.277.1.34match_available	1526
6.277.1.35match_length	1526
6.277.1.36match_start	1526
6.277.1.37matches	1526

6.277.1.38	max_chain_length	1526
6.277.1.39	max_lazy_match	1526
6.277.1.40	method	1526
6.277.1.41	nice_match	1526
6.277.1.42	opt_len	1526
6.277.1.43	pending	1526
6.277.1.44	pending_buf	1526
6.277.1.45	pending_buf_size	1526
6.277.1.46	pending_out	1526
6.277.1.47	prev	1526
6.277.1.48	prev_length	1526
6.277.1.49	prev_match	1526
6.277.1.50	static_len	1526
6.277.1.51	status	1526
6.277.1.52	strategy	1526
6.277.1.53	strm	1526
6.277.1.54	strstart	1526
6.277.1.55	w_bits	1527
6.277.1.56	w_mask	1527
6.277.1.57	w_size	1527
6.277.1.58	window	1527
6.277.1.59	window_size	1527
6.277.1.60	wrap	1527
6.278	activemq::transport::mock::InternalCommandListener Class Reference	1527
6.278.1	Detailed Description	1527
6.278.2	Constructor & Destructor Documentation	1528
6.278.2.1	InternalCommandListener	1528
6.278.2.2	~InternalCommandListener	1528
6.278.3	Member Function Documentation	1528
6.278.3.1	onCommand	1528
6.278.3.2	run	1528
6.278.3.3	setResponseBuilder	1528
6.278.3.4	setTransport	1528
6.279	decaf::lang::exceptions::InterruptedException Class Reference	1529

6.279.1 Constructor & Destructor Documentation	1529
6.279.1.1 InterruptedException	1529
6.279.1.2 InterruptedException	1529
6.279.1.3 InterruptedException	1530
6.279.1.4 InterruptedException	1530
6.279.1.5 InterruptedException	1530
6.279.1.6 InterruptedException	1530
6.279.1.7 ~InterruptedException	1531
6.279.2 Member Function Documentation	1531
6.279.2.1 clone	1531
6.280decaf::io::InterruptedException Class Reference	1531
6.280.1 Constructor & Destructor Documentation	1532
6.280.1.1 InterruptedException	1532
6.280.1.2 InterruptedException	1532
6.280.1.3 InterruptedException	1532
6.280.1.4 InterruptedException	1532
6.280.1.5 InterruptedException	1533
6.280.1.6 InterruptedException	1533
6.280.1.7 ~InterruptedException	1533
6.280.2 Member Function Documentation	1533
6.280.2.1 clone	1533
6.281cms::InvalidClientIdException Class Reference	1534
6.281.1 Detailed Description	1534
6.281.2 Constructor & Destructor Documentation	1534
6.281.2.1 InvalidClientIdException	1534
6.281.2.2 InvalidClientIdException	1535
6.281.2.3 InvalidClientIdException	1535
6.281.2.4 InvalidClientIdException	1535
6.281.2.5 InvalidClientIdException	1535
6.281.2.6 ~InvalidClientIdException	1535
6.282cms::InvalidDestinationException Class Reference	1535
6.282.1 Detailed Description	1535
6.282.2 Constructor & Destructor Documentation	1536
6.282.2.1 InvalidDestinationException	1536

6.282.2.2 InvalidDestinationException	1536
6.282.2.3 InvalidDestinationException	1536
6.282.2.4 InvalidDestinationException	1536
6.282.2.5 InvalidDestinationException	1536
6.282.2.6 ~InvalidDestinationException	1536
6.283decaf::security::InvalidKeyException Class Reference	1536
6.283.1 Constructor & Destructor Documentation	1537
6.283.1.1 InvalidKeyException	1537
6.283.1.2 InvalidKeyException	1537
6.283.1.3 InvalidKeyException	1537
6.283.1.4 InvalidKeyException	1537
6.283.1.5 InvalidKeyException	1538
6.283.1.6 InvalidKeyException	1538
6.283.1.7 ~InvalidKeyException	1538
6.283.2 Member Function Documentation	1538
6.283.2.1 clone	1538
6.284decaf::nio::InvalidMarkException Class Reference	1539
6.284.1 Constructor & Destructor Documentation	1539
6.284.1.1 InvalidMarkException	1539
6.284.1.2 InvalidMarkException	1540
6.284.1.3 InvalidMarkException	1540
6.284.1.4 InvalidMarkException	1540
6.284.1.5 InvalidMarkException	1540
6.284.1.6 InvalidMarkException	1541
6.284.1.7 ~InvalidMarkException	1541
6.284.2 Member Function Documentation	1541
6.284.2.1 clone	1541
6.285cms::InvalidSelectorException Class Reference	1541
6.285.1 Detailed Description	1542
6.285.2 Constructor & Destructor Documentation	1542
6.285.2.1 InvalidSelectorException	1542
6.285.2.2 InvalidSelectorException	1542
6.285.2.3 InvalidSelectorException	1542
6.285.2.4 InvalidSelectorException	1542

6.285.2.5 InvalidSelectorException	1542
6.285.2.6 ~InvalidSelectorException	1542
6.286decaf::lang::exceptions::InvalidStateException Class Reference	1543
6.286.1 Constructor & Destructor Documentation	1543
6.286.1.1 InvalidStateException	1543
6.286.1.2 InvalidStateException	1543
6.286.1.3 InvalidStateException	1544
6.286.1.4 InvalidStateException	1544
6.286.1.5 InvalidStateException	1544
6.286.1.6 InvalidStateException	1544
6.286.1.7 ~InvalidStateException	1545
6.286.2 Member Function Documentation	1545
6.286.2.1 clone	1545
6.287decaf::io::IOException Class Reference	1545
6.287.1 Constructor & Destructor Documentation	1546
6.287.1.1 IOException	1546
6.287.1.2 IOException	1546
6.287.1.3 IOException	1546
6.287.1.4 IOException	1546
6.287.1.5 IOException	1547
6.287.1.6 IOException	1547
6.287.1.7 ~IOException	1547
6.287.2 Member Function Documentation	1547
6.287.2.1 clone	1547
6.288activemq::transport::IOTransport Class Reference	1548
6.288.1 Detailed Description	1550
6.288.2 Constructor & Destructor Documentation	1550
6.288.2.1 IOTransport	1550
6.288.2.2 IOTransport	1550
6.288.2.3 ~IOTransport	1550
6.288.3 Member Function Documentation	1551
6.288.3.1 close	1551
6.288.3.2 getRemoteAddress	1551
6.288.3.3 getTransportListener	1551

6.288.3.4	getWireFormat	1551
6.288.3.5	isClosed	1552
6.288.3.6	isConnected	1552
6.288.3.7	isFaultTolerant	1552
6.288.3.8	isReconnectSupported	1552
6.288.3.9	isUpdateURIsSupported	1552
6.288.3.10	narrow	1553
6.288.3.11	oneway	1553
6.288.3.12	reconnect	1553
6.288.3.13	request	1553
6.288.3.14	request	1554
6.288.3.15	run	1554
6.288.3.16	setInputStream	1555
6.288.3.17	setOutputStream	1555
6.288.3.18	setTransportListener	1555
6.288.3.19	setWireFormat	1555
6.288.3.20	start	1555
6.288.3.21	stop	1556
6.288.3.22	updateURIs	1556
6.289	decaf::lang::Iterable< E > Class Template Reference	1556
6.289.1	Detailed Description	1556
6.289.2	Constructor & Destructor Documentation	1557
6.289.2.1	~Iterable	1557
6.289.3	Member Function Documentation	1557
6.289.3.1	iterator	1557
6.289.3.2	iterator	1558
6.290	decaf::util::Iterator< E > Class Template Reference	1559
6.290.1	Detailed Description	1559
6.290.2	Constructor & Destructor Documentation	1559
6.290.2.1	~Iterator	1559
6.290.3	Member Function Documentation	1559
6.290.3.1	hasNext	1559
6.290.3.2	next	1560
6.290.3.3	remove	1560

6.291 activemq::commands::JournalQueueAck Class Reference	1561
6.291.1 Constructor & Destructor Documentation	1562
6.291.1.1 JournalQueueAck	1562
6.291.1.2 ~JournalQueueAck	1562
6.291.2 Member Function Documentation	1562
6.291.2.1 cloneDataStructure	1562
6.291.2.2 copyDataStructure	1562
6.291.2.3 equals	1562
6.291.2.4 getDataStructureType	1563
6.291.2.5 getDestination	1563
6.291.2.6 getDestination	1563
6.291.2.7 getMessageAck	1563
6.291.2.8 getMessageAck	1563
6.291.2.9 setDestination	1563
6.291.2.10 setMessageAck	1563
6.291.2.11 toString	1563
6.291.3 Field Documentation	1563
6.291.3.1 destination	1563
6.291.3.2 ID_JOURNALQUEUEACK	1564
6.291.3.3 messageAck	1564
6.292 activemq::wireformat::openwire::marshal::generated::JournalQueue- AckMarshaller Class Reference	1564
6.292.1 Detailed Description	1565
6.292.2 Constructor & Destructor Documentation	1565
6.292.2.1 JournalQueueAckMarshaller	1565
6.292.2.2 ~JournalQueueAckMarshaller	1565
6.292.3 Member Function Documentation	1565
6.292.3.1 createObject	1565
6.292.3.2 getDataStructureType	1565
6.292.3.3 looseMarshal	1566
6.292.3.4 looseUnmarshal	1566
6.292.3.5 tightMarshal1	1566
6.292.3.6 tightMarshal2	1567
6.292.3.7 tightUnmarshal	1567

6.293activemq::commands::JournalTopicAck Class Reference	1568
6.293.1 Constructor & Destructor Documentation	1569
6.293.1.1 JournalTopicAck	1569
6.293.1.2 ~JournalTopicAck	1569
6.293.2 Member Function Documentation	1569
6.293.2.1 cloneDataStructure	1569
6.293.2.2 copyDataStructure	1570
6.293.2.3 equals	1570
6.293.2.4 getClientId	1570
6.293.2.5 getClientId	1570
6.293.2.6 getDataStructureType	1570
6.293.2.7 getDestination	1570
6.293.2.8 getDestination	1571
6.293.2.9 getMessageId	1571
6.293.2.10getMessageId	1571
6.293.2.11getMessageSequenceId	1571
6.293.2.12getSubscriptionName	1571
6.293.2.13getSubscriptionName	1571
6.293.2.14getTransactionId	1571
6.293.2.15getTransactionId	1571
6.293.2.16setClientId	1571
6.293.2.17setDestination	1571
6.293.2.18setMessageId	1571
6.293.2.19setMessageSequenceId	1571
6.293.2.20setSubscriptionName	1571
6.293.2.21setTransactionId	1571
6.293.2.22toString	1572
6.293.3 Field Documentation	1572
6.293.3.1 clientId	1572
6.293.3.2 destination	1572
6.293.3.3 ID_JOURNALTOPICACK	1572
6.293.3.4 messageId	1572
6.293.3.5 messageSequenceId	1572
6.293.3.6 subscriptionName	1572

6.293.3.7 transactionId	1572
6.294activemq::wireformat::openwire::marshal::generated::JournalTopicAck- Marshaller Class Reference	1572
6.294.1 Detailed Description	1573
6.294.2 Constructor & Destructor Documentation	1573
6.294.2.1 JournalTopicAckMarshaller	1573
6.294.2.2 ~JournalTopicAckMarshaller	1574
6.294.3 Member Function Documentation	1574
6.294.3.1 createObject	1574
6.294.3.2 getDataStructureType	1574
6.294.3.3 looseMarshal	1574
6.294.3.4 looseUnmarshal	1575
6.294.3.5 tightMarshal1	1575
6.294.3.6 tightMarshal2	1576
6.294.3.7 tightUnmarshal	1576
6.295activemq::commands::JournalTrace Class Reference	1577
6.295.1 Constructor & Destructor Documentation	1577
6.295.1.1 JournalTrace	1577
6.295.1.2 ~JournalTrace	1577
6.295.2 Member Function Documentation	1578
6.295.2.1 cloneDataStructure	1578
6.295.2.2 copyDataStructure	1578
6.295.2.3 equals	1578
6.295.2.4 getDataStructureType	1578
6.295.2.5 getMessage	1579
6.295.2.6 getMessage	1579
6.295.2.7 setMessage	1579
6.295.2.8 toString	1579
6.295.3 Field Documentation	1579
6.295.3.1 ID_JOURNALTRACE	1579
6.295.3.2 message	1579
6.296activemq::wireformat::openwire::marshal::generated::JournalTrace- Marshaller Class Reference	1579
6.296.1 Detailed Description	1580

6.296.2 Constructor & Destructor Documentation	1580
6.296.2.1 JournalTraceMarshaller	1580
6.296.2.2 ~JournalTraceMarshaller	1580
6.296.3 Member Function Documentation	1580
6.296.3.1 createObject	1581
6.296.3.2 getDataStructureType	1581
6.296.3.3 looseMarshal	1581
6.296.3.4 looseUnmarshal	1582
6.296.3.5 tightMarshal1	1582
6.296.3.6 tightMarshal2	1582
6.296.3.7 tightUnmarshal	1583
6.297activemq::commands::JournalTransaction Class Reference	1583
6.297.1 Constructor & Destructor Documentation	1584
6.297.1.1 JournalTransaction	1584
6.297.1.2 ~JournalTransaction	1584
6.297.2 Member Function Documentation	1585
6.297.2.1 cloneDataStructure	1585
6.297.2.2 copyDataStructure	1585
6.297.2.3 equals	1585
6.297.2.4 getDataStructureType	1585
6.297.2.5 getTransactionId	1586
6.297.2.6 getTransactionId	1586
6.297.2.7 getType	1586
6.297.2.8 getWasPrepared	1586
6.297.2.9 setTransactionId	1586
6.297.2.10setType	1586
6.297.2.11setWasPrepared	1586
6.297.2.12toString	1586
6.297.3 Field Documentation	1586
6.297.3.1 ID_JOURNALTRANSACTION	1586
6.297.3.2 transactionId	1587
6.297.3.3 type	1587
6.297.3.4 wasPrepared	1587

6.298activemq::wireformat::openwire::marshal::generated::JournalTransaction-	
Marshaller Class Reference	1587
6.298.1 Detailed Description	1588
6.298.2 Constructor & Destructor Documentation	1588
6.298.2.1 JournalTransactionMarshaller	1588
6.298.2.2 ~JournalTransactionMarshaller	1588
6.298.3 Member Function Documentation	1588
6.298.3.1 createObject	1588
6.298.3.2 getDataStructureType	1588
6.298.3.3 looseMarshal	1589
6.298.3.4 looseUnmarshal	1589
6.298.3.5 tightMarshal1	1590
6.298.3.6 tightMarshal2	1590
6.298.3.7 tightUnmarshal	1591
6.299activemq::commands::KeepAliveInfo Class Reference	1591
6.299.1 Constructor & Destructor Documentation	1592
6.299.1.1 KeepAliveInfo	1592
6.299.1.2 ~KeepAliveInfo	1592
6.299.2 Member Function Documentation	1592
6.299.2.1 cloneDataStructure	1592
6.299.2.2 copyDataStructure	1592
6.299.2.3 equals	1593
6.299.2.4 getDataStructureType	1593
6.299.2.5 isKeepAliveInfo	1593
6.299.2.6 toString	1593
6.299.2.7 visit	1594
6.299.3 Field Documentation	1594
6.299.3.1 ID_KEEPLIVEINFO	1594
6.300activemq::wireformat::openwire::marshal::generated::KeepAliveInfo-	
Marshaller Class Reference	1594
6.300.1 Detailed Description	1595
6.300.2 Constructor & Destructor Documentation	1595
6.300.2.1 KeepAliveInfoMarshaller	1595
6.300.2.2 ~KeepAliveInfoMarshaller	1595

6.300.3 Member Function Documentation	1595
6.300.3.1 createObject	1595
6.300.3.2 getDataStructureType	1596
6.300.3.3 looseMarshal	1596
6.300.3.4 looseUnmarshal	1596
6.300.3.5 tightMarshal1	1597
6.300.3.6 tightMarshal2	1597
6.300.3.7 tightUnmarshal	1598
6.301decaf::security::Key Class Reference	1598
6.301.1 Detailed Description	1599
6.301.2 Constructor & Destructor Documentation	1599
6.301.2.1 ~Key	1599
6.301.3 Member Function Documentation	1599
6.301.3.1 getAlgorithm	1599
6.301.3.2 getEncoded	1600
6.301.3.3 getFormat	1600
6.302decaf::security::KeyException Class Reference	1600
6.302.1 Constructor & Destructor Documentation	1601
6.302.1.1 KeyException	1601
6.302.1.2 KeyException	1601
6.302.1.3 KeyException	1601
6.302.1.4 KeyException	1602
6.302.1.5 KeyException	1602
6.302.1.6 KeyException	1602
6.302.1.7 ~KeyException	1602
6.302.2 Member Function Documentation	1602
6.302.2.1 clone	1603
6.303decaf::security::KeyManagementException Class Reference	1603
6.303.1 Constructor & Destructor Documentation	1604
6.303.1.1 KeyManagementException	1604
6.303.1.2 KeyManagementException	1604
6.303.1.3 KeyManagementException	1604
6.303.1.4 KeyManagementException	1604
6.303.1.5 KeyManagementException	1604

6.303.1.6 KeyManagementException	1605
6.303.1.7 ~KeyManagementException	1605
6.303.2 Member Function Documentation	1605
6.303.2.1 clone	1605
6.304activemq::commands::LastPartialCommand Class Reference	1606
6.304.1 Constructor & Destructor Documentation	1606
6.304.1.1 LastPartialCommand	1606
6.304.1.2 ~LastPartialCommand	1606
6.304.2 Member Function Documentation	1606
6.304.2.1 cloneDataStructure	1606
6.304.2.2 copyDataStructure	1607
6.304.2.3 equals	1607
6.304.2.4 getDataStructureType	1607
6.304.2.5 toString	1608
6.304.3 Field Documentation	1608
6.304.3.1 ID_LASTPARTIALCOMMAND	1608
6.305activemq::wireformat::openwire::marshal::generated::LastPartial- CommandMarshaller Class Reference	1608
6.305.1 Detailed Description	1609
6.305.2 Constructor & Destructor Documentation	1609
6.305.2.1 LastPartialCommandMarshaller	1609
6.305.2.2 ~LastPartialCommandMarshaller	1609
6.305.3 Member Function Documentation	1609
6.305.3.1 createObject	1609
6.305.3.2 getDataStructureType	1610
6.305.3.3 looseMarshal	1610
6.305.3.4 looseUnmarshal	1610
6.305.3.5 tightMarshal1	1611
6.305.3.6 tightMarshal2	1611
6.305.3.7 tightUnmarshal	1612
6.306decaf::util::comparators::Less< E > Class Template Reference	1612
6.306.1 Detailed Description	1613
6.306.2 Constructor & Destructor Documentation	1613
6.306.2.1 Less	1613

6.306.2.2 ~Less	1613
6.306.3 Member Function Documentation	1613
6.306.3.1 compare	1613
6.306.3.2 operator()	1614
6.307std::less< decaf::lang::ArrayPointer< T > > Struct Template Reference	1614
6.307.1 Detailed Description	1614
6.307.2 Member Function Documentation	1615
6.307.2.1 operator()	1615
6.308std::less< decaf::lang::Pointer< T > > Struct Template Reference . . .	1615
6.308.1 Detailed Description	1615
6.308.2 Member Function Documentation	1615
6.308.2.1 operator()	1615
6.309decaf::util::logging::Level Class Reference	1616
6.309.1 Detailed Description	1617
6.309.2 Constructor & Destructor Documentation	1617
6.309.2.1 Level	1617
6.309.2.2 ~Level	1618
6.309.3 Member Function Documentation	1618
6.309.3.1 compareTo	1618
6.309.3.2 equals	1618
6.309.3.3 getName	1618
6.309.3.4 intValue	1618
6.309.3.5 operator<	1618
6.309.3.6 operator==	1618
6.309.3.7 parse	1618
6.309.3.8 toString	1619
6.309.4 Field Documentation	1619
6.309.4.1 ALL	1619
6.309.4.2 CONFIG	1619
6.309.4.3 DEBUG	1619
6.309.4.4 FINE	1619
6.309.4.5 FINER	1620
6.309.4.6 FINEST	1620
6.309.4.7 INFO	1620

6.309.4.8 INHERIT	1620
6.309.4.9 OFF	1620
6.309.4.10SEVERE	1620
6.309.4.11WARNING	1620
6.310decaf::util::concurrent::LinkedBlockingQueue< E > Class Template - Reference	1621
6.310.1 Detailed Description	1623
6.310.2 Constructor & Destructor Documentation	1623
6.310.2.1 LinkedBlockingQueue	1623
6.310.2.2 LinkedBlockingQueue	1624
6.310.2.3 LinkedBlockingQueue	1624
6.310.2.4 ~LinkedBlockingQueue	1624
6.310.3 Member Function Documentation	1624
6.310.3.1 clear	1624
6.310.3.2 drainTo	1625
6.310.3.3 drainTo	1626
6.310.3.4 iterator	1626
6.310.3.5 iterator	1627
6.310.3.6 offer	1627
6.310.3.7 offer	1627
6.310.3.8 operator=	1628
6.310.3.9 operator=	1628
6.310.3.10peek	1628
6.310.3.11poll	1629
6.310.3.12poll	1629
6.310.3.13put	1630
6.310.3.14remainingCapacity	1630
6.310.3.15remove	1631
6.310.3.16size	1632
6.310.3.17take	1632
6.310.3.18toArray	1632
6.310.3.19toString	1633
6.311decaf::util::LinkedList< E > Class Template Reference	1633
6.311.1 Detailed Description	1639

6.311.2 Constructor & Destructor Documentation	1639
6.311.2.1 LinkedList	1639
6.311.2.2 LinkedList	1639
6.311.2.3 LinkedList	1639
6.311.2.4 ~LinkedList	1639
6.311.3 Member Function Documentation	1639
6.311.3.1 add	1639
6.311.3.2 add	1640
6.311.3.3 addAll	1641
6.311.3.4 addAll	1642
6.311.3.5 addFirst	1643
6.311.3.6 addLast	1643
6.311.3.7 clear	1644
6.311.3.8 contains	1644
6.311.3.9 copy	1645
6.311.3.10 descendingIterator	1645
6.311.3.11 descendingIterator	1645
6.311.3.12 element	1645
6.311.3.13 get	1646
6.311.3.14 getFirst	1646
6.311.3.15 getFirst	1647
6.311.3.16 getLast	1647
6.311.3.17 getLast	1647
6.311.3.18 indexOf	1647
6.311.3.19 isEmpty	1648
6.311.3.20 lastIndexOf	1648
6.311.3.21 listIterator	1649
6.311.3.22 listIterator	1649
6.311.3.23 offer	1649
6.311.3.24 offerFirst	1650
6.311.3.25 offerLast	1650
6.311.3.26 operator=	1651
6.311.3.27 operator=	1651
6.311.3.28 peek	1651

6.311.3.29	peekFirst	1651
6.311.3.30	peekLast	1652
6.311.3.31	poll	1652
6.311.3.32	pollFirst	1652
6.311.3.33	pollLast	1653
6.311.3.34	pop	1653
6.311.3.35	push	1654
6.311.3.36	remove	1654
6.311.3.37	remove	1655
6.311.3.38	removeFirst	1655
6.311.3.39	removeFirstOccurrence	1656
6.311.3.40	removeLast	1656
6.311.3.41	removeLastOccurrence	1657
6.311.3.42	set	1657
6.311.3.43	size	1658
6.311.3.44	toArray	1658
6.312	decaf::util::List< E > Class Template Reference	1658
6.312.1	Detailed Description	1659
6.312.2	Constructor & Destructor Documentation	1660
6.312.2.1	List	1660
6.312.2.2	~List	1660
6.312.3	Member Function Documentation	1660
6.312.3.1	add	1660
6.312.3.2	addAll	1661
6.312.3.3	get	1662
6.312.3.4	indexOf	1663
6.312.3.5	lastIndexOf	1664
6.312.3.6	listIterator	1665
6.312.3.7	listIterator	1666
6.312.3.8	listIterator	1666
6.312.3.9	listIterator	1668
6.312.3.10	removeAt	1668
6.312.3.11	set	1669
6.313	decaf::util::ListIterator< E > Class Template Reference	1671

6.313.1 Detailed Description	1671
6.313.2 Constructor & Destructor Documentation	1672
6.313.2.1 ~ListIterator	1672
6.313.3 Member Function Documentation	1672
6.313.3.1 add	1672
6.313.3.2 hasPrevious	1672
6.313.3.3 nextIndex	1673
6.313.3.4 previous	1673
6.313.3.5 previousIndex	1673
6.313.3.6 set	1674
6.314activemq::commands::LocalTransactionId Class Reference	1674
6.314.1 Member Typedef Documentation	1675
6.314.1.1 COMPARATOR	1675
6.314.2 Constructor & Destructor Documentation	1676
6.314.2.1 LocalTransactionId	1676
6.314.2.2 LocalTransactionId	1676
6.314.2.3 ~LocalTransactionId	1676
6.314.3 Member Function Documentation	1676
6.314.3.1 cloneDataStructure	1676
6.314.3.2 compareTo	1676
6.314.3.3 copyDataStructure	1676
6.314.3.4 equals	1676
6.314.3.5 equals	1677
6.314.3.6 getConnectionId	1677
6.314.3.7 getConnectionId	1677
6.314.3.8 getDataStructureType	1677
6.314.3.9 getValue	1677
6.314.3.10sLocalTransactionId	1677
6.314.3.11operator<	1677
6.314.3.12operator=	1677
6.314.3.13operator==	1677
6.314.3.14setConnectionId	1677
6.314.3.15setValue	1678
6.314.3.16toString	1678

6.314.4 Field Documentation	1678
6.314.4.1 connectionId	1678
6.314.4.2 ID_LOCALTRANSACTIONID	1678
6.314.4.3 value	1678
6.315activemq::wireformat::openwire::marshal::generated::LocalTransaction-	
IdMarshaller Class Reference	1678
6.315.1 Detailed Description	1679
6.315.2 Constructor & Destructor Documentation	1679
6.315.2.1 LocalTransactionIdMarshaller	1679
6.315.2.2 ~LocalTransactionIdMarshaller	1679
6.315.3 Member Function Documentation	1679
6.315.3.1 createObject	1679
6.315.3.2 getDataStructureType	1680
6.315.3.3 looseMarshal	1680
6.315.3.4 looseUnmarshal	1680
6.315.3.5 tightMarshal1	1681
6.315.3.6 tightMarshal2	1681
6.315.3.7 tightUnmarshal	1682
6.316decaf::util::concurrent::Lock Class Reference	1682
6.316.1 Detailed Description	1683
6.316.2 Constructor & Destructor Documentation	1683
6.316.2.1 Lock	1683
6.316.2.2 ~Lock	1683
6.316.3 Member Function Documentation	1683
6.316.3.1 isLocked	1683
6.316.3.2 lock	1683
6.316.3.3 unlock	1684
6.317decaf::util::concurrent::locks::Lock Class Reference	1684
6.317.1 Detailed Description	1684
6.317.2 Constructor & Destructor Documentation	1686
6.317.2.1 ~Lock	1686
6.317.3 Member Function Documentation	1686
6.317.3.1 lock	1686
6.317.3.2 lockInterruptibly	1686

6.317.3.3 newCondition	1687
6.317.3.4 tryLock	1687
6.317.3.5 tryLock	1688
6.317.3.6 unlock	1689
6.318decaf::util::concurrent::locks::LockSupport Class Reference	1690
6.318.1 Detailed Description	1690
6.318.2 Constructor & Destructor Documentation	1691
6.318.2.1 ~LockSupport	1691
6.318.3 Member Function Documentation	1691
6.318.3.1 park	1691
6.318.3.2 parkNanos	1692
6.318.3.3 parkUntil	1692
6.318.3.4 unpark	1693
6.319decaf::util::logging::Logger Class Reference	1693
6.319.1 Detailed Description	1695
6.319.2 Constructor & Destructor Documentation	1696
6.319.2.1 Logger	1696
6.319.2.2 ~Logger	1696
6.319.3 Member Function Documentation	1696
6.319.3.1 addHandler	1696
6.319.3.2 config	1697
6.319.3.3 debug	1697
6.319.3.4 entering	1698
6.319.3.5 exiting	1698
6.319.3.6 fine	1698
6.319.3.7 finer	1699
6.319.3.8 finest	1699
6.319.3.9 getAnonymousLogger	1699
6.319.3.10getFilter	1700
6.319.3.11getHandlers	1700
6.319.3.12getLevel	1700
6.319.3.13getLogger	1700
6.319.3.14getName	1701
6.319.3.15getParent	1701

6.319.3.16	getUseParentHandlers	1701
6.319.3.17	info	1701
6.319.3.18	isLoggable	1702
6.319.3.19	log	1702
6.319.3.20	log	1702
6.319.3.21	log	1702
6.319.3.22	log	1703
6.319.3.23	removeHandler	1703
6.319.3.24	setFilter	1703
6.319.3.25	setLevel	1704
6.319.3.26	setParent	1704
6.319.3.27	setUseParentHandlers	1704
6.319.3.28	severe	1704
6.319.3.29	throwing	1705
6.319.3.30	warning	1705
6.320	decaf::util::logging::LoggerHierarchy Class Reference	1706
6.320.1	Constructor & Destructor Documentation	1706
6.320.1.1	LoggerHierarchy	1706
6.320.1.2	~LoggerHierarchy	1706
6.321	activemq::io::LoggingInputStream Class Reference	1706
6.321.1	Constructor & Destructor Documentation	1707
6.321.1.1	LoggingInputStream	1707
6.321.1.2	~LoggingInputStream	1707
6.321.2	Member Function Documentation	1707
6.321.2.1	doReadArrayBounded	1707
6.321.2.2	doReadByte	1707
6.322	activemq::io::LoggingOutputStream Class Reference	1707
6.322.1	Detailed Description	1708
6.322.2	Constructor & Destructor Documentation	1708
6.322.2.1	LoggingOutputStream	1708
6.322.2.2	~LoggingOutputStream	1708
6.322.3	Member Function Documentation	1708
6.322.3.1	doWriteArrayBounded	1708
6.322.3.2	doWriteByte	1708

6.323activemq::transport::logging::LoggingTransport Class Reference	1709
6.323.1 Detailed Description	1710
6.323.2 Constructor & Destructor Documentation	1710
6.323.2.1 LoggingTransport	1710
6.323.2.2 ~LoggingTransport	1710
6.323.3 Member Function Documentation	1710
6.323.3.1 onCommand	1710
6.323.3.2 oneway	1710
6.323.3.3 request	1711
6.323.3.4 request	1711
6.324decaf::util::logging::LogManager Class Reference	1712
6.324.1 Detailed Description	1713
6.324.2 Constructor & Destructor Documentation	1714
6.324.2.1 ~LogManager	1715
6.324.2.2 LogManager	1715
6.324.2.3 LogManager	1715
6.324.3 Member Function Documentation	1715
6.324.3.1 addLogger	1715
6.324.3.2 addPropertyChangeListener	1715
6.324.3.3 getLogger	1716
6.324.3.4 getLoggerNames	1716
6.324.3.5 getLogManager	1716
6.324.3.6 getProperties	1716
6.324.3.7 getProperty	1717
6.324.3.8 operator=	1717
6.324.3.9 readConfiguration	1717
6.324.3.10readConfiguration	1717
6.324.3.11removePropertyChangeListener	1718
6.324.3.12reset	1718
6.324.3.13setProperties	1718
6.324.4 Friends And Related Function Documentation	1718
6.324.4.1 decaf::lang::Runtime	1718
6.325decaf::util::logging::LogRecord Class Reference	1719
6.325.1 Detailed Description	1720

6.325.2 Constructor & Destructor Documentation	1720
6.325.2.1 LogRecord	1720
6.325.2.2 ~LogRecord	1720
6.325.3 Member Function Documentation	1720
6.325.3.1 getLevel	1720
6.325.3.2 getLoggerName	1720
6.325.3.3 getMessage	1721
6.325.3.4 getSourceFile	1721
6.325.3.5 getSourceFunction	1721
6.325.3.6 getSourceLine	1721
6.325.3.7 getThrown	1721
6.325.3.8 getTimestamp	1722
6.325.3.9 getTreadId	1722
6.325.3.10 setLevel	1722
6.325.3.11 setLoggerName	1722
6.325.3.12 setMessage	1722
6.325.3.13 setSourceFile	1723
6.325.3.14 setSourceFunction	1723
6.325.3.15 setSourceLine	1723
6.325.3.16 setThrown	1723
6.325.3.17 setTimestamp	1723
6.325.3.18 setTreadId	1724
6.326 decaf::util::logging::LogWriter Class Reference	1724
6.326.1 Constructor & Destructor Documentation	1725
6.326.1.1 LogWriter	1725
6.326.1.2 ~LogWriter	1725
6.326.2 Member Function Documentation	1725
6.326.2.1 destroy	1725
6.326.2.2 getInstance	1725
6.326.2.3 log	1725
6.326.2.4 log	1725
6.326.2.5 returnInstance	1725
6.327 decaf::lang::Long Class Reference	1726
6.327.1 Constructor & Destructor Documentation	1728

6.327.1.1 Long	1728
6.327.1.2 Long	1728
6.327.1.3 ~Long	1729
6.327.2 Member Function Documentation	1729
6.327.2.1 bitCount	1729
6.327.2.2 byteValue	1729
6.327.2.3 compareTo	1729
6.327.2.4 compareTo	1730
6.327.2.5 decode	1730
6.327.2.6 doubleValue	1731
6.327.2.7 equals	1731
6.327.2.8 equals	1731
6.327.2.9 floatValue	1731
6.327.2.10highestOneBit	1732
6.327.2.11intValue	1732
6.327.2.12longValue	1732
6.327.2.13lowestOneBit	1732
6.327.2.14numberOfLeadingZeros	1733
6.327.2.15numberOfTrailingZeros	1733
6.327.2.16operator<	1734
6.327.2.17operator<	1734
6.327.2.18operator==	1734
6.327.2.19operator==	1735
6.327.2.20parseLong	1735
6.327.2.21parseLong	1736
6.327.2.22reverse	1736
6.327.2.23reverseBytes	1736
6.327.2.24rotateLeft	1737
6.327.2.25rotateRight	1737
6.327.2.26shortValue	1738
6.327.2.27signum	1738
6.327.2.28toBinaryString	1738
6.327.2.29toHexString	1739
6.327.2.30toOctalString	1739

6.327.2.31toString	1739
6.327.2.32toString	1740
6.327.2.33toString	1740
6.327.2.34valueOf	1740
6.327.2.35valueOf	1740
6.327.2.36valueOf	1741
6.327.3 Field Documentation	1741
6.327.3.1 MAX_VALUE	1741
6.327.3.2 MIN_VALUE	1741
6.327.3.3 SIZE	1742
6.328decaf::internal::nio::LongArrayBuffer Class Reference	1742
6.328.1 Constructor & Destructor Documentation	1745
6.328.1.1 LongArrayBuffer	1745
6.328.1.2 LongArrayBuffer	1746
6.328.1.3 LongArrayBuffer	1746
6.328.1.4 LongArrayBuffer	1747
6.328.1.5 ~LongArrayBuffer	1747
6.328.2 Member Function Documentation	1747
6.328.2.1 array	1747
6.328.2.2 arrayOffset	1748
6.328.2.3 asReadOnlyBuffer	1748
6.328.2.4 compact	1748
6.328.2.5 duplicate	1749
6.328.2.6 get	1749
6.328.2.7 get	1750
6.328.2.8 hasArray	1750
6.328.2.9 isReadOnly	1750
6.328.2.10put	1751
6.328.2.11put	1751
6.328.2.12setReadOnly	1752
6.328.2.13slice	1752
6.329decaf::nio::LongBuffer Class Reference	1752
6.329.1 Detailed Description	1754
6.329.2 Constructor & Destructor Documentation	1755

6.329.2.1 LongBuffer	1755
6.329.2.2 ~LongBuffer	1755
6.329.3 Member Function Documentation	1755
6.329.3.1 allocate	1755
6.329.3.2 array	1755
6.329.3.3 arrayOffset	1756
6.329.3.4 asReadOnlyBuffer	1756
6.329.3.5 compact	1757
6.329.3.6 compareTo	1757
6.329.3.7 duplicate	1757
6.329.3.8 equals	1758
6.329.3.9 get	1758
6.329.3.10get	1758
6.329.3.11get	1759
6.329.3.12get	1759
6.329.3.13hasArray	1760
6.329.3.14operator<	1760
6.329.3.15operator==	1760
6.329.3.16put	1760
6.329.3.17put	1761
6.329.3.18put	1762
6.329.3.19put	1762
6.329.3.20put	1763
6.329.3.21slice	1763
6.329.3.22toString	1763
6.329.3.23wrap	1764
6.329.3.24wrap	1764
6.330activemq::util::LongSequenceGenerator Class Reference	1765
6.330.1 Detailed Description	1765
6.330.2 Constructor & Destructor Documentation	1765
6.330.2.1 LongSequenceGenerator	1765
6.330.2.2 ~LongSequenceGenerator	1765
6.330.3 Member Function Documentation	1765
6.330.3.1 getLastSequenceId	1765

6.330.3.2 getNextSequenced	1766
6.331decaf::net::MalformedURLException Class Reference	1766
6.331.1 Constructor & Destructor Documentation	1766
6.331.1.1 MalformedURLException	1766
6.331.1.2 MalformedURLException	1767
6.331.1.3 MalformedURLException	1767
6.331.1.4 MalformedURLException	1767
6.331.1.5 MalformedURLException	1767
6.331.1.6 MalformedURLException	1768
6.331.1.7 ~MalformedURLException	1768
6.331.2 Member Function Documentation	1768
6.331.2.1 clone	1768
6.332decaf::util::Map< K, V, COMPARATOR > Class Template Reference	1768
6.332.1 Detailed Description	1769
6.332.2 Constructor & Destructor Documentation	1770
6.332.2.1 Map	1770
6.332.2.2 ~Map	1770
6.332.3 Member Function Documentation	1770
6.332.3.1 clear	1770
6.332.3.2 containsKey	1771
6.332.3.3 containsValue	1771
6.332.3.4 copy	1772
6.332.3.5 equals	1772
6.332.3.6 get	1773
6.332.3.7 get	1774
6.332.3.8 isEmpty	1774
6.332.3.9 keySet	1775
6.332.3.10put	1776
6.332.3.11putAll	1777
6.332.3.12remove	1777
6.332.3.13size	1778
6.332.3.14values	1779
6.333cms::MapMessage Class Reference	1779
6.333.1 Detailed Description	1781

6.333.2 Constructor & Destructor Documentation	1782
6.333.2.1 ~MapMessage	1782
6.333.3 Member Function Documentation	1782
6.333.3.1 getBoolean	1782
6.333.3.2 getByte	1782
6.333.3.3 getBytes	1783
6.333.3.4 getChar	1783
6.333.3.5 getDouble	1783
6.333.3.6 getFloat	1784
6.333.3.7 getInt	1784
6.333.3.8 getLong	1785
6.333.3.9 getMapNames	1785
6.333.3.10getShort	1785
6.333.3.11getString	1786
6.333.3.12sEmpty	1786
6.333.3.13itemExists	1787
6.333.3.14setBoolean	1787
6.333.3.15setByte	1787
6.333.3.16setBytes	1788
6.333.3.17setChar	1788
6.333.3.18setDouble	1789
6.333.3.19setFloat	1789
6.333.3.20setInt	1789
6.333.3.21setLong	1790
6.333.3.22setShort	1790
6.333.3.23setString	1791
6.334decaf::util::logging::MarkBlockLogger Class Reference	1791
6.334.1 Detailed Description	1792
6.334.2 Constructor & Destructor Documentation	1792
6.334.2.1 MarkBlockLogger	1792
6.334.2.2 ~MarkBlockLogger	1792
6.335activemq::wireformat::MarshalAware Class Reference	1792
6.335.1 Constructor & Destructor Documentation	1793
6.335.1.1 ~MarshalAware	1793

6.335.2 Member Function Documentation	1793
6.335.2.1 afterMarshal	1793
6.335.2.2 afterUnmarshal	1793
6.335.2.3 beforeMarshal	1794
6.335.2.4 beforeUnmarshal	1794
6.335.2.5 getMarshaledForm	1794
6.335.2.6 isMarshalAware	1795
6.335.2.7 setMarshaledForm	1795
6.336activemq::wireformat::openwire::marshal::generated::MarshallerFactory	
Class Reference	1795
6.336.1 Detailed Description	1796
6.336.2 Constructor & Destructor Documentation	1796
6.336.2.1 ~MarshallerFactory	1796
6.336.3 Member Function Documentation	1796
6.336.3.1 configure	1796
6.337activemq::util::MarshallingSupport Class Reference	1796
6.337.1 Constructor & Destructor Documentation	1797
6.337.1.1 MarshallingSupport	1797
6.337.1.2 ~MarshallingSupport	1797
6.337.2 Member Function Documentation	1797
6.337.2.1 asciiToModifiedUtf8	1797
6.337.2.2 modifiedUtf8ToAscii	1798
6.337.2.3 readString16	1798
6.337.2.4 readString32	1799
6.337.2.5 writeString	1799
6.337.2.6 writeString16	1800
6.337.2.7 writeString32	1800
6.338decaf::lang::Math Class Reference	1800
6.338.1 Detailed Description	1802
6.338.2 Constructor & Destructor Documentation	1802
6.338.2.1 Math	1802
6.338.2.2 ~Math	1802
6.338.3 Member Function Documentation	1802
6.338.3.1 abs	1802

6.338.3.2 abs	1803
6.338.3.3 abs	1803
6.338.3.4 abs	1803
6.338.3.5 ceil	1804
6.338.3.6 floor	1805
6.338.3.7 max	1806
6.338.3.8 max	1806
6.338.3.9 max	1806
6.338.3.10max	1807
6.338.3.11max	1807
6.338.3.12min	1808
6.338.3.13min	1808
6.338.3.14min	1809
6.338.3.15min	1809
6.338.3.16min	1809
6.338.3.17min	1810
6.338.3.18pow	1810
6.338.3.19random	1810
6.338.3.20round	1811
6.338.3.21round	1812
6.338.3.22signum	1812
6.338.3.23signum	1813
6.338.3.24sqrt	1814
6.338.3.25toDegrees	1817
6.338.3.26toRadians	1817
6.338.4 Field Documentation	1818
6.338.4.1 E	1818
6.338.4.2 PI	1818
6.339activemq::util::MemoryUsage Class Reference	1818
6.339.1 Constructor & Destructor Documentation	1819
6.339.1.1 MemoryUsage	1819
6.339.1.2 MemoryUsage	1819
6.339.1.3 ~MemoryUsage	1819
6.339.2 Member Function Documentation	1819

6.339.2.1 decreaseUsage	1819
6.339.2.2 enqueueUsage	1819
6.339.2.3 getLimit	1820
6.339.2.4 getUsage	1820
6.339.2.5 increaseUsage	1820
6.339.2.6 isFull	1820
6.339.2.7 setLimit	1820
6.339.2.8 setUsage	1821
6.339.2.9 waitForSpace	1821
6.339.2.10waitForSpace	1821
6.340activemq::commands::Message Class Reference	1821
6.340.1 Constructor & Destructor Documentation	1826
6.340.1.1 Message	1826
6.340.1.2 ~Message	1826
6.340.2 Member Function Documentation	1826
6.340.2.1 afterUnmarshal	1826
6.340.2.2 beforeMarshal	1826
6.340.2.3 cloneDataStructure	1826
6.340.2.4 copyDataStructure	1827
6.340.2.5 equals	1827
6.340.2.6 getAckHandler	1828
6.340.2.7 getArrival	1828
6.340.2.8 getBrokerInTime	1828
6.340.2.9 getBrokerOutTime	1828
6.340.2.10getBrokerPath	1828
6.340.2.11getBrokerPath	1828
6.340.2.12getCluster	1828
6.340.2.13getCluster	1828
6.340.2.14getConnection	1828
6.340.2.15getContent	1829
6.340.2.16getContent	1829
6.340.2.17getCorrelationId	1829
6.340.2.18getCorrelationId	1829
6.340.2.19getDataStructure	1829

6.340.2.20	getDataStructure	1829
6.340.2.21	getDataStructureType	1829
6.340.2.22	getDestination	1829
6.340.2.23	getDestination	1829
6.340.2.24	getExpiration	1829
6.340.2.25	getGroupID	1830
6.340.2.26	getGroupID	1830
6.340.2.27	getGroupSequence	1830
6.340.2.28	getMarshaledProperties	1830
6.340.2.29	getMarshaledProperties	1830
6.340.2.30	getMessageId	1830
6.340.2.31	getMessageId	1830
6.340.2.32	getMessageProperties	1830
6.340.2.33	getMessageProperties	1830
6.340.2.34	getOriginalDestination	1830
6.340.2.35	getOriginalDestination	1830
6.340.2.36	getOriginalTransactionId	1830
6.340.2.37	getOriginalTransactionId	1831
6.340.2.38	getPriority	1831
6.340.2.39	getProducerId	1831
6.340.2.40	getProducerId	1831
6.340.2.41	getRedeliveryCounter	1831
6.340.2.42	getReplyTo	1831
6.340.2.43	getReplyTo	1831
6.340.2.44	getSize	1831
6.340.2.45	getTargetConsumerId	1831
6.340.2.46	getTargetConsumerId	1831
6.340.2.47	getTimestamp	1831
6.340.2.48	getTransactionId	1831
6.340.2.49	getTransactionId	1832
6.340.2.50	getType	1832
6.340.2.51	getType	1832
6.340.2.52	getUserId	1832
6.340.2.53	getUserId	1832

6.340.2.54	isCompressed	1832
6.340.2.55	isDroppable	1832
6.340.2.56	isExpired	1832
6.340.2.57	isMarshalAware	1832
6.340.2.58	isMessage	1832
6.340.2.59	isPersistent	1833
6.340.2.60	isReadOnlyBody	1833
6.340.2.61	isReadOnlyProperties	1833
6.340.2.62	isRecievedByDFBridge	1833
6.340.2.63	onSend	1833
6.340.2.64	setAckHandler	1834
6.340.2.65	setArrival	1834
6.340.2.66	setBrokerInTime	1834
6.340.2.67	setBrokerOutTime	1834
6.340.2.68	setBrokerPath	1834
6.340.2.69	setCluster	1834
6.340.2.70	setCompressed	1834
6.340.2.71	setConnection	1834
6.340.2.72	setContent	1834
6.340.2.73	setCorrelationId	1834
6.340.2.74	setDataStructure	1835
6.340.2.75	setDestination	1835
6.340.2.76	setDroppable	1835
6.340.2.77	setExpiration	1835
6.340.2.78	setGroupId	1835
6.340.2.79	setGroupSequence	1835
6.340.2.80	setMarshaledProperties	1835
6.340.2.81	setMessageId	1835
6.340.2.82	setOriginalDestination	1835
6.340.2.83	setOriginalTransactionId	1835
6.340.2.84	setPersistent	1835
6.340.2.85	setPriority	1835
6.340.2.86	setProducerId	1835
6.340.2.87	setReadOnlyBody	1835

6.340.2.88	setReadOnlyProperties	1836
6.340.2.89	setRecievedByDFBridge	1836
6.340.2.90	setRedeliveryCounter	1836
6.340.2.91	setReplyTo	1836
6.340.2.92	setTargetConsumerId	1836
6.340.2.93	setTimestamp	1836
6.340.2.94	setTransactionId	1836
6.340.2.95	setType	1836
6.340.2.96	setUserId	1836
6.340.2.97	toString	1836
6.340.2.98	visit	1837
6.340.3	Field Documentation	1837
6.340.3.1	arrival	1837
6.340.3.2	brokerInTime	1837
6.340.3.3	brokerOutTime	1837
6.340.3.4	brokerPath	1837
6.340.3.5	cluster	1837
6.340.3.6	compressed	1837
6.340.3.7	connection	1837
6.340.3.8	content	1837
6.340.3.9	correlationId	1837
6.340.3.10	dataStructure	1837
6.340.3.11	DEFAULT_MESSAGE_SIZE	1838
6.340.3.12	destination	1838
6.340.3.13	droppable	1838
6.340.3.14	expiration	1838
6.340.3.15	groupId	1838
6.340.3.16	groupSequence	1838
6.340.3.17	ID_MESSAGE	1838
6.340.3.18	marshalledProperties	1838
6.340.3.19	messageId	1838
6.340.3.20	originalDestination	1838
6.340.3.21	originalTransactionId	1838
6.340.3.22	persistent	1838

6.340.3.23	priority	1838
6.340.3.24	producerId	1838
6.340.3.25	recievedByDFBridge	1838
6.340.3.26	redeliveryCounter	1838
6.340.3.27	replyTo	1838
6.340.3.28	targetConsumerId	1838
6.340.3.29	timestamp	1839
6.340.3.30	transactionId	1839
6.340.3.31	type	1839
6.340.3.32	userId	1839
6.341	cms::Message Class Reference	1839
6.341.1	Detailed Description	1841
6.341.2	Constructor & Destructor Documentation	1843
6.341.2.1	~Message	1843
6.341.3	Member Function Documentation	1843
6.341.3.1	acknowledge	1843
6.341.3.2	clearBody	1843
6.341.3.3	clearProperties	1844
6.341.3.4	clone	1844
6.341.3.5	getBooleanProperty	1845
6.341.3.6	getByteProperty	1845
6.341.3.7	getCMSCorrelationID	1846
6.341.3.8	getCMSDeliveryMode	1846
6.341.3.9	getCMSDestination	1847
6.341.3.10	getCMSExpiration	1847
6.341.3.11	getCMSMessageID	1848
6.341.3.12	getCMSPriority	1849
6.341.3.13	getCMSRedelivered	1850
6.341.3.14	getCMSReplyTo	1850
6.341.3.15	getCMSTimestamp	1851
6.341.3.16	getCMSType	1851
6.341.3.17	getDoubleProperty	1852
6.341.3.18	getFloatProperty	1853
6.341.3.19	getIntProperty	1853

6.341.3.20	getLongProperty	1854
6.341.3.21	getPropertyNames	1854
6.341.3.22	getShortProperty	1855
6.341.3.23	getStringProperty	1855
6.341.3.24	propertyExists	1856
6.341.3.25	setBooleanProperty	1857
6.341.3.26	setByteProperty	1857
6.341.3.27	setCMSCorrelationID	1858
6.341.3.28	setCMSDeliveryMode	1859
6.341.3.29	setCMSDestination	1859
6.341.3.30	setCMSExpiration	1860
6.341.3.31	setCMSMessageID	1860
6.341.3.32	setCMSPriority	1861
6.341.3.33	setCMSRedelivered	1861
6.341.3.34	setCMSReplyTo	1862
6.341.3.35	setCMSTimestamp	1862
6.341.3.36	setCMSType	1863
6.341.3.37	setDoubleProperty	1864
6.341.3.38	setFloatProperty	1864
6.341.3.39	setIntProperty	1865
6.341.3.40	setLongProperty	1865
6.341.3.41	setShortProperty	1866
6.341.3.42	setStringProperty	1866
6.341.4	Field Documentation	1867
6.341.4.1	DEFAULT_DELIVERY_MODE	1867
6.341.4.2	DEFAULT_MSG_PRIORITY	1867
6.341.4.3	DEFAULT_TIME_TO_LIVE	1867
6.342	activemq::commands::MessageAck Class Reference	1867
6.342.1	Constructor & Destructor Documentation	1869
6.342.1.1	MessageAck	1869
6.342.1.2	~MessageAck	1869
6.342.2	Member Function Documentation	1869
6.342.2.1	cloneDataStructure	1869
6.342.2.2	copyDataStructure	1869

6.342.2.3 equals	1869
6.342.2.4 getAckType	1870
6.342.2.5 getConsumerId	1870
6.342.2.6 getConsumerId	1870
6.342.2.7 getDataStructureType	1870
6.342.2.8 getDestination	1870
6.342.2.9 getDestination	1870
6.342.2.10getFirstMessageld	1870
6.342.2.11getFirstMessageld	1870
6.342.2.12getLastMessageld	1870
6.342.2.13getLastMessageld	1871
6.342.2.14getMessageCount	1871
6.342.2.15getTransactionId	1871
6.342.2.16getTransactionId	1871
6.342.2.17sMessageAck	1871
6.342.2.18setAckType	1871
6.342.2.19setConsumerId	1871
6.342.2.20setDestination	1871
6.342.2.21setFirstMessageld	1871
6.342.2.22setLastMessageld	1871
6.342.2.23setMessageCount	1871
6.342.2.24setTransactionId	1871
6.342.2.25toString	1871
6.342.2.26visit	1872
6.342.3 Field Documentation	1872
6.342.3.1 ackType	1872
6.342.3.2 consumerId	1872
6.342.3.3 destination	1872
6.342.3.4 firstMessageld	1872
6.342.3.5 ID_MESSAGEACK	1872
6.342.3.6 lastMessageld	1872
6.342.3.7 messageCount	1872
6.342.3.8 transactionId	1872

6.343activemq::wireformat::openwire::marshal::generated::MessageAck-	
Marshaller Class Reference	1873
6.343.1 Detailed Description	1873
6.343.2 Constructor & Destructor Documentation	1874
6.343.2.1 MessageAckMarshaller	1874
6.343.2.2 ~MessageAckMarshaller	1874
6.343.3 Member Function Documentation	1874
6.343.3.1 createObject	1874
6.343.3.2 getDataStructureType	1874
6.343.3.3 looseMarshal	1874
6.343.3.4 looseUnmarshal	1875
6.343.3.5 tightMarshal1	1875
6.343.3.6 tightMarshal2	1876
6.343.3.7 tightUnmarshal	1876
6.344cms::MessageConsumer Class Reference	1877
6.344.1 Detailed Description	1877
6.344.2 Constructor & Destructor Documentation	1878
6.344.2.1 ~MessageConsumer	1878
6.344.3 Member Function Documentation	1878
6.344.3.1 getMessageListener	1878
6.344.3.2 getMessageSelector	1878
6.344.3.3 receive	1879
6.344.3.4 receive	1879
6.344.3.5 receiveNoWait	1880
6.344.3.6 setMessageListener	1880
6.345activemq::cmsutil::MessageCreator Class Reference	1880
6.345.1 Detailed Description	1881
6.345.2 Constructor & Destructor Documentation	1881
6.345.2.1 ~MessageCreator	1881
6.345.3 Member Function Documentation	1881
6.345.3.1 createMessage	1881
6.346activemq::commands::MessageDispatch Class Reference	1881
6.346.1 Constructor & Destructor Documentation	1883
6.346.1.1 MessageDispatch	1883

6.346.1.2	~MessageDispatch	1883
6.346.2	Member Function Documentation	1883
6.346.2.1	cloneDataStructure	1883
6.346.2.2	copyDataStructure	1883
6.346.2.3	equals	1883
6.346.2.4	getConsumerId	1884
6.346.2.5	getConsumerId	1884
6.346.2.6	getDataStructureType	1884
6.346.2.7	getDestination	1884
6.346.2.8	getDestination	1884
6.346.2.9	getMessage	1884
6.346.2.10	getMessage	1884
6.346.2.11	getRedeliveryCounter	1884
6.346.2.12	isMessageDispatch	1884
6.346.2.13	setConsumerId	1884
6.346.2.14	setDestination	1885
6.346.2.15	setMessage	1885
6.346.2.16	setRedeliveryCounter	1885
6.346.2.17	toString	1885
6.346.2.18	visit	1885
6.346.3	Field Documentation	1885
6.346.3.1	consumerId	1885
6.346.3.2	destination	1885
6.346.3.3	ID_MESSAGEDISPATCH	1885
6.346.3.4	message	1886
6.346.3.5	redeliveryCounter	1886
6.347	activemq::core::MessageDispatchChannel Class Reference	1886
6.347.1	Constructor & Destructor Documentation	1887
6.347.1.1	~MessageDispatchChannel	1887
6.347.2	Member Function Documentation	1887
6.347.2.1	clear	1887
6.347.2.2	close	1887
6.347.2.3	dequeue	1887
6.347.2.4	dequeueNoWait	1888

6.347.2.5 enqueue	1888
6.347.2.6 enqueueFirst	1888
6.347.2.7 isClosed	1888
6.347.2.8 isEmpty	1889
6.347.2.9 isRunning	1889
6.347.2.10 peek	1889
6.347.2.11 removeAll	1889
6.347.2.12 size	1890
6.347.2.13 start	1890
6.347.2.14 stop	1890
6.348activemq::wireformat::openwire::marshal::generated::MessageDispatch-	
Marshaller Class Reference	1890
6.348.1 Detailed Description	1891
6.348.2 Constructor & Destructor Documentation	1891
6.348.2.1 MessageDispatchMarshaller	1891
6.348.2.2 ~MessageDispatchMarshaller	1891
6.348.3 Member Function Documentation	1891
6.348.3.1 createObject	1892
6.348.3.2 getDataStructureType	1892
6.348.3.3 looseMarshal	1892
6.348.3.4 looseUnmarshal	1893
6.348.3.5 tightMarshal1	1893
6.348.3.6 tightMarshal2	1893
6.348.3.7 tightUnmarshal	1894
6.349activemq::commands::MessageDispatchNotification Class Reference . .	1894
6.349.1 Constructor & Destructor Documentation	1896
6.349.1.1 MessageDispatchNotification	1896
6.349.1.2 ~MessageDispatchNotification	1896
6.349.2 Member Function Documentation	1896
6.349.2.1 cloneDataStructure	1896
6.349.2.2 copyDataStructure	1896
6.349.2.3 equals	1896
6.349.2.4 getConsumerId	1897
6.349.2.5 getConsumerId	1897

6.349.2.6	getDataStructureType	1897
6.349.2.7	getDeliverySequenceId	1897
6.349.2.8	getDestination	1897
6.349.2.9	getDestination	1897
6.349.2.10	getMessageId	1897
6.349.2.11	getMessageId	1897
6.349.2.12	sendMessageDispatchNotification	1898
6.349.2.13	setConsumerId	1898
6.349.2.14	setDeliverySequenceId	1898
6.349.2.15	setDestination	1898
6.349.2.16	setMessageId	1898
6.349.2.17	toString	1898
6.349.2.18	visit	1898
6.349.3	Field Documentation	1899
6.349.3.1	consumerId	1899
6.349.3.2	deliverySequenceId	1899
6.349.3.3	destination	1899
6.349.3.4	ID_MESSAGE_DISPATCH_NOTIFICATION	1899
6.349.3.5	messageId	1899
6.350	activemq::wireformat::openwire::marshal::generated::MessageDispatch- NotificationMarshaller Class Reference	1899
6.350.1	Detailed Description	1900
6.350.2	Constructor & Destructor Documentation	1900
6.350.2.1	MessageDispatchNotificationMarshaller	1900
6.350.2.2	~MessageDispatchNotificationMarshaller	1900
6.350.3	Member Function Documentation	1900
6.350.3.1	createObject	1901
6.350.3.2	getDataStructureType	1901
6.350.3.3	looseMarshal	1901
6.350.3.4	looseUnmarshal	1902
6.350.3.5	tightMarshal1	1902
6.350.3.6	tightMarshal2	1902
6.350.3.7	tightUnmarshal	1903
6.351	cms::MessageEnumeration Class Reference	1903

6.351.1 Detailed Description	1904
6.351.2 Constructor & Destructor Documentation	1904
6.351.2.1 ~MessageEnumeration	1904
6.351.3 Member Function Documentation	1904
6.351.3.1 hasMoreMessages	1904
6.351.3.2 nextMessage	1905
6.352cms::MessageEOFException Class Reference	1905
6.352.1 Detailed Description	1906
6.352.2 Constructor & Destructor Documentation	1906
6.352.2.1 MessageEOFException	1906
6.352.2.2 MessageEOFException	1906
6.352.2.3 MessageEOFException	1906
6.352.2.4 MessageEOFException	1906
6.352.2.5 MessageEOFException	1906
6.352.2.6 ~MessageEOFException	1906
6.353cms::MessageFormatException Class Reference	1906
6.353.1 Detailed Description	1907
6.353.2 Constructor & Destructor Documentation	1907
6.353.2.1 MessageFormatException	1907
6.353.2.2 MessageFormatException	1907
6.353.2.3 MessageFormatException	1907
6.353.2.4 MessageFormatException	1907
6.353.2.5 MessageFormatException	1907
6.353.2.6 ~MessageFormatException	1907
6.354activemq::commands::MessageId Class Reference	1908
6.354.1 Member Typedef Documentation	1909
6.354.1.1 COMPARATOR	1909
6.354.2 Constructor & Destructor Documentation	1909
6.354.2.1 MessageId	1909
6.354.2.2 MessageId	1909
6.354.2.3 MessageId	1909
6.354.2.4 MessageId	1909
6.354.2.5 MessageId	1909
6.354.2.6 MessageId	1909

6.354.2.7 ~MessageId	1909
6.354.3 Member Function Documentation	1909
6.354.3.1 cloneDataStructure	1910
6.354.3.2 compareTo	1910
6.354.3.3 copyDataStructure	1910
6.354.3.4 equals	1910
6.354.3.5 equals	1910
6.354.3.6 getBrokerSequenceId	1910
6.354.3.7 getDataStructureType	1911
6.354.3.8 getProducerId	1911
6.354.3.9 getProducerId	1911
6.354.3.10getProducerSequenceId	1911
6.354.3.11operator<	1911
6.354.3.12operator=	1911
6.354.3.13operator==	1911
6.354.3.14setBrokerSequenceId	1911
6.354.3.15setProducerId	1911
6.354.3.16setProducerSequenceId	1911
6.354.3.17setTextView	1911
6.354.3.18setValue	1911
6.354.3.19toString	1911
6.354.4 Field Documentation	1912
6.354.4.1 brokerSequenceId	1912
6.354.4.2 ID_MESSAGEID	1912
6.354.4.3 producerId	1912
6.354.4.4 producerSequenceId	1912
6.355activemq::wireformat::openwire::marshal::generated::MessageId- Marshaller Class Reference	1912
6.355.1 Detailed Description	1913
6.355.2 Constructor & Destructor Documentation	1913
6.355.2.1 MessageIdMarshaller	1913
6.355.2.2 ~MessageIdMarshaller	1913
6.355.3 Member Function Documentation	1913
6.355.3.1 createObject	1913

6.355.3.2	getDataStructureType	1914
6.355.3.3	looseMarshal	1914
6.355.3.4	looseUnmarshal	1914
6.355.3.5	tightMarshal1	1915
6.355.3.6	tightMarshal2	1915
6.355.3.7	tightUnmarshal	1916
6.356	cms::MessageListener Class Reference	1916
6.356.1	Detailed Description	1917
6.356.2	Constructor & Destructor Documentation	1917
6.356.2.1	~MessageListener	1917
6.356.3	Member Function Documentation	1917
6.356.3.1	onMessage	1917
6.357	activemq::wireformat::openwire::marshal::generated::MessageMarshaller Class Reference	1917
6.357.1	Detailed Description	1918
6.357.2	Constructor & Destructor Documentation	1918
6.357.2.1	MessageMarshaller	1918
6.357.2.2	~MessageMarshaller	1918
6.357.3	Member Function Documentation	1918
6.357.3.1	looseMarshal	1919
6.357.3.2	looseUnmarshal	1919
6.357.3.3	tightMarshal1	1920
6.357.3.4	tightMarshal2	1921
6.357.3.5	tightUnmarshal	1921
6.358	cms::MessageNotReadableException Class Reference	1922
6.358.1	Detailed Description	1922
6.358.2	Constructor & Destructor Documentation	1923
6.358.2.1	MessageNotReadableException	1923
6.358.2.2	MessageNotReadableException	1923
6.358.2.3	MessageNotReadableException	1923
6.358.2.4	MessageNotReadableException	1923
6.358.2.5	MessageNotReadableException	1923
6.358.2.6	~MessageNotReadableException	1923
6.359	cms::MessageNotWriteableException Class Reference	1923

6.359.1 Detailed Description	1924
6.359.2 Constructor & Destructor Documentation	1924
6.359.2.1 MessageNotWriteableException	1924
6.359.2.2 MessageNotWriteableException	1924
6.359.2.3 MessageNotWriteableException	1924
6.359.2.4 MessageNotWriteableException	1924
6.359.2.5 MessageNotWriteableException	1924
6.359.2.6 ~MessageNotWriteableException	1924
6.360cms::MessageProducer Class Reference	1924
6.360.1 Detailed Description	1926
6.360.2 Constructor & Destructor Documentation	1926
6.360.2.1 ~MessageProducer	1926
6.360.3 Member Function Documentation	1926
6.360.3.1 getDeliveryMode	1926
6.360.3.2 getDisableMessageID	1927
6.360.3.3 getDisableMessageTimeStamp	1927
6.360.3.4 getPriority	1927
6.360.3.5 getTimeToLive	1928
6.360.3.6 send	1928
6.360.3.7 send	1929
6.360.3.8 send	1929
6.360.3.9 send	1930
6.360.3.10setDeliveryMode	1931
6.360.3.11setDisableMessageID	1931
6.360.3.12setDisableMessageTimeStamp	1931
6.360.3.13setPriority	1932
6.360.3.14setTimeToLive	1932
6.361 activemq::wireformat::openwire::utils::MessagePropertyInterceptor - Class Reference	1933
6.361.1 Detailed Description	1934
6.361.2 Constructor & Destructor Documentation	1934
6.361.2.1 MessagePropertyInterceptor	1934
6.361.2.2 ~MessagePropertyInterceptor	1934
6.361.3 Member Function Documentation	1934

6.361.3.1	getBooleanProperty	1935
6.361.3.2	getByteProperty	1935
6.361.3.3	getDoubleProperty	1935
6.361.3.4	getFloatProperty	1936
6.361.3.5	getIntProperty	1936
6.361.3.6	getLongProperty	1936
6.361.3.7	getShortProperty	1937
6.361.3.8	getStringProperty	1937
6.361.3.9	setBooleanProperty	1937
6.361.3.10	setByteProperty	1937
6.361.3.11	setDoubleProperty	1938
6.361.3.12	setFloatProperty	1938
6.361.3.13	setIntProperty	1938
6.361.3.14	setLongProperty	1938
6.361.3.15	setShortProperty	1939
6.361.3.16	setStringProperty	1939
6.362	activemq::commands::MessagePull Class Reference	1939
6.362.1	Constructor & Destructor Documentation	1941
6.362.1.1	MessagePull	1941
6.362.1.2	~MessagePull	1941
6.362.2	Member Function Documentation	1941
6.362.2.1	cloneDataStructure	1941
6.362.2.2	copyDataStructure	1941
6.362.2.3	equals	1941
6.362.2.4	getConsumerId	1942
6.362.2.5	getConsumerId	1942
6.362.2.6	getCorrelationId	1942
6.362.2.7	getCorrelationId	1942
6.362.2.8	getDataStructureType	1942
6.362.2.9	getDestination	1942
6.362.2.10	getDestination	1942
6.362.2.11	getMessageId	1942
6.362.2.12	getMessageId	1942
6.362.2.13	getTimeout	1942

6.362.2.14	setConsumerId	1942
6.362.2.15	setCorrelationId	1942
6.362.2.16	setDestination	1943
6.362.2.17	setMessageId	1943
6.362.2.18	setTimeout	1943
6.362.2.19	toString	1943
6.362.2.20	visit	1943
6.362.3	Field Documentation	1943
6.362.3.1	consumerId	1943
6.362.3.2	correlationId	1943
6.362.3.3	destination	1943
6.362.3.4	ID_MESSAGEPULL	1943
6.362.3.5	messageId	1944
6.362.3.6	timeout	1944
6.363	activemq::wireformat::openwire::marshal::generated::MessagePull- Marshaller Class Reference	1944
6.363.1	Detailed Description	1945
6.363.2	Constructor & Destructor Documentation	1945
6.363.2.1	MessagePullMarshaller	1945
6.363.2.2	~MessagePullMarshaller	1945
6.363.3	Member Function Documentation	1945
6.363.3.1	createObject	1945
6.363.3.2	getDataStructureType	1945
6.363.3.3	looseMarshal	1946
6.363.3.4	looseUnmarshal	1946
6.363.3.5	tightMarshal1	1946
6.363.3.6	tightMarshal2	1947
6.363.3.7	tightUnmarshal	1947
6.364	activemq::transport::mock::MockTransport Class Reference	1948
6.364.1	Detailed Description	1950
6.364.2	Constructor & Destructor Documentation	1950
6.364.2.1	MockTransport	1950
6.364.2.2	~MockTransport	1950
6.364.3	Member Function Documentation	1950

6.364.3.1 close	1951
6.364.3.2 fireCommand	1951
6.364.3.3 fireException	1951
6.364.3.4 getInstance	1951
6.364.3.5 getName	1951
6.364.3.6 getNumReceivedMessageBeforeFail	1951
6.364.3.7 getNumReceivedMessages	1951
6.364.3.8 getNumSentKeepAlives	1952
6.364.3.9 getNumSentKeepAlivesBeforeFail	1952
6.364.3.10getNumSentMessageBeforeFail	1952
6.364.3.11getNumSentMessages	1952
6.364.3.12getRemoteAddress	1952
6.364.3.13getTransportListener	1952
6.364.3.14getWireFormat	1952
6.364.3.15sClosed	1952
6.364.3.16sConnected	1953
6.364.3.17sFailOnClose	1953
6.364.3.18sFailOnKeepAliveSends	1953
6.364.3.19sFailOnReceiveMessage	1953
6.364.3.20sFailOnSendMessage	1953
6.364.3.21sFailOnStart	1953
6.364.3.22sFailOnStop	1953
6.364.3.23sFaultTolerant	1953
6.364.3.24sReconnectSupported	1954
6.364.3.25sUpdateURIsSupported	1954
6.364.3.26narrow	1954
6.364.3.27oneway	1954
6.364.3.28reconnect	1955
6.364.3.29request	1955
6.364.3.30request	1955
6.364.3.31setFailOnClose	1956
6.364.3.32setFailOnKeepAliveSends	1956
6.364.3.33setFailOnReceiveMessage	1956
6.364.3.34setFailOnSendMessage	1956

6.364.3.35	setFailOnStart	1956
6.364.3.36	setFailOnStop	1956
6.364.3.37	setName	1956
6.364.3.38	setNumReceivedMessageBeforeFail	1956
6.364.3.39	setNumReceivedMessages	1956
6.364.3.40	setNumSentKeepAlives	1956
6.364.3.41	setNumSentKeepAlivesBeforeFail	1956
6.364.3.42	setNumSentMessageBeforeFail	1956
6.364.3.43	setNumSentMessages	1957
6.364.3.44	setOutgoingListener	1957
6.364.3.45	setResponseBuilder	1957
6.364.3.46	setTransportListener	1957
6.364.3.47	setWireFormat	1957
6.364.3.48	start	1958
6.364.3.49	stop	1958
6.364.3.50	updateURIs	1958
6.365	activemq::transport::mock::MockTransportFactory Class Reference . . .	1958
6.365.1	Detailed Description	1959
6.365.2	Constructor & Destructor Documentation	1959
6.365.2.1	~MockTransportFactory	1959
6.365.3	Member Function Documentation	1959
6.365.3.1	create	1959
6.365.3.2	createComposite	1960
6.365.3.3	doCreateComposite	1960
6.366	decaf::util::concurrent::Mutex Class Reference	1960
6.366.1	Detailed Description	1961
6.366.2	Constructor & Destructor Documentation	1961
6.366.2.1	Mutex	1961
6.366.2.2	Mutex	1961
6.366.2.3	~Mutex	1962
6.366.3	Member Function Documentation	1962
6.366.3.1	getName	1962
6.366.3.2	lock	1962
6.366.3.3	notify	1962

6.366.3.4	notifyAll	1962
6.366.3.5	toString	1963
6.366.3.6	tryLock	1963
6.366.3.7	unlock	1963
6.366.3.8	wait	1964
6.366.3.9	wait	1964
6.366.3.10	wait	1965
6.367	decaf::util::concurrent::MutexHandle Class Reference	1965
6.367.1	Constructor & Destructor Documentation	1966
6.367.1.1	MutexHandle	1966
6.367.1.2	~MutexHandle	1966
6.367.1.3	MutexHandle	1966
6.367.1.4	~MutexHandle	1966
6.367.2	Field Documentation	1966
6.367.2.1	lock_count	1966
6.367.2.2	lock_owner	1966
6.367.2.3	mutex	1966
6.367.2.4	mutex	1966
6.368	decaf::internal::util::concurrent::MutexImpl Class Reference	1966
6.368.1	Member Function Documentation	1967
6.368.1.1	create	1967
6.368.1.2	destroy	1967
6.368.1.3	lock	1967
6.368.1.4	trylock	1967
6.368.1.5	unlock	1968
6.369	decaf::internal::net::Network Class Reference	1968
6.369.1	Detailed Description	1969
6.369.2	Constructor & Destructor Documentation	1969
6.369.2.1	Network	1969
6.369.2.2	~Network	1969
6.369.3	Member Function Documentation	1969
6.369.3.1	addAsResource	1969
6.369.3.2	addNetworkResource	1969
6.369.3.3	addShutdownTask	1970

6.369.3.4	getNetworkRuntime	1970
6.369.3.5	getRuntimeLock	1970
6.369.3.6	initializeNetworking	1971
6.369.3.7	shutdownNetworking	1971
6.370	activemq::commands::NetworkBridgeFilter Class Reference	1971
6.370.1	Constructor & Destructor Documentation	1972
6.370.1.1	NetworkBridgeFilter	1972
6.370.1.2	~NetworkBridgeFilter	1972
6.370.2	Member Function Documentation	1972
6.370.2.1	cloneDataStructure	1972
6.370.2.2	copyDataStructure	1972
6.370.2.3	equals	1973
6.370.2.4	getDataStructureType	1973
6.370.2.5	getNetworkBrokerId	1973
6.370.2.6	getNetworkBrokerId	1973
6.370.2.7	getNetworkTTL	1973
6.370.2.8	setNetworkBrokerId	1973
6.370.2.9	setNetworkTTL	1974
6.370.2.10	toString	1974
6.370.3	Field Documentation	1974
6.370.3.1	ID_NETWORKBRIDGEFILTER	1974
6.370.3.2	networkBrokerId	1974
6.370.3.3	networkTTL	1974
6.371	activemq::wireformat::openwire::marshal::generated::NetworkBridge- FilterMarshaller Class Reference	1974
6.371.1	Detailed Description	1975
6.371.2	Constructor & Destructor Documentation	1975
6.371.2.1	NetworkBridgeFilterMarshaller	1975
6.371.2.2	~NetworkBridgeFilterMarshaller	1975
6.371.3	Member Function Documentation	1975
6.371.3.1	createObject	1975
6.371.3.2	getDataStructureType	1976
6.371.3.3	looseMarshal	1976
6.371.3.4	looseUnmarshal	1976

6.371.3.5 tightMarshal1	1977
6.371.3.6 tightMarshal2	1977
6.371.3.7 tightUnmarshal	1978
6.372decaf::net::NoRouteToHostException Class Reference	1978
6.372.1 Constructor & Destructor Documentation	1979
6.372.1.1 NoRouteToHostException	1979
6.372.1.2 NoRouteToHostException	1979
6.372.1.3 NoRouteToHostException	1979
6.372.1.4 NoRouteToHostException	1980
6.372.1.5 NoRouteToHostException	1980
6.372.1.6 NoRouteToHostException	1980
6.372.1.7 ~NoRouteToHostException	1980
6.372.2 Member Function Documentation	1981
6.372.2.1 clone	1981
6.373decaf::security::NoSuchAlgorithmException Class Reference	1981
6.373.1 Constructor & Destructor Documentation	1982
6.373.1.1 NoSuchAlgorithmException	1982
6.373.1.2 NoSuchAlgorithmException	1982
6.373.1.3 NoSuchAlgorithmException	1982
6.373.1.4 NoSuchAlgorithmException	1982
6.373.1.5 NoSuchAlgorithmException	1983
6.373.1.6 NoSuchAlgorithmException	1983
6.373.1.7 ~NoSuchAlgorithmException	1983
6.373.2 Member Function Documentation	1983
6.373.2.1 clone	1983
6.374decaf::util::NoSuchElementException Class Reference	1984
6.374.1 Constructor & Destructor Documentation	1984
6.374.1.1 NoSuchElementException	1984
6.374.1.2 NoSuchElementException	1984
6.374.1.3 NoSuchElementException	1985
6.374.1.4 NoSuchElementException	1985
6.374.1.5 NoSuchElementException	1985
6.374.1.6 NoSuchElementException	1985
6.374.1.7 ~NoSuchElementException	1986

6.374.2 Member Function Documentation	1986
6.374.2.1 clone	1986
6.375decaf::security::NoSuchProviderException Class Reference	1986
6.375.1 Constructor & Destructor Documentation	1987
6.375.1.1 NoSuchProviderException	1987
6.375.1.2 NoSuchProviderException	1987
6.375.1.3 NoSuchProviderException	1987
6.375.1.4 NoSuchProviderException	1988
6.375.1.5 NoSuchProviderException	1988
6.375.1.6 NoSuchProviderException	1988
6.375.1.7 ~NoSuchProviderException	1988
6.375.2 Member Function Documentation	1989
6.375.2.1 clone	1989
6.376decaf::lang::exceptions::NullPointerException Class Reference	1989
6.376.1 Constructor & Destructor Documentation	1990
6.376.1.1 NullPointerException	1990
6.376.1.2 NullPointerException	1990
6.376.1.3 NullPointerException	1990
6.376.1.4 NullPointerException	1990
6.376.1.5 NullPointerException	1991
6.376.1.6 NullPointerException	1991
6.376.1.7 ~NullPointerException	1991
6.376.2 Member Function Documentation	1991
6.376.2.1 clone	1991
6.377decaf::lang::Number Class Reference	1992
6.377.1 Detailed Description	1992
6.377.2 Constructor & Destructor Documentation	1992
6.377.2.1 ~Number	1992
6.377.3 Member Function Documentation	1992
6.377.3.1 byteValue	1993
6.377.3.2 doubleValue	1993
6.377.3.3 floatValue	1993
6.377.3.4 intValue	1993
6.377.3.5 longValue	1994

6.377.3.6 shortValue	1994
6.378decaf::lang::exceptions::NumberFormatException Class Reference . . .	1994
6.378.1 Constructor & Destructor Documentation	1995
6.378.1.1 NumberFormatException	1995
6.378.1.2 NumberFormatException	1995
6.378.1.3 NumberFormatException	1995
6.378.1.4 NumberFormatException	1996
6.378.1.5 NumberFormatException	1996
6.378.1.6 NumberFormatException	1996
6.378.1.7 ~NumberFormatException	1997
6.378.2 Member Function Documentation	1997
6.378.2.1 clone	1997
6.379cms::ObjectMessage Class Reference	1997
6.379.1 Detailed Description	1998
6.379.2 Constructor & Destructor Documentation	1998
6.379.2.1 ~ObjectMessage	1998
6.380decaf::internal::net::ssl::openssl::OpenSSLContextSpi Class Reference .	1998
6.380.1 Detailed Description	1999
6.380.2 Constructor & Destructor Documentation	1999
6.380.2.1 OpenSSLContextSpi	1999
6.380.2.2 ~OpenSSLContextSpi	1999
6.380.3 Member Function Documentation	1999
6.380.3.1 providerGetServerSocketFactory	2000
6.380.3.2 providerGetSocketFactory	2000
6.380.3.3 providerInit	2000
6.380.4 Friends And Related Function Documentation	2001
6.380.4.1 OpenSSLSocket	2001
6.380.4.2 OpenSSLSocketFactory	2001
6.381decaf::internal::net::ssl::openssl::OpenSSLParameters Class Reference	2001
6.381.1 Detailed Description	2002
6.381.2 Constructor & Destructor Documentation	2002
6.381.2.1 ~OpenSSLParameters	2002
6.381.3 Member Function Documentation	2002
6.381.3.1 clone	2002

6.381.3.2	getEnabledCipherSuites	2002
6.381.3.3	getEnabledProtocols	2002
6.381.3.4	getNeedClientAuth	2002
6.381.3.5	getSupportedCipherSuites	2002
6.381.3.6	getSupportedProtocols	2002
6.381.3.7	getUseClientMode	2002
6.381.3.8	getWantClientAuth	2003
6.381.3.9	setEnabledCipherSuites	2003
6.381.3.10	setEnabledProtocols	2003
6.381.3.11	setNeedClientAuth	2003
6.381.3.12	setUseClientMode	2003
6.381.3.13	setWantClientAuth	2003
6.382	decaf::internal::net::ssl::openssl::OpenSSLServerSocket Class Reference	2003
6.382.1	Detailed Description	2005
6.382.2	Constructor & Destructor Documentation	2005
6.382.2.1	OpenSSLServerSocket	2005
6.382.2.2	~OpenSSLServerSocket	2005
6.382.3	Member Function Documentation	2006
6.382.3.1	accept	2006
6.382.3.2	getEnabledCipherSuites	2006
6.382.3.3	getEnabledProtocols	2006
6.382.3.4	getNeedClientAuth	2007
6.382.3.5	getSupportedCipherSuites	2007
6.382.3.6	getSupportedProtocols	2007
6.382.3.7	getWantClientAuth	2007
6.382.3.8	setEnabledCipherSuites	2008
6.382.3.9	setEnabledProtocols	2008
6.382.3.10	setNeedClientAuth	2009
6.382.3.11	setWantClientAuth	2009
6.383	decaf::internal::net::ssl::openssl::OpenSSLServerSocketFactory Class - Reference	2009
6.383.1	Detailed Description	2011
6.383.2	Constructor & Destructor Documentation	2011
6.383.2.1	OpenSSLServerSocketFactory	2011

6.383.2.2 ~OpenSSLServerSocketFactory	2011
6.383.3 Member Function Documentation	2011
6.383.3.1 createServerSocket	2012
6.383.3.2 createServerSocket	2012
6.383.3.3 createServerSocket	2012
6.383.3.4 createServerSocket	2013
6.383.3.5 getDefaultCipherSuites	2014
6.383.3.6 getSupportedCipherSuites	2014
6.384decaf::internal::net::ssl::openssl::OpenSSLSocket Class Reference . . .	2014
6.384.1 Detailed Description	2019
6.384.2 Constructor & Destructor Documentation	2019
6.384.2.1 OpenSSLSocket	2019
6.384.2.2 OpenSSLSocket	2019
6.384.2.3 OpenSSLSocket	2020
6.384.2.4 OpenSSLSocket	2020
6.384.2.5 OpenSSLSocket	2020
6.384.2.6 ~OpenSSLSocket	2020
6.384.3 Member Function Documentation	2020
6.384.3.1 available	2020
6.384.3.2 close	2020
6.384.3.3 connect	2020
6.384.3.4 getEnabledCipherSuites	2021
6.384.3.5 getEnabledProtocols	2021
6.384.3.6 getInputStream	2022
6.384.3.7 getNeedClientAuth	2022
6.384.3.8 getOutputStream	2022
6.384.3.9 getSupportedCipherSuites	2023
6.384.3.10getSupportedProtocols	2023
6.384.3.11getUseClientMode	2023
6.384.3.12getWantClientAuth	2024
6.384.3.13read	2024
6.384.3.14sendUrgentData	2024
6.384.3.15setEnabledCipherSuites	2025
6.384.3.16setEnabledProtocols	2025

6.384.3.17	setNeedClientAuth	2026
6.384.3.18	setOOBInline	2026
6.384.3.19	setUseClientMode	2026
6.384.3.20	setWantClientAuth	2027
6.384.3.21	shutdownInput	2027
6.384.3.22	shutdownOutput	2027
6.384.3.23	startHandshake	2028
6.384.3.24	write	2028
6.385	decaf::internal::net::ssl::openssl::OpenSSLSocketException Class -	
	Reference	2029
6.385.1	Detailed Description	2030
6.385.2	Constructor & Destructor Documentation	2030
6.385.2.1	OpenSSLSocketException	2030
6.385.2.2	OpenSSLSocketException	2030
6.385.2.3	OpenSSLSocketException	2030
6.385.2.4	OpenSSLSocketException	2030
6.385.2.5	OpenSSLSocketException	2031
6.385.2.6	OpenSSLSocketException	2031
6.385.2.7	OpenSSLSocketException	2031
6.385.2.8	~OpenSSLSocketException	2032
6.385.3	Member Function Documentation	2032
6.385.3.1	clone	2032
6.385.3.2	getErrorString	2032
6.386	decaf::internal::net::ssl::openssl::OpenSSLSocketFactory Class -	
	Reference	2032
6.386.1	Detailed Description	2035
6.386.2	Constructor & Destructor Documentation	2036
6.386.2.1	OpenSSLSocketFactory	2036
6.386.2.2	~OpenSSLSocketFactory	2036
6.386.3	Member Function Documentation	2036
6.386.3.1	createSocket	2036
6.386.3.2	createSocket	2036
6.386.3.3	createSocket	2037
6.386.3.4	createSocket	2037

6.386.3.5 createSocket	2038
6.386.3.6 createSocket	2038
6.386.3.7 getDefaultCipherSuites	2039
6.386.3.8 getSupportedCipherSuites	2040
6.387decaf::internal::net::ssl::openssl::OpenSSLSocketInputStream Class	
Reference	2040
6.387.1 Detailed Description	2041
6.387.2 Constructor & Destructor Documentation	2041
6.387.2.1 OpenSSLSocketInputStream	2041
6.387.2.2 ~OpenSSLSocketInputStream	2041
6.387.3 Member Function Documentation	2041
6.387.3.1 available	2041
6.387.3.2 close	2042
6.387.3.3 doReadArrayBounded	2042
6.387.3.4 doReadByte	2042
6.387.3.5 skip	2042
6.388decaf::internal::net::ssl::openssl::OpenSSLSocketOutputStream Class -	
Reference	2043
6.388.1 Detailed Description	2044
6.388.2 Constructor & Destructor Documentation	2044
6.388.2.1 OpenSSLSocketOutputStream	2044
6.388.2.2 ~OpenSSLSocketOutputStream	2044
6.388.3 Member Function Documentation	2044
6.388.3.1 close	2044
6.388.3.2 doWriteArrayBounded	2045
6.388.3.3 doWriteByte	2045
6.389activemq::wireformat::openwire::OpenWireFormat Class Reference . . .	2045
6.389.1 Constructor & Destructor Documentation	2048
6.389.1.1 OpenWireFormat	2048
6.389.1.2 ~OpenWireFormat	2048
6.389.2 Member Function Documentation	2048
6.389.2.1 addMarshaller	2049
6.389.2.2 createNegotiator	2049
6.389.2.3 destroyMarshalers	2049

6.389.2.4 doUnmarshal	2049
6.389.2.5 getCacheSize	2050
6.389.2.6 getMaxInactivityDuration	2050
6.389.2.7 getMaxInactivityDurationInitialDelay	2050
6.389.2.8 getPreferredWireFormatInfo	2051
6.389.2.9 getVersion	2051
6.389.2.10 hasNegotiator	2051
6.389.2.11 inReceive	2051
6.389.2.12 isCacheEnabled	2052
6.389.2.13 isSizePrefixDisabled	2052
6.389.2.14 isStackTraceEnabled	2052
6.389.2.15 isTcpNoDelayEnabled	2052
6.389.2.16 isTightEncodingEnabled	2052
6.389.2.17 looseMarshalNestedObject	2053
6.389.2.18 looseUnmarshalNestedObject	2053
6.389.2.19 marshal	2053
6.389.2.20 renegotiateWireFormat	2054
6.389.2.21 setCacheEnabled	2054
6.389.2.22 setCacheSize	2054
6.389.2.23 setMaxInactivityDuration	2055
6.389.2.24 setMaxInactivityDurationInitialDelay	2055
6.389.2.25 setPreferredWireFormatInfo	2055
6.389.2.26 setSizePrefixDisabled	2055
6.389.2.27 setStackTraceEnabled	2056
6.389.2.28 setTcpNoDelayEnabled	2056
6.389.2.29 setTightEncodingEnabled	2056
6.389.2.30 setVersion	2056
6.389.2.31 tightMarshalNestedObject1	2057
6.389.2.32 tightMarshalNestedObject2	2057
6.389.2.33 tightUnmarshalNestedObject	2057
6.389.2.34 unmarshal	2058
6.389.3 Field Documentation	2058
6.389.3.1 DEFAULT_VERSION	2058
6.389.3.2 MAX_SUPPORTED_VERSION	2059

6.389.3.3 NULL_TYPE	2059
6.390activemq::wireformat::openwire::OpenWireFormatFactory Class -	
Reference	2059
6.390.1 Constructor & Destructor Documentation	2059
6.390.1.1 OpenWireFormatFactory	2059
6.390.1.2 ~OpenWireFormatFactory	2060
6.390.2 Member Function Documentation	2060
6.390.2.1 createWireFormat	2060
6.391activemq::wireformat::openwire::OpenWireFormatNegotiator Class -	
Reference	2060
6.391.1 Constructor & Destructor Documentation	2061
6.391.1.1 OpenWireFormatNegotiator	2061
6.391.1.2 ~OpenWireFormatNegotiator	2061
6.391.2 Member Function Documentation	2061
6.391.2.1 close	2062
6.391.2.2 onCommand	2062
6.391.2.3 oneway	2062
6.391.2.4 onTransportException	2063
6.391.2.5 request	2063
6.391.2.6 request	2063
6.391.2.7 start	2064
6.392activemq::wireformat::openwire::OpenWireResponseBuilder Class -	
Reference	2064
6.392.1 Detailed Description	2065
6.392.2 Constructor & Destructor Documentation	2065
6.392.2.1 OpenWireResponseBuilder	2065
6.392.2.2 ~OpenWireResponseBuilder	2065
6.392.3 Member Function Documentation	2065
6.392.3.1 buildIncomingCommands	2065
6.392.3.2 buildResponse	2066
6.393decaf::io::OutputStream Class Reference	2066
6.393.1 Detailed Description	2068
6.393.2 Constructor & Destructor Documentation	2068
6.393.2.1 OutputStream	2068
6.393.2.2 ~OutputStream	2068

6.393.3 Member Function Documentation	2068
6.393.3.1 close	2068
6.393.3.2 doWriteArray	2068
6.393.3.3 doWriteArrayBounded	2068
6.393.3.4 doWriteByte	2069
6.393.3.5 flush	2069
6.393.3.6 lock	2069
6.393.3.7 notify	2069
6.393.3.8 notifyAll	2070
6.393.3.9 toString	2070
6.393.3.10 tryLock	2070
6.393.3.11 unlock	2071
6.393.3.12 wait	2071
6.393.3.13 wait	2071
6.393.3.14 wait	2072
6.393.3.15 write	2072
6.393.3.16 write	2073
6.393.3.17 write	2073
6.394 decaf::io::OutputStreamWriter Class Reference	2074
6.394.1 Detailed Description	2074
6.394.2 Constructor & Destructor Documentation	2075
6.394.2.1 OutputStreamWriter	2075
6.394.2.2 ~OutputStreamWriter	2075
6.394.3 Member Function Documentation	2075
6.394.3.1 checkClosed	2075
6.394.3.2 close	2075
6.394.3.3 doWriteArrayBounded	2075
6.394.3.4 flush	2076
6.395 activemq::commands::PartialCommand Class Reference	2076
6.395.1 Constructor & Destructor Documentation	2077
6.395.1.1 PartialCommand	2077
6.395.1.2 ~PartialCommand	2077
6.395.2 Member Function Documentation	2077
6.395.2.1 cloneDataStructure	2077

6.395.2.2 copyDataStructure	2077
6.395.2.3 equals	2078
6.395.2.4 getCommandId	2078
6.395.2.5 getData	2078
6.395.2.6 getData	2078
6.395.2.7 getDataStructureType	2078
6.395.2.8 setCommandId	2078
6.395.2.9 setData	2078
6.395.2.10 toString	2078
6.395.3 Field Documentation	2079
6.395.3.1 commandId	2079
6.395.3.2 data	2079
6.395.3.3 ID_PARTIALCOMMAND	2079
6.396activemq::wireformat::openwire::marshal::generated::PartialCommand- Marshaller Class Reference	2079
6.396.1 Detailed Description	2080
6.396.2 Constructor & Destructor Documentation	2080
6.396.2.1 PartialCommandMarshaller	2080
6.396.2.2 ~PartialCommandMarshaller	2080
6.396.3 Member Function Documentation	2080
6.396.3.1 createObject	2080
6.396.3.2 getDataStructureType	2081
6.396.3.3 looseMarshal	2081
6.396.3.4 looseUnmarshal	2081
6.396.3.5 tightMarshal1	2082
6.396.3.6 tightMarshal2	2082
6.396.3.7 tightUnmarshal	2083
6.397decaf::lang::Pointer< T, REFCOUNTER > Class Template Reference	2083
6.397.1 Detailed Description	2085
6.397.2 Member Typedef Documentation	2085
6.397.2.1 CounterType	2085
6.397.2.2 PointerType	2086
6.397.2.3 ReferenceType	2086
6.397.3 Constructor & Destructor Documentation	2086

6.397.3.1 Pointer	2086
6.397.3.2 Pointer	2086
6.397.3.3 Pointer	2086
6.397.3.4 Pointer	2086
6.397.3.5 Pointer	2087
6.397.3.6 Pointer	2087
6.397.3.7 ~Pointer	2087
6.397.4 Member Function Documentation	2087
6.397.4.1 dynamicCast	2088
6.397.4.2 get	2088
6.397.4.3 operator!	2088
6.397.4.4 operator!=	2088
6.397.4.5 operator*	2088
6.397.4.6 operator*	2089
6.397.4.7 operator->	2089
6.397.4.8 operator->	2089
6.397.4.9 operator=	2089
6.397.4.10operator=	2089
6.397.4.11operator==	2089
6.397.4.12release	2090
6.397.4.13reset	2090
6.397.4.14staticCast	2090
6.397.4.15swap	2090
6.397.5 Friends And Related Function Documentation	2091
6.397.5.1 operator!=	2091
6.397.5.2 operator!=	2091
6.397.5.3 operator==	2091
6.397.5.4 operator==	2091
6.398decaf::lang::PointerComparator< T, R > Class Template Reference	2091
6.398.1 Detailed Description	2092
6.398.2 Constructor & Destructor Documentation	2092
6.398.2.1 ~PointerComparator	2092
6.398.3 Member Function Documentation	2092
6.398.3.1 compare	2092

6.398.3.2 operator()	2092
6.399activemq::cmsutil::PooledSession Class Reference	2092
6.399.1 Detailed Description	2095
6.399.2 Constructor & Destructor Documentation	2095
6.399.2.1 PooledSession	2095
6.399.2.2 ~PooledSession	2095
6.399.3 Member Function Documentation	2095
6.399.3.1 close	2095
6.399.3.2 commit	2095
6.399.3.3 createBrowser	2096
6.399.3.4 createBrowser	2096
6.399.3.5 createBytesMessage	2097
6.399.3.6 createBytesMessage	2097
6.399.3.7 createCachedConsumer	2097
6.399.3.8 createCachedProducer	2098
6.399.3.9 createConsumer	2098
6.399.3.10createConsumer	2099
6.399.3.11createConsumer	2099
6.399.3.12createDurableConsumer	2100
6.399.3.13createMapMessage	2101
6.399.3.14createMessage	2101
6.399.3.15createProducer	2101
6.399.3.16createQueue	2102
6.399.3.17createStreamMessage	2102
6.399.3.18createTemporaryQueue	2102
6.399.3.19createTemporaryTopic	2103
6.399.3.20createTextMessage	2103
6.399.3.21createTextMessage	2103
6.399.3.22createTopic	2104
6.399.3.23getAcknowledgeMode	2104
6.399.3.24getSession	2104
6.399.3.25getSession	2105
6.399.3.26sTransacted	2105
6.399.3.27recover	2105

6.399.3.28	rollback	2106
6.399.3.29	start	2106
6.399.3.30	stop	2106
6.399.3.31	unsubscribe	2107
6.400	decaf::net::PortUnreachableException Class Reference	2107
6.400.1	Constructor & Destructor Documentation	2108
6.400.1.1	PortUnreachableException	2108
6.400.1.2	PortUnreachableException	2108
6.400.1.3	PortUnreachableException	2108
6.400.1.4	PortUnreachableException	2108
6.400.1.5	PortUnreachableException	2109
6.400.1.6	PortUnreachableException	2109
6.400.1.7	~PortUnreachableException	2109
6.400.2	Member Function Documentation	2109
6.400.2.1	clone	2109
6.401	activemq::core::PrefetchPolicy Class Reference	2110
6.401.1	Detailed Description	2111
6.401.2	Constructor & Destructor Documentation	2111
6.401.2.1	PrefetchPolicy	2111
6.401.2.2	~PrefetchPolicy	2111
6.401.3	Member Function Documentation	2111
6.401.3.1	clone	2111
6.401.3.2	configure	2111
6.401.3.3	getDurableTopicPrefetch	2112
6.401.3.4	getMaxPrefetchLimit	2112
6.401.3.5	getQueueBrowserPrefetch	2112
6.401.3.6	getQueuePrefetch	2113
6.401.3.7	getTopicPrefetch	2113
6.401.3.8	setDurableTopicPrefetch	2113
6.401.3.9	setQueueBrowserPrefetch	2113
6.401.3.10	setQueuePrefetch	2114
6.401.3.11	setTopicPrefetch	2114
6.402	activemq::util::PrimitiveList Class Reference	2114
6.402.1	Detailed Description	2116

6.402.2 Constructor & Destructor Documentation	2116
6.402.2.1 PrimitiveList	2116
6.402.2.2 ~PrimitiveList	2116
6.402.2.3 PrimitiveList	2116
6.402.2.4 PrimitiveList	2116
6.402.3 Member Function Documentation	2117
6.402.3.1 getBool	2117
6.402.3.2 getByte	2117
6.402.3.3 getByteArray	2118
6.402.3.4 getChar	2118
6.402.3.5 getDouble	2118
6.402.3.6 getFloat	2119
6.402.3.7 getInt	2119
6.402.3.8 getLong	2120
6.402.3.9 getShort	2120
6.402.3.10 getString	2121
6.402.3.11 setBool	2121
6.402.3.12 setByte	2122
6.402.3.13 setByteArray	2122
6.402.3.14 setChar	2122
6.402.3.15 setDouble	2123
6.402.3.16 setFloat	2123
6.402.3.17 setInt	2123
6.402.3.18 setLong	2124
6.402.3.19 setShort	2124
6.402.3.20 setString	2125
6.402.3.21 toString	2125
6.403 activemq::util::PrimitiveMap Class Reference	2125
6.403.1 Detailed Description	2127
6.403.2 Constructor & Destructor Documentation	2127
6.403.2.1 PrimitiveMap	2127
6.403.2.2 ~PrimitiveMap	2127
6.403.2.3 PrimitiveMap	2127
6.403.2.4 PrimitiveMap	2127

6.403.3 Member Function Documentation	2128
6.403.3.1 getBool	2128
6.403.3.2 getByte	2128
6.403.3.3 getByteArray	2129
6.403.3.4 getChar	2129
6.403.3.5 getDouble	2130
6.403.3.6 getFloat	2130
6.403.3.7 getInt	2131
6.403.3.8 getLong	2131
6.403.3.9 getShort	2132
6.403.3.10 getString	2132
6.403.3.11 setBool	2132
6.403.3.12 setByte	2133
6.403.3.13 setByteArray	2133
6.403.3.14 setChar	2133
6.403.3.15 setDouble	2133
6.403.3.16 setFloat	2134
6.403.3.17 setInt	2134
6.403.3.18 setLong	2134
6.403.3.19 setShort	2134
6.403.3.20 setString	2135
6.403.3.21 toString	2135
6.404 activemq::wireformat::openwire::marshal::PrimitiveTypesMarshaller -	
Class Reference	2135
6.404.1 Detailed Description	2137
6.404.2 Constructor & Destructor Documentation	2137
6.404.2.1 PrimitiveTypesMarshaller	2137
6.404.2.2 ~PrimitiveTypesMarshaller	2137
6.404.3 Member Function Documentation	2137
6.404.3.1 marshal	2137
6.404.3.2 marshal	2137
6.404.3.3 marshalList	2138
6.404.3.4 marshalMap	2138
6.404.3.5 marshalPrimitive	2138

6.404.3.6 marshalPrimitiveList	2139
6.404.3.7 marshalPrimitiveMap	2139
6.404.3.8 unmarshal	2140
6.404.3.9 unmarshal	2140
6.404.3.10unmarshalList	2140
6.404.3.11unmarshalMap	2141
6.404.3.12unmarshalPrimitive	2141
6.404.3.13unmarshalPrimitiveList	2142
6.404.3.14unmarshalPrimitiveMap	2142
6.405activemq::util::PrimitiveValueNode::PrimitiveValue Union Reference . . .	2142
6.405.1 Detailed Description	2143
6.405.2 Field Documentation	2143
6.405.2.1 boolValue	2143
6.405.2.2 byteArrayValue	2143
6.405.2.3 byteValue	2143
6.405.2.4 charValue	2143
6.405.2.5 doubleValue	2143
6.405.2.6 floatValue	2143
6.405.2.7 intValue	2143
6.405.2.8 listValue	2143
6.405.2.9 longValue	2143
6.405.2.10mapValue	2143
6.405.2.11shortValue	2143
6.405.2.12stringValue	2144
6.406activemq::util::PrimitiveValueConverter Class Reference	2144
6.406.1 Detailed Description	2144
6.406.2 Constructor & Destructor Documentation	2144
6.406.2.1 PrimitiveValueConverter	2144
6.406.2.2 ~PrimitiveValueConverter	2145
6.406.3 Member Function Documentation	2145
6.406.3.1 convert	2145
6.407activemq::util::PrimitiveValueNode Class Reference	2145
6.407.1 Detailed Description	2148
6.407.2 Member Enumeration Documentation	2148

6.407.2.1 PrimitiveType	2148
6.407.3 Constructor & Destructor Documentation	2149
6.407.3.1 PrimitiveValueNode	2149
6.407.3.2 PrimitiveValueNode	2149
6.407.3.3 PrimitiveValueNode	2149
6.407.3.4 PrimitiveValueNode	2149
6.407.3.5 PrimitiveValueNode	2149
6.407.3.6 PrimitiveValueNode	2149
6.407.3.7 PrimitiveValueNode	2150
6.407.3.8 PrimitiveValueNode	2150
6.407.3.9 PrimitiveValueNode	2150
6.407.3.10 PrimitiveValueNode	2150
6.407.3.11 PrimitiveValueNode	2150
6.407.3.12 PrimitiveValueNode	2151
6.407.3.13 PrimitiveValueNode	2151
6.407.3.14 PrimitiveValueNode	2151
6.407.3.15 PrimitiveValueNode	2151
6.407.3.16 PrimitiveValueNode	2151
6.407.4 Member Function Documentation	2151
6.407.4.1 clear	2152
6.407.4.2 getBool	2152
6.407.4.3 getByte	2152
6.407.4.4 getByteArray	2152
6.407.4.5 getChar	2153
6.407.4.6 getDouble	2153
6.407.4.7 getFloat	2153
6.407.4.8 getInt	2153
6.407.4.9 getList	2154
6.407.4.10 getLong	2154
6.407.4.11 getMap	2154
6.407.4.12 getShort	2155
6.407.4.13 getString	2155
6.407.4.14 getType	2155
6.407.4.15 getValue	2155

6.407.4.16operator=	2156
6.407.4.17operator==	2156
6.407.4.18setBool	2156
6.407.4.19setByte	2156
6.407.4.20setByteArray	2157
6.407.4.21setChar	2157
6.407.4.22setDouble	2157
6.407.4.23setFloat	2157
6.407.4.24setInt	2157
6.407.4.25setList	2158
6.407.4.26setLong	2158
6.407.4.27setMap	2158
6.407.4.28setShort	2158
6.407.4.29setString	2158
6.407.4.30setValue	2159
6.407.4.31toString	2159
6.408decaf::security::Principal Class Reference	2159
6.408.1 Detailed Description	2160
6.408.2 Constructor & Destructor Documentation	2160
6.408.2.1 ~Principal	2160
6.408.3 Member Function Documentation	2160
6.408.3.1 equals	2160
6.408.3.2 getName	2160
6.409decaf::util::PriorityQueue< E > Class Template Reference	2160
6.409.1 Detailed Description	2163
6.409.2 Constructor & Destructor Documentation	2164
6.409.2.1 PriorityQueue	2164
6.409.2.2 PriorityQueue	2164
6.409.2.3 PriorityQueue	2164
6.409.2.4 PriorityQueue	2165
6.409.2.5 PriorityQueue	2165
6.409.2.6 ~PriorityQueue	2165
6.409.3 Member Function Documentation	2165
6.409.3.1 add	2165

6.409.3.2 clear	2166
6.409.3.3 comparator	2167
6.409.3.4 iterator	2167
6.409.3.5 iterator	2167
6.409.3.6 offer	2167
6.409.3.7 operator=	2168
6.409.3.8 operator=	2168
6.409.3.9 peek	2168
6.409.3.10poll	2169
6.409.3.11remove	2169
6.409.3.12remove	2170
6.409.3.13size	2171
6.409.4 Friends And Related Function Documentation	2171
6.409.4.1 PriorityQueueIterator	2171
6.410activemq::commands::ProducerAck Class Reference	2171
6.410.1 Constructor & Destructor Documentation	2172
6.410.1.1 ProducerAck	2172
6.410.1.2 ~ProducerAck	2172
6.410.2 Member Function Documentation	2172
6.410.2.1 cloneDataStructure	2172
6.410.2.2 copyDataStructure	2173
6.410.2.3 equals	2173
6.410.2.4 getDataStructureType	2173
6.410.2.5 getProducerId	2173
6.410.2.6 getProducerId	2173
6.410.2.7 getSize	2173
6.410.2.8 isProducerAck	2173
6.410.2.9 setProducerId	2174
6.410.2.10setSize	2174
6.410.2.11toString	2174
6.410.2.12visit	2174
6.410.3 Field Documentation	2174
6.410.3.1 ID_PRODUCERACK	2174
6.410.3.2 producerId	2174

6.410.3.3 size	2174
6.411activemq::wireformat::openwire::marshal::generated::ProducerAck- Marshaller Class Reference	2175
6.411.1 Detailed Description	2176
6.411.2 Constructor & Destructor Documentation	2176
6.411.2.1 ProducerAckMarshaller	2176
6.411.2.2 ~ProducerAckMarshaller	2176
6.411.3 Member Function Documentation	2176
6.411.3.1 createObject	2176
6.411.3.2 getDataStructureType	2176
6.411.3.3 looseMarshal	2177
6.411.3.4 looseUnmarshal	2177
6.411.3.5 tightMarshal1	2177
6.411.3.6 tightMarshal2	2178
6.411.3.7 tightUnmarshal	2178
6.412activemq::cmsutil::ProducerCallback Class Reference	2179
6.412.1 Detailed Description	2179
6.412.2 Constructor & Destructor Documentation	2179
6.412.2.1 ~ProducerCallback	2179
6.412.3 Member Function Documentation	2179
6.412.3.1 doInCms	2179
6.413activemq::cmsutil::CmsTemplate::ProducerExecutor Class Reference . .	2180
6.413.1 Constructor & Destructor Documentation	2180
6.413.1.1 ProducerExecutor	2181
6.413.1.2 ~ProducerExecutor	2181
6.413.2 Member Function Documentation	2181
6.413.2.1 doInCms	2181
6.413.2.2 getDestination	2181
6.413.3 Field Documentation	2181
6.413.3.1 action	2181
6.413.3.2 destination	2181
6.413.3.3 parent	2181
6.414activemq::commands::ProducerId Class Reference	2182
6.414.1 Member Typedef Documentation	2183

6.414.1.1	COMPARATOR	2183
6.414.2	Constructor & Destructor Documentation	2183
6.414.2.1	ProducerId	2183
6.414.2.2	ProducerId	2183
6.414.2.3	ProducerId	2183
6.414.2.4	ProducerId	2183
6.414.2.5	~ProducerId	2183
6.414.3	Member Function Documentation	2183
6.414.3.1	cloneDataStructure	2183
6.414.3.2	compareTo	2183
6.414.3.3	copyDataStructure	2184
6.414.3.4	equals	2184
6.414.3.5	equals	2184
6.414.3.6	getConnectionId	2184
6.414.3.7	getConnectionId	2184
6.414.3.8	getDataStructureType	2184
6.414.3.9	getParentId	2185
6.414.3.10	getSessionId	2185
6.414.3.11	getValue	2185
6.414.3.12	operator<	2185
6.414.3.13	operator=	2185
6.414.3.14	operator==	2185
6.414.3.15	setConnectionId	2185
6.414.3.16	setProducerSessionKey	2185
6.414.3.17	setSessionId	2185
6.414.3.18	setValue	2185
6.414.3.19	toString	2185
6.414.4	Field Documentation	2185
6.414.4.1	connectionId	2185
6.414.4.2	ID_PRODUCERID	2186
6.414.4.3	sessionId	2186
6.414.4.4	value	2186
6.415	activemq::wireformat::openwire::marshal::generated::ProducerId- Marshaller Class Reference	2186

6.415.1 Detailed Description	2187
6.415.2 Constructor & Destructor Documentation	2187
6.415.2.1 ProducerIdMarshaller	2187
6.415.2.2 ~ProducerIdMarshaller	2187
6.415.3 Member Function Documentation	2187
6.415.3.1 createObject	2187
6.415.3.2 getDataStructureType	2187
6.415.3.3 looseMarshal	2188
6.415.3.4 looseUnmarshal	2188
6.415.3.5 tightMarshal1	2189
6.415.3.6 tightMarshal2	2189
6.415.3.7 tightUnmarshal	2189
6.416activemq::commands::ProducerInfo Class Reference	2190
6.416.1 Constructor & Destructor Documentation	2191
6.416.1.1 ProducerInfo	2191
6.416.1.2 ~ProducerInfo	2191
6.416.2 Member Function Documentation	2191
6.416.2.1 cloneDataStructure	2191
6.416.2.2 copyDataStructure	2192
6.416.2.3 createRemoveCommand	2192
6.416.2.4 equals	2192
6.416.2.5 getBrokerPath	2192
6.416.2.6 getBrokerPath	2192
6.416.2.7 getDataStructureType	2192
6.416.2.8 getDestination	2193
6.416.2.9 getDestination	2193
6.416.2.10getProducerId	2193
6.416.2.11getProducerId	2193
6.416.2.12getWindowSize	2193
6.416.2.13sDispatchAsync	2193
6.416.2.14sProducerInfo	2193
6.416.2.15setBrokerPath	2193
6.416.2.16setDestination	2193
6.416.2.17setDispatchAsync	2193

6.416.2.18	setProducerId	2193
6.416.2.19	setWindowSize	2194
6.416.2.20	toString	2194
6.416.2.21	visit	2194
6.416.3	Field Documentation	2194
6.416.3.1	brokerPath	2194
6.416.3.2	destination	2194
6.416.3.3	dispatchAsync	2194
6.416.3.4	ID_PRODUCERINFO	2194
6.416.3.5	producerId	2194
6.416.3.6	windowSize	2194
6.417	activemq::wireformat::openwire::marshal::generated::ProducerInfo-	
	Marshaller Class Reference	2195
6.417.1	Detailed Description	2196
6.417.2	Constructor & Destructor Documentation	2196
6.417.2.1	ProducerInfoMarshaller	2196
6.417.2.2	~ProducerInfoMarshaller	2196
6.417.3	Member Function Documentation	2196
6.417.3.1	createObject	2196
6.417.3.2	getDataStructureType	2196
6.417.3.3	looseMarshal	2197
6.417.3.4	looseUnmarshal	2197
6.417.3.5	tightMarshal1	2197
6.417.3.6	tightMarshal2	2198
6.417.3.7	tightUnmarshal	2198
6.418	activemq::state::ProducerState Class Reference	2199
6.418.1	Constructor & Destructor Documentation	2199
6.418.1.1	ProducerState	2199
6.418.1.2	~ProducerState	2199
6.418.2	Member Function Documentation	2199
6.418.2.1	getInfo	2199
6.418.2.2	getTransactionState	2199
6.418.2.3	setTransactionState	2199
6.418.2.4	toString	2200

6.419decaf::util::Properties Class Reference	2200
6.419.1 Detailed Description	2201
6.419.2 Constructor & Destructor Documentation	2201
6.419.2.1 Properties	2201
6.419.2.2 Properties	2202
6.419.2.3 ~Properties	2202
6.419.3 Member Function Documentation	2202
6.419.3.1 clear	2202
6.419.3.2 clone	2202
6.419.3.3 copy	2202
6.419.3.4 equals	2202
6.419.3.5 getProperty	2203
6.419.3.6 getProperty	2203
6.419.3.7 hasProperty	2203
6.419.3.8 isEmpty	2204
6.419.3.9 load	2204
6.419.3.10load	2204
6.419.3.11operator=	2206
6.419.3.12propertyNames	2206
6.419.3.13remove	2207
6.419.3.14setProperty	2207
6.419.3.15size	2207
6.419.3.16store	2207
6.419.3.17store	2208
6.419.3.18toArray	2209
6.419.3.19toString	2209
6.419.4 Field Documentation	2209
6.419.4.1 defaults	2209
6.420decaf::util::logging::PropertiesChangeListener Class Reference	2210
6.420.1 Detailed Description	2210
6.420.2 Constructor & Destructor Documentation	2210
6.420.2.1 ~PropertiesChangeListener	2210
6.420.3 Member Function Documentation	2210
6.420.3.1 onPropertiesReset	2210

6.420.3.2 onPropertyChanged	2211
6.421decaf::net::ProtocolException Class Reference	2211
6.421.1 Constructor & Destructor Documentation	2212
6.421.1.1 ProtocolException	2212
6.421.1.2 ProtocolException	2212
6.421.1.3 ProtocolException	2212
6.421.1.4 ProtocolException	2212
6.421.1.5 ProtocolException	2212
6.421.1.6 ProtocolException	2213
6.421.1.7 ~ProtocolException	2213
6.421.2 Member Function Documentation	2213
6.421.2.1 clone	2213
6.422decaf::security::PublicKey Class Reference	2213
6.422.1 Detailed Description	2214
6.422.2 Constructor & Destructor Documentation	2214
6.422.2.1 ~PublicKey	2214
6.423decaf::io::PushbackInputStream Class Reference	2214
6.423.1 Detailed Description	2216
6.423.2 Constructor & Destructor Documentation	2216
6.423.2.1 PushbackInputStream	2216
6.423.2.2 PushbackInputStream	2216
6.423.2.3 ~PushbackInputStream	2217
6.423.3 Member Function Documentation	2217
6.423.3.1 available	2217
6.423.3.2 doReadArrayBounded	2217
6.423.3.3 doReadByte	2218
6.423.3.4 mark	2218
6.423.3.5 markSupported	2218
6.423.3.6 reset	2218
6.423.3.7 skip	2219
6.423.3.8 unread	2220
6.423.3.9 unread	2220
6.423.3.10unread	2220
6.424cms::Queue Class Reference	2221

6.424.1 Detailed Description	2221
6.424.2 Constructor & Destructor Documentation	2222
6.424.2.1 ~Queue	2222
6.424.3 Member Function Documentation	2222
6.424.3.1 getQueueName	2222
6.425decaf::util::Queue< E > Class Template Reference	2222
6.425.1 Detailed Description	2223
6.425.2 Constructor & Destructor Documentation	2223
6.425.2.1 ~Queue	2223
6.425.3 Member Function Documentation	2223
6.425.3.1 element	2223
6.425.3.2 offer	2224
6.425.3.3 peek	2225
6.425.3.4 poll	2225
6.425.3.5 remove	2226
6.426cms::QueueBrowser Class Reference	2227
6.426.1 Detailed Description	2227
6.426.2 Constructor & Destructor Documentation	2228
6.426.2.1 ~QueueBrowser	2228
6.426.3 Member Function Documentation	2228
6.426.3.1 getEnumeration	2228
6.426.3.2 getMessageSelector	2228
6.426.3.3 getQueue	2229
6.427decaf::util::Random Class Reference	2229
6.427.1 Detailed Description	2230
6.427.2 Constructor & Destructor Documentation	2230
6.427.2.1 Random	2230
6.427.2.2 Random	2231
6.427.2.3 ~Random	2231
6.427.3 Member Function Documentation	2231
6.427.3.1 next	2231
6.427.3.2 nextBoolean	2232
6.427.3.3 nextBytes	2232
6.427.3.4 nextBytes	2232

6.427.3.5	nextDouble	2233
6.427.3.6	nextFloat	2233
6.427.3.7	nextGaussian	2233
6.427.3.8	nextInt	2234
6.427.3.9	nextInt	2234
6.427.3.10	nextLong	2234
6.427.3.11	setSeed	2235
6.428	decaf::lang::Readable Class Reference	2235
6.428.1	Detailed Description	2235
6.428.2	Constructor & Destructor Documentation	2236
6.428.2.1	~Readable	2236
6.428.3	Member Function Documentation	2236
6.428.3.1	read	2236
6.429	activemq::transport::inactivity::ReadChecker Class Reference	2236
6.429.1	Detailed Description	2237
6.429.2	Constructor & Destructor Documentation	2237
6.429.2.1	ReadChecker	2237
6.429.2.2	~ReadChecker	2237
6.429.3	Member Function Documentation	2237
6.429.3.1	run	2237
6.430	decaf::io::Reader Class Reference	2237
6.430.1	Constructor & Destructor Documentation	2239
6.430.1.1	Reader	2239
6.430.1.2	~Reader	2239
6.430.2	Member Function Documentation	2239
6.430.2.1	doReadArray	2239
6.430.2.2	doReadArrayBounded	2239
6.430.2.3	doReadChar	2239
6.430.2.4	doReadCharBuffer	2239
6.430.2.5	doReadVector	2239
6.430.2.6	mark	2239
6.430.2.7	markSupported	2240
6.430.2.8	read	2240
6.430.2.9	read	2241

6.430.2.10	read	2241
6.430.2.11	read	2242
6.430.2.12	read	2242
6.430.2.13	ready	2242
6.430.2.14	reset	2243
6.430.2.15	skip	2243
6.431	decaf::nio::ReadOnlyBufferException Class Reference	2244
6.431.1	Constructor & Destructor Documentation	2244
6.431.1.1	ReadOnlyBufferException	2244
6.431.1.2	ReadOnlyBufferException	2244
6.431.1.3	ReadOnlyBufferException	2245
6.431.1.4	ReadOnlyBufferException	2245
6.431.1.5	ReadOnlyBufferException	2245
6.431.1.6	ReadOnlyBufferException	2245
6.431.1.7	~ReadOnlyBufferException	2246
6.431.2	Member Function Documentation	2246
6.431.2.1	clone	2246
6.432	decaf::util::concurrent::locks::ReadWriteLock Class Reference	2246
6.432.1	Detailed Description	2246
6.432.2	Constructor & Destructor Documentation	2248
6.432.2.1	~ReadWriteLock	2248
6.432.3	Member Function Documentation	2248
6.432.3.1	readLock	2248
6.432.3.2	writeLock	2248
6.433	activemq::cmsutil::CmsTemplate::ReceiveExecutor Class Reference	2248
6.433.1	Constructor & Destructor Documentation	2249
6.433.1.1	ReceiveExecutor	2249
6.433.1.2	~ReceiveExecutor	2249
6.433.2	Member Function Documentation	2249
6.433.2.1	doInCms	2249
6.433.2.2	getDestination	2249
6.433.2.3	getMessage	2250
6.433.3	Field Documentation	2250
6.433.3.1	destination	2250

6.433.3.2 message	2250
6.433.3.3 noLocal	2250
6.433.3.4 parent	2250
6.433.3.5 selector	2250
6.434activemq::core::RedeliveryPolicy Class Reference	2250
6.434.1 Detailed Description	2251
6.434.2 Constructor & Destructor Documentation	2252
6.434.2.1 RedeliveryPolicy	2252
6.434.2.2 ~RedeliveryPolicy	2252
6.434.3 Member Function Documentation	2252
6.434.3.1 clone	2252
6.434.3.2 configure	2252
6.434.3.3 getBackOffMultiplier	2252
6.434.3.4 getCollisionAvoidancePercent	2253
6.434.3.5 getInitialRedeliveryDelay	2253
6.434.3.6 getMaximumRedeliveries	2253
6.434.3.7 getNextRedeliveryDelay	2253
6.434.3.8 getRedeliveryDelay	2254
6.434.3.9 isUseCollisionAvoidance	2254
6.434.3.10isUseExponentialBackOff	2254
6.434.3.11setBackOffMultiplier	2254
6.434.3.12setCollisionAvoidancePercent	2255
6.434.3.13setInitialRedeliveryDelay	2255
6.434.3.14setMaximumRedeliveries	2255
6.434.3.15setRedeliveryDelay	2255
6.434.3.16setUseCollisionAvoidance	2255
6.434.3.17setUseExponentialBackOff	2256
6.434.4 Field Documentation	2256
6.434.4.1 NO_MAXIMUM_REDELIVERIES	2256
6.435decaf::util::concurrent::locks::ReentrantLock Class Reference	2256
6.435.1 Detailed Description	2257
6.435.2 Constructor & Destructor Documentation	2258
6.435.2.1 ReentrantLock	2258
6.435.2.2 ~ReentrantLock	2258

6.435.3 Member Function Documentation	2258
6.435.3.1 getHoldCount	2258
6.435.3.2 isFair	2258
6.435.3.3 isHeldByCurrentThread	2259
6.435.3.4 isLocked	2259
6.435.3.5 lock	2259
6.435.3.6 lockInterruptibly	2260
6.435.3.7 newCondition	2260
6.435.3.8 toString	2261
6.435.3.9 tryLock	2261
6.435.3.10 tryLock	2262
6.435.3.11 unlock	2263
6.436 decaf::util::concurrent::RejectedExecutionException Class Reference	2263
6.436.1 Constructor & Destructor Documentation	2264
6.436.1.1 RejectedExecutionException	2264
6.436.1.2 RejectedExecutionException	2264
6.436.1.3 RejectedExecutionException	2264
6.436.1.4 RejectedExecutionException	2264
6.436.1.5 RejectedExecutionException	2265
6.436.1.6 RejectedExecutionException	2265
6.436.1.7 ~RejectedExecutionException	2265
6.436.2 Member Function Documentation	2265
6.436.2.1 clone	2266
6.437 decaf::util::concurrent::RejectedExecutionHandler Class Reference	2266
6.437.1 Detailed Description	2266
6.437.2 Constructor & Destructor Documentation	2267
6.437.2.1 RejectedExecutionHandler	2267
6.437.2.2 ~RejectedExecutionHandler	2267
6.437.3 Member Function Documentation	2267
6.437.3.1 rejectedExecution	2267
6.438 activemq::commands::RemoveInfo Class Reference	2267
6.438.1 Constructor & Destructor Documentation	2268
6.438.1.1 RemoveInfo	2268
6.438.1.2 ~RemoveInfo	2268

6.438.2 Member Function Documentation	2269
6.438.2.1 cloneDataStructure	2269
6.438.2.2 copyDataStructure	2269
6.438.2.3 equals	2269
6.438.2.4 getDataStructureType	2269
6.438.2.5 getLastDeliveredSequenceId	2270
6.438.2.6 getObjectId	2270
6.438.2.7 getObjectId	2270
6.438.2.8 isRemoveInfo	2270
6.438.2.9 setLastDeliveredSequenceId	2270
6.438.2.10 setObjectId	2270
6.438.2.11 toString	2270
6.438.2.12 visit	2270
6.438.3 Field Documentation	2271
6.438.3.1 ID_REMOVEINFO	2271
6.438.3.2 lastDeliveredSequenceId	2271
6.438.3.3 objectId	2271
6.439activemq::wireformat::openwire::marshal::generated::RemoveInfo-	
Marshaller Class Reference	2271
6.439.1 Detailed Description	2272
6.439.2 Constructor & Destructor Documentation	2272
6.439.2.1 RemoveInfoMarshaller	2272
6.439.2.2 ~RemoveInfoMarshaller	2272
6.439.3 Member Function Documentation	2272
6.439.3.1 createObject	2272
6.439.3.2 getDataStructureType	2273
6.439.3.3 looseMarshal	2273
6.439.3.4 looseUnmarshal	2273
6.439.3.5 tightMarshal1	2274
6.439.3.6 tightMarshal2	2274
6.439.3.7 tightUnmarshal	2275
6.440activemq::commands::RemoveSubscriptionInfo Class Reference	2275
6.440.1 Constructor & Destructor Documentation	2276
6.440.1.1 RemoveSubscriptionInfo	2276

6.440.1.2 ~RemoveSubscriptionInfo	2276
6.440.2 Member Function Documentation	2276
6.440.2.1 cloneDataStructure	2277
6.440.2.2 copyDataStructure	2277
6.440.2.3 equals	2277
6.440.2.4 getClientId	2277
6.440.2.5 getClientId	2277
6.440.2.6 getConnectionId	2278
6.440.2.7 getConnectionId	2278
6.440.2.8 getDataStructureType	2278
6.440.2.9 getSubscriptionName	2278
6.440.2.10getSubscriptionName	2278
6.440.2.11isRemoveSubscriptionInfo	2278
6.440.2.12setClientId	2278
6.440.2.13setConnectionId	2278
6.440.2.14setSubscriptionName	2278
6.440.2.15toString	2279
6.440.2.16visit	2279
6.440.3 Field Documentation	2279
6.440.3.1 clientId	2279
6.440.3.2 connectionId	2279
6.440.3.3 ID_REMOVESUBSCRIPTIONINFO	2279
6.440.3.4 subscriptionName	2279
6.441activemq::wireformat::openwire::marshal::generated::RemoveSubscription- InfoMarshaller Class Reference	2280
6.441.1 Detailed Description	2280
6.441.2 Constructor & Destructor Documentation	2281
6.441.2.1 RemoveSubscriptionInfoMarshaller	2281
6.441.2.2 ~RemoveSubscriptionInfoMarshaller	2281
6.441.3 Member Function Documentation	2281
6.441.3.1 createObject	2281
6.441.3.2 getDataStructureType	2281
6.441.3.3 looseMarshal	2281
6.441.3.4 looseUnmarshal	2282

6.441.3.5 tightMarshal1	2282
6.441.3.6 tightMarshal2	2283
6.441.3.7 tightUnmarshal	2283
6.442activemq::commands::ReplayCommand Class Reference	2284
6.442.1 Constructor & Destructor Documentation	2285
6.442.1.1 ReplayCommand	2285
6.442.1.2 ~ReplayCommand	2285
6.442.2 Member Function Documentation	2285
6.442.2.1 cloneDataStructure	2285
6.442.2.2 copyDataStructure	2285
6.442.2.3 equals	2285
6.442.2.4 getDataStructureType	2286
6.442.2.5 getFirstNakNumber	2286
6.442.2.6 getLastNakNumber	2286
6.442.2.7 setFirstNakNumber	2286
6.442.2.8 setLastNakNumber	2286
6.442.2.9 toString	2286
6.442.2.10visit	2286
6.442.3 Field Documentation	2287
6.442.3.1 firstNakNumber	2287
6.442.3.2 ID_REPLAYCOMMAND	2287
6.442.3.3 lastNakNumber	2287
6.443activemq::wireformat::openwire::marshal::generated::ReplayCommand- Marshaller Class Reference	2287
6.443.1 Detailed Description	2288
6.443.2 Constructor & Destructor Documentation	2288
6.443.2.1 ReplayCommandMarshaller	2288
6.443.2.2 ~ReplayCommandMarshaller	2288
6.443.3 Member Function Documentation	2288
6.443.3.1 createObject	2288
6.443.3.2 getDataStructureType	2288
6.443.3.3 looseMarshal	2289
6.443.3.4 looseUnmarshal	2289
6.443.3.5 tightMarshal1	2290

6.443.3.6 tightMarshal2	2290
6.443.3.7 tightUnmarshal	2290
6.444activemq::cmsutil::CmsTemplate::ResolveProducerExecutor Class -	
Reference	2291
6.444.1 Constructor & Destructor Documentation	2291
6.444.1.1 ResolveProducerExecutor	2291
6.444.1.2 ~ResolveProducerExecutor	2291
6.444.2 Member Function Documentation	2291
6.444.2.1 getDestination	2292
6.445activemq::cmsutil::CmsTemplate::ResolveReceiveExecutor Class -	
Reference	2292
6.445.1 Constructor & Destructor Documentation	2292
6.445.1.1 ResolveReceiveExecutor	2292
6.445.1.2 ~ResolveReceiveExecutor	2292
6.445.2 Member Function Documentation	2292
6.445.2.1 getDestination	2292
6.446decaf::internal::util::Resource Class Reference	2293
6.446.1 Detailed Description	2293
6.446.2 Constructor & Destructor Documentation	2293
6.446.2.1 ~Resource	2293
6.447activemq::cmsutil::ResourceLifecycleManager Class Reference	2293
6.447.1 Detailed Description	2294
6.447.2 Constructor & Destructor Documentation	2294
6.447.2.1 ResourceLifecycleManager	2294
6.447.2.2 ResourceLifecycleManager	2294
6.447.2.3 ~ResourceLifecycleManager	2294
6.447.3 Member Function Documentation	2295
6.447.3.1 addConnection	2295
6.447.3.2 addDestination	2295
6.447.3.3 addMessageConsumer	2295
6.447.3.4 addMessageProducer	2296
6.447.3.5 addSession	2296
6.447.3.6 destroy	2296
6.447.3.7 operator=	2296

6.447.3.8	releaseAll	2296
6.448	decaf::internal::util::ResourceLifecycleManager Class Reference	2297
6.448.1	Detailed Description	2297
6.448.2	Constructor & Destructor Documentation	2297
6.448.2.1	ResourceLifecycleManager	2297
6.448.2.2	~ResourceLifecycleManager	2297
6.448.3	Member Function Documentation	2297
6.448.3.1	addResource	2297
6.448.3.2	destroyResources	2297
6.449	activemq::commands::Response Class Reference	2298
6.449.1	Constructor & Destructor Documentation	2299
6.449.1.1	Response	2299
6.449.1.2	~Response	2299
6.449.2	Member Function Documentation	2299
6.449.2.1	cloneDataStructure	2299
6.449.2.2	copyDataStructure	2299
6.449.2.3	equals	2299
6.449.2.4	getCorrelationId	2300
6.449.2.5	getDataStructureType	2300
6.449.2.6	isResponse	2300
6.449.2.7	setCorrelationId	2300
6.449.2.8	toString	2300
6.449.2.9	visit	2301
6.449.3	Field Documentation	2301
6.449.3.1	correlationId	2301
6.449.3.2	ID_RESPONSE	2301
6.450	activemq::transport::mock::ResponseBuilder Class Reference	2301
6.450.1	Detailed Description	2302
6.450.2	Constructor & Destructor Documentation	2302
6.450.2.1	~ResponseBuilder	2302
6.450.3	Member Function Documentation	2302
6.450.3.1	buildIncomingCommands	2302
6.450.3.2	buildResponse	2303
6.451	activemq::transport::correlator::ResponseCorrelator Class Reference	2303

6.451.1 Detailed Description	2304
6.451.2 Constructor & Destructor Documentation	2304
6.451.2.1 ResponseCorrelator	2304
6.451.2.2 ~ResponseCorrelator	2304
6.451.3 Member Function Documentation	2304
6.451.3.1 close	2304
6.451.3.2 onCommand	2305
6.451.3.3 oneway	2305
6.451.3.4 onTransportException	2305
6.451.3.5 request	2306
6.451.3.6 request	2306
6.451.3.7 start	2307
6.452activemq::wireformat::openwire::marshal::generated::Response-	
Marshaller Class Reference	2307
6.452.1 Detailed Description	2308
6.452.2 Constructor & Destructor Documentation	2308
6.452.2.1 ResponseMarshaller	2308
6.452.2.2 ~ResponseMarshaller	2308
6.452.3 Member Function Documentation	2308
6.452.3.1 createObject	2308
6.452.3.2 getDataStructureType	2309
6.452.3.3 looseMarshal	2309
6.452.3.4 looseUnmarshal	2310
6.452.3.5 tightMarshal1	2310
6.452.3.6 tightMarshal2	2311
6.452.3.7 tightUnmarshal	2311
6.453decaf::lang::Runnable Class Reference	2312
6.453.1 Detailed Description	2312
6.453.2 Constructor & Destructor Documentation	2312
6.453.2.1 ~Runnable	2312
6.453.3 Member Function Documentation	2312
6.453.3.1 run	2312
6.454decaf::lang::Runtime Class Reference	2313
6.454.1 Constructor & Destructor Documentation	2313

6.454.1.1 ~Runtime	2313
6.454.2 Member Function Documentation	2313
6.454.2.1 getRuntime	2313
6.454.2.2 initializeRuntime	2314
6.454.2.3 initializeRuntime	2314
6.454.2.4 shutdownRuntime	2314
6.455decaf::lang::exceptions::RuntimeException Class Reference	2315
6.455.1 Constructor & Destructor Documentation	2315
6.455.1.1 RuntimeException	2315
6.455.1.2 RuntimeException	2315
6.455.1.3 RuntimeException	2316
6.455.1.4 RuntimeException	2316
6.455.1.5 RuntimeException	2316
6.455.1.6 RuntimeException	2316
6.455.1.7 ~RuntimeException	2317
6.455.2 Member Function Documentation	2317
6.455.2.1 clone	2317
6.456activemq::threads::Scheduler Class Reference	2317
6.456.1 Detailed Description	2318
6.456.2 Constructor & Destructor Documentation	2318
6.456.2.1 Scheduler	2318
6.456.2.2 ~Scheduler	2318
6.456.3 Member Function Documentation	2318
6.456.3.1 cancel	2318
6.456.3.2 doStart	2318
6.456.3.3 doStop	2319
6.456.3.4 executeAfterDelay	2319
6.456.3.5 executePeriodically	2319
6.456.3.6 schedualPeriodically	2319
6.456.3.7 shutdown	2319
6.457activemq::threads::SchedulerTimerTask Class Reference	2319
6.457.1 Detailed Description	2319
6.457.2 Constructor & Destructor Documentation	2320
6.457.2.1 SchedulerTimerTask	2320

6.457.2.2 ~SchedulerTimerTask	2320
6.457.3 Member Function Documentation	2320
6.457.3.1 run	2320
6.458decaf::security::SecureRandom Class Reference	2320
6.458.1 Detailed Description	2322
6.458.2 Constructor & Destructor Documentation	2322
6.458.2.1 SecureRandom	2322
6.458.2.2 SecureRandom	2322
6.458.2.3 SecureRandom	2322
6.458.2.4 ~SecureRandom	2323
6.458.3 Member Function Documentation	2323
6.458.3.1 next	2323
6.458.3.2 nextBytes	2324
6.458.3.3 nextBytes	2324
6.458.3.4 setSeed	2324
6.458.3.5 setSeed	2325
6.458.3.6 setSeed	2325
6.459decaf::internal::security::SecureRandomImpl Class Reference	2325
6.459.1 Detailed Description	2326
6.459.2 Constructor & Destructor Documentation	2327
6.459.2.1 SecureRandomImpl	2327
6.459.2.2 ~SecureRandomImpl	2327
6.459.2.3 SecureRandomImpl	2327
6.459.2.4 ~SecureRandomImpl	2327
6.459.3 Member Function Documentation	2327
6.459.3.1 providerGenerateSeed	2327
6.459.3.2 providerGenerateSeed	2327
6.459.3.3 providerNextBytes	2328
6.459.3.4 providerNextBytes	2328
6.459.3.5 providerSetSeed	2328
6.459.3.6 providerSetSeed	2329
6.460decaf::security::SecureRandomSpi Class Reference	2329
6.460.1 Detailed Description	2329
6.460.2 Constructor & Destructor Documentation	2330

6.460.2.1 SecureRandomSpi	2330
6.460.2.2 ~SecureRandomSpi	2330
6.460.3 Member Function Documentation	2330
6.460.3.1 providerGenerateSeed	2330
6.460.3.2 providerNextBytes	2330
6.460.3.3 providerSetSeed	2331
6.461 decaf::util::concurrent::Semaphore Class Reference	2331
6.461.1 Detailed Description	2332
6.461.2 Constructor & Destructor Documentation	2333
6.461.2.1 Semaphore	2334
6.461.2.2 Semaphore	2334
6.461.2.3 ~Semaphore	2334
6.461.3 Member Function Documentation	2334
6.461.3.1 acquire	2334
6.461.3.2 acquire	2335
6.461.3.3 acquireUninterruptibly	2335
6.461.3.4 acquireUninterruptibly	2336
6.461.3.5 availablePermits	2336
6.461.3.6 drainPermits	2337
6.461.3.7 isFair	2337
6.461.3.8 release	2337
6.461.3.9 release	2337
6.461.3.10 toString	2338
6.461.3.11 tryAcquire	2338
6.461.3.12 tryAcquire	2339
6.461.3.13 tryAcquire	2339
6.461.3.14 tryAcquire	2340
6.462 activemq::cmsutil::CmsTemplate::SendExecutor Class Reference	2341
6.462.1 Constructor & Destructor Documentation	2342
6.462.1.1 SendExecutor	2342
6.462.1.2 ~SendExecutor	2342
6.462.2 Member Function Documentation	2342
6.462.2.1 doInCms	2342
6.463 decaf::net::ServerSocket Class Reference	2342

6.463.1 Detailed Description	2344
6.463.2 Constructor & Destructor Documentation	2344
6.463.2.1 ServerSocket	2344
6.463.2.2 ServerSocket	2344
6.463.2.3 ServerSocket	2345
6.463.2.4 ServerSocket	2345
6.463.2.5 ~ServerSocket	2346
6.463.2.6 ServerSocket	2346
6.463.3 Member Function Documentation	2346
6.463.3.1 accept	2346
6.463.3.2 bind	2347
6.463.3.3 bind	2347
6.463.3.4 checkClosed	2348
6.463.3.5 close	2348
6.463.3.6 ensureCreated	2348
6.463.3.7 getDefaultBacklog	2348
6.463.3.8 getLocalPort	2348
6.463.3.9 getReceiveBufferSize	2348
6.463.3.10 getReuseAddress	2349
6.463.3.11 getSoTimeout	2349
6.463.3.12 implAccept	2349
6.463.3.13 isBound	2350
6.463.3.14 isClosed	2350
6.463.3.15 setReceiveBufferSize	2350
6.463.3.16 setReuseAddress	2350
6.463.3.17 setSocketImplFactory	2351
6.463.3.18 setSoTimeout	2351
6.463.3.19 setupSocketImpl	2351
6.463.3.20 toString	2352
6.464 decaf::net::ServerSocketFactory Class Reference	2352
6.464.1 Detailed Description	2353
6.464.2 Constructor & Destructor Documentation	2353
6.464.2.1 ServerSocketFactory	2353
6.464.2.2 ~ServerSocketFactory	2353

6.464.3 Member Function Documentation	2353
6.464.3.1 createServerSocket	2353
6.464.3.2 createServerSocket	2353
6.464.3.3 createServerSocket	2354
6.464.3.4 createServerSocket	2354
6.464.3.5 getDefault	2355
6.465activemq::util::Service Class Reference	2355
6.465.1 Detailed Description	2356
6.465.2 Constructor & Destructor Documentation	2356
6.465.2.1 ~Service	2356
6.465.3 Member Function Documentation	2356
6.465.3.1 start	2356
6.465.3.2 stop	2356
6.466activemq::util::ServiceListener Class Reference	2356
6.466.1 Detailed Description	2357
6.466.2 Constructor & Destructor Documentation	2357
6.466.2.1 ~ServiceListener	2357
6.466.3 Member Function Documentation	2357
6.466.3.1 started	2357
6.466.3.2 stopped	2357
6.467activemq::util::ServiceStopper Class Reference	2358
6.467.1 Constructor & Destructor Documentation	2358
6.467.1.1 ServiceStopper	2358
6.467.1.2 ~ServiceStopper	2358
6.467.2 Member Function Documentation	2358
6.467.2.1 onException	2358
6.467.2.2 stop	2358
6.467.2.3 throwFirstException	2358
6.468activemq::util::ServiceSupport Class Reference	2358
6.468.1 Detailed Description	2359
6.468.2 Constructor & Destructor Documentation	2360
6.468.2.1 ServiceSupport	2360
6.468.2.2 ServiceSupport	2360
6.468.2.3 ~ServiceSupport	2360

6.468.3 Member Function Documentation	2360
6.468.3.1 addServiceListener	2360
6.468.3.2 dispose	2360
6.468.3.3 doStart	2360
6.468.3.4 doStop	2360
6.468.3.5 isStarted	2360
6.468.3.6 isStopped	2361
6.468.3.7 isStopping	2361
6.468.3.8 operator=	2361
6.468.3.9 removeServiceListener	2361
6.468.3.10start	2361
6.468.3.11stop	2361
6.469cms::Session Class Reference	2361
6.469.1 Detailed Description	2363
6.469.2 Member Enumeration Documentation	2365
6.469.2.1 AcknowledgeMode	2365
6.469.3 Constructor & Destructor Documentation	2365
6.469.3.1 ~Session	2365
6.469.4 Member Function Documentation	2365
6.469.4.1 close	2365
6.469.4.2 commit	2366
6.469.4.3 createBrowser	2366
6.469.4.4 createBrowser	2367
6.469.4.5 createBytesMessage	2367
6.469.4.6 createBytesMessage	2367
6.469.4.7 createConsumer	2368
6.469.4.8 createConsumer	2368
6.469.4.9 createConsumer	2369
6.469.4.10createDurableConsumer	2370
6.469.4.11createMapMessage	2371
6.469.4.12createMessage	2371
6.469.4.13createProducer	2371
6.469.4.14createQueue	2372
6.469.4.15createStreamMessage	2372

6.469.4.16createTemporaryQueue	2373
6.469.4.17createTemporaryTopic	2373
6.469.4.18createTextMessage	2373
6.469.4.19createTextMessage	2374
6.469.4.20createTopic	2374
6.469.4.21getAcknowledgeMode	2374
6.469.4.22sTransacted	2375
6.469.4.23recover	2375
6.469.4.24rollback	2376
6.469.4.25unsubscribe	2376
6.470activemq::cmsutil::SessionCallback Class Reference	2377
6.470.1 Detailed Description	2377
6.470.2 Constructor & Destructor Documentation	2377
6.470.2.1 ~SessionCallback	2377
6.470.3 Member Function Documentation	2377
6.470.3.1 doInCms	2377
6.471activemq::commands::SessionId Class Reference	2378
6.471.1 Member Typedef Documentation	2379
6.471.1.1 COMPARATOR	2379
6.471.2 Constructor & Destructor Documentation	2379
6.471.2.1 SessionId	2379
6.471.2.2 SessionId	2379
6.471.2.3 SessionId	2379
6.471.2.4 SessionId	2379
6.471.2.5 SessionId	2379
6.471.2.6 ~SessionId	2379
6.471.3 Member Function Documentation	2379
6.471.3.1 cloneDataStructure	2380
6.471.3.2 compareTo	2380
6.471.3.3 copyDataStructure	2380
6.471.3.4 equals	2380
6.471.3.5 equals	2380
6.471.3.6 getConnectionId	2380
6.471.3.7 getConnectionId	2381

6.471.3.8	getDataStructureType	2381
6.471.3.9	getParentId	2381
6.471.3.10	getValue	2381
6.471.3.11	operator<	2381
6.471.3.12	operator=	2381
6.471.3.13	operator==	2381
6.471.3.14	setConnectionId	2381
6.471.3.15	setValue	2381
6.471.3.16	toString	2381
6.471.4	Field Documentation	2382
6.471.4.1	connectionId	2382
6.471.4.2	ID_SESSIONID	2382
6.471.4.3	value	2382
6.472	activemq::wireformat::openwire::marshal::generated::SessionIdMarshaller Class Reference	2382
6.472.1	Detailed Description	2383
6.472.2	Constructor & Destructor Documentation	2383
6.472.2.1	SessionIdMarshaller	2383
6.472.2.2	~SessionIdMarshaller	2383
6.472.3	Member Function Documentation	2383
6.472.3.1	createObject	2383
6.472.3.2	getDataStructureType	2384
6.472.3.3	looseMarshal	2384
6.472.3.4	looseUnmarshal	2384
6.472.3.5	tightMarshal1	2385
6.472.3.6	tightMarshal2	2385
6.472.3.7	tightUnmarshal	2386
6.473	activemq::commands::SessionInfo Class Reference	2386
6.473.1	Constructor & Destructor Documentation	2387
6.473.1.1	SessionInfo	2387
6.473.1.2	~SessionInfo	2387
6.473.2	Member Function Documentation	2387
6.473.2.1	cloneDataStructure	2387
6.473.2.2	copyDataStructure	2388

6.473.2.3 createRemoveCommand	2388
6.473.2.4 equals	2388
6.473.2.5 getAckMode	2388
6.473.2.6 getDataStructureType	2388
6.473.2.7 getSessionId	2388
6.473.2.8 getSessionId	2389
6.473.2.9 setAckMode	2389
6.473.2.10setSessionId	2389
6.473.2.11toString	2389
6.473.2.12visit	2389
6.473.3 Field Documentation	2389
6.473.3.1 ID_SESSIONINFO	2389
6.473.3.2 sessionId	2389
6.474activemq::wireformat::openwire::marshal::generated::SessionInfo- Marshaller Class Reference	2390
6.474.1 Detailed Description	2390
6.474.2 Constructor & Destructor Documentation	2391
6.474.2.1 SessionInfoMarshaller	2391
6.474.2.2 ~SessionInfoMarshaller	2391
6.474.3 Member Function Documentation	2391
6.474.3.1 createObject	2391
6.474.3.2 getDataStructureType	2391
6.474.3.3 looseMarshal	2391
6.474.3.4 looseUnmarshal	2392
6.474.3.5 tightMarshal1	2392
6.474.3.6 tightMarshal2	2393
6.474.3.7 tightUnmarshal	2393
6.475activemq::cmsutil::SessionPool Class Reference	2394
6.475.1 Detailed Description	2394
6.475.2 Constructor & Destructor Documentation	2394
6.475.2.1 SessionPool	2394
6.475.2.2 ~SessionPool	2395
6.475.3 Member Function Documentation	2395
6.475.3.1 getResourceLifecycleManager	2395

6.475.3.2 returnSession	2395
6.475.3.3 takeSession	2395
6.476activemq::state::SessionState Class Reference	2395
6.476.1 Constructor & Destructor Documentation	2396
6.476.1.1 SessionState	2396
6.476.1.2 ~SessionState	2396
6.476.2 Member Function Documentation	2396
6.476.2.1 addConsumer	2396
6.476.2.2 addProducer	2396
6.476.2.3 checkShutdown	2396
6.476.2.4 getConsumerState	2396
6.476.2.5 getConsumerStates	2396
6.476.2.6 getInfo	2397
6.476.2.7 getProducerState	2397
6.476.2.8 getProducerStates	2397
6.476.2.9 removeConsumer	2397
6.476.2.10removeProducer	2397
6.476.2.11shutdown	2397
6.476.2.12toString	2397
6.477decaf::util::Set< E > Class Template Reference	2397
6.477.1 Detailed Description	2397
6.477.2 Constructor & Destructor Documentation	2398
6.477.2.1 ~Set	2398
6.478decaf::lang::Short Class Reference	2398
6.478.1 Constructor & Destructor Documentation	2400
6.478.1.1 Short	2400
6.478.1.2 Short	2400
6.478.1.3 ~Short	2400
6.478.2 Member Function Documentation	2400
6.478.2.1 byteValue	2400
6.478.2.2 compareTo	2401
6.478.2.3 compareTo	2401
6.478.2.4 decode	2401
6.478.2.5 doubleValue	2402

6.478.2.6 equals	2402
6.478.2.7 equals	2402
6.478.2.8 floatValue	2402
6.478.2.9 intValue	2403
6.478.2.10longValue	2403
6.478.2.11operator<	2403
6.478.2.12operator<	2404
6.478.2.13operator==	2404
6.478.2.14operator==	2404
6.478.2.15parseShort	2405
6.478.2.16parseShort	2405
6.478.2.17reverseBytes	2406
6.478.2.18shortValue	2406
6.478.2.19toString	2406
6.478.2.20toString	2406
6.478.2.21valueOf	2407
6.478.2.22valueOf	2407
6.478.2.23valueOf	2407
6.478.3 Field Documentation	2408
6.478.3.1 MAX_VALUE	2408
6.478.3.2 MIN_VALUE	2408
6.478.3.3 SIZE	2408
6.479decaf::internal::nio::ShortArrayBuffer Class Reference	2408
6.479.1 Constructor & Destructor Documentation	2412
6.479.1.1 ShortArrayBuffer	2412
6.479.1.2 ShortArrayBuffer	2412
6.479.1.3 ShortArrayBuffer	2413
6.479.1.4 ShortArrayBuffer	2413
6.479.1.5 ~ShortArrayBuffer	2414
6.479.2 Member Function Documentation	2414
6.479.2.1 array	2414
6.479.2.2 arrayOffset	2414
6.479.2.3 asReadOnlyBuffer	2415
6.479.2.4 compact	2415

6.479.2.5 duplicate	2416
6.479.2.6 get	2416
6.479.2.7 get	2416
6.479.2.8 hasArray	2417
6.479.2.9 isReadOnly	2417
6.479.2.10put	2417
6.479.2.11put	2418
6.479.2.12setReadOnly	2418
6.479.2.13slice	2419
6.480decaf::nio::ShortBuffer Class Reference	2419
6.480.1 Detailed Description	2421
6.480.2 Constructor & Destructor Documentation	2421
6.480.2.1 ShortBuffer	2421
6.480.2.2 ~ShortBuffer	2421
6.480.3 Member Function Documentation	2421
6.480.3.1 allocate	2422
6.480.3.2 array	2422
6.480.3.3 arrayOffset	2422
6.480.3.4 asReadOnlyBuffer	2423
6.480.3.5 compact	2423
6.480.3.6 compareTo	2424
6.480.3.7 duplicate	2424
6.480.3.8 equals	2424
6.480.3.9 get	2424
6.480.3.10get	2425
6.480.3.11get	2425
6.480.3.12get	2425
6.480.3.13hasArray	2426
6.480.3.14operator<	2426
6.480.3.15operator==	2427
6.480.3.16put	2427
6.480.3.17put	2427
6.480.3.18put	2428
6.480.3.19put	2429

6.480.3.20	put	2429
6.480.3.21	slice	2430
6.480.3.22	toString	2430
6.480.3.23	wrap	2430
6.480.3.24	wrap	2431
6.481	activemq::commands::ShutdownInfo Class Reference	2431
6.481.1	Constructor & Destructor Documentation	2432
6.481.1.1	ShutdownInfo	2432
6.481.1.2	~ShutdownInfo	2432
6.481.2	Member Function Documentation	2432
6.481.2.1	cloneDataStructure	2432
6.481.2.2	copyDataStructure	2432
6.481.2.3	equals	2433
6.481.2.4	getDataStructureType	2433
6.481.2.5	isShutdownInfo	2433
6.481.2.6	toString	2433
6.481.2.7	visit	2434
6.481.3	Field Documentation	2434
6.481.3.1	ID_SHUTDOWNINFO	2434
6.482	activemq::wireformat::openwire::marshal::generated::ShutdownInfo- Marshaller Class Reference	2434
6.482.1	Detailed Description	2435
6.482.2	Constructor & Destructor Documentation	2435
6.482.2.1	ShutdownInfoMarshaller	2435
6.482.2.2	~ShutdownInfoMarshaller	2435
6.482.3	Member Function Documentation	2435
6.482.3.1	createObject	2435
6.482.3.2	getDataStructureType	2436
6.482.3.3	looseMarshal	2436
6.482.3.4	looseUnmarshal	2436
6.482.3.5	tightMarshal1	2437
6.482.3.6	tightMarshal2	2437
6.482.3.7	tightUnmarshal	2438
6.483	decaf::security::SignatureException Class Reference	2438

6.483.1 Constructor & Destructor Documentation	2439
6.483.1.1 SignatureException	2439
6.483.1.2 SignatureException	2439
6.483.1.3 SignatureException	2439
6.483.1.4 SignatureException	2439
6.483.1.5 SignatureException	2440
6.483.1.6 SignatureException	2440
6.483.1.7 ~SignatureException	2440
6.483.2 Member Function Documentation	2440
6.483.2.1 clone	2440
6.484decaf::util::logging::SimpleFormatter Class Reference	2441
6.484.1 Detailed Description	2441
6.484.2 Constructor & Destructor Documentation	2441
6.484.2.1 SimpleFormatter	2441
6.484.2.2 ~SimpleFormatter	2441
6.484.3 Member Function Documentation	2441
6.484.3.1 format	2442
6.485decaf::util::logging::SimpleLogger Class Reference	2442
6.485.1 Constructor & Destructor Documentation	2443
6.485.1.1 SimpleLogger	2443
6.485.1.2 ~SimpleLogger	2443
6.485.2 Member Function Documentation	2443
6.485.2.1 debug	2443
6.485.2.2 error	2443
6.485.2.3 fatal	2443
6.485.2.4 info	2443
6.485.2.5 log	2443
6.485.2.6 mark	2443
6.485.2.7 warn	2444
6.486activemq::core::SimplePriorityMessageDispatchChannel Class Reference	2444
6.486.1 Constructor & Destructor Documentation	2445
6.486.1.1 SimplePriorityMessageDispatchChannel	2445
6.486.1.2 ~SimplePriorityMessageDispatchChannel	2445
6.486.2 Member Function Documentation	2445

6.486.2.1 clear	2445
6.486.2.2 close	2446
6.486.2.3 dequeue	2446
6.486.2.4 dequeueNoWait	2446
6.486.2.5 enqueue	2446
6.486.2.6 enqueueFirst	2447
6.486.2.7 isClosed	2447
6.486.2.8 isEmpty	2447
6.486.2.9 isRunning	2447
6.486.2.10 lock	2448
6.486.2.11 notify	2448
6.486.2.12 notifyAll	2448
6.486.2.13 peek	2449
6.486.2.14 removeAll	2449
6.486.2.15 size	2449
6.486.2.16 start	2449
6.486.2.17 stop	2449
6.486.2.18 tryLock	2450
6.486.2.19 unlock	2450
6.486.2.20 wait	2450
6.486.2.21 wait	2451
6.486.2.22 wait	2451
6.487 decaf::net::Socket Class Reference	2452
6.487.1 Detailed Description	2455
6.487.2 Constructor & Destructor Documentation	2455
6.487.2.1 Socket	2455
6.487.2.2 Socket	2455
6.487.2.3 Socket	2455
6.487.2.4 Socket	2456
6.487.2.5 Socket	2456
6.487.2.6 Socket	2457
6.487.2.7 ~Socket	2457
6.487.3 Member Function Documentation	2457
6.487.3.1 accepted	2458

6.487.3.2 bind	2458
6.487.3.3 checkClosed	2458
6.487.3.4 close	2458
6.487.3.5 connect	2458
6.487.3.6 connect	2459
6.487.3.7 ensureCreated	2459
6.487.3.8 getInetAddress	2459
6.487.3.9 getInputStream	2459
6.487.3.10getKeepAlive	2460
6.487.3.11getLocalAddress	2460
6.487.3.12getLocalPort	2460
6.487.3.13getOOBInline	2461
6.487.3.14getOutputStream	2461
6.487.3.15getPort	2461
6.487.3.16getReceiveBufferSize	2462
6.487.3.17getReuseAddress	2462
6.487.3.18getSendBufferSize	2462
6.487.3.19getSoLinger	2463
6.487.3.20getSoTimeout	2463
6.487.3.21getTcpNoDelay	2463
6.487.3.22getTrafficClass	2463
6.487.3.23nitSocketImpl	2464
6.487.3.24sBound	2464
6.487.3.25sClosed	2464
6.487.3.26sConnected	2464
6.487.3.27sInputShutdown	2464
6.487.3.28sOutputShutdown	2465
6.487.3.29sendUrgentData	2465
6.487.3.30setKeepAlive	2465
6.487.3.31setOOBInline	2465
6.487.3.32setReceiveBufferSize	2466
6.487.3.33setReuseAddress	2466
6.487.3.34setSendBufferSize	2466
6.487.3.35setSocketImplFactory	2467

6.487.3.36	setSoLinger	2467
6.487.3.37	setSoTimeout	2468
6.487.3.38	setTcpNoDelay	2468
6.487.3.39	setTrafficClass	2468
6.487.3.40	shutdownInput	2469
6.487.3.41	shutdownOutput	2469
6.487.3.42	toString	2469
6.487.4	Friends And Related Function Documentation	2469
6.487.4.1	ServerSocket	2469
6.487.5	Field Documentation	2469
6.487.5.1	impl	2469
6.488	decaf::net::SocketAddress Class Reference	2470
6.488.1	Detailed Description	2470
6.488.2	Constructor & Destructor Documentation	2470
6.488.2.1	~SocketAddress	2470
6.489	decaf::net::SocketError Class Reference	2470
6.489.1	Detailed Description	2471
6.489.2	Member Function Documentation	2471
6.489.2.1	getErrorCode	2471
6.489.2.2	getErrorString	2471
6.490	decaf::net::SocketException Class Reference	2471
6.490.1	Detailed Description	2472
6.490.2	Constructor & Destructor Documentation	2472
6.490.2.1	SocketException	2472
6.490.2.2	SocketException	2472
6.490.2.3	SocketException	2472
6.490.2.4	SocketException	2472
6.490.2.5	SocketException	2472
6.490.2.6	SocketException	2472
6.490.2.7	~SocketException	2473
6.490.3	Member Function Documentation	2473
6.490.3.1	clone	2473
6.491	decaf::net::SocketFactory Class Reference	2473
6.491.1	Detailed Description	2474

6.491.2 Constructor & Destructor Documentation	2475
6.491.2.1 SocketFactory	2475
6.491.2.2 ~SocketFactory	2475
6.491.3 Member Function Documentation	2475
6.491.3.1 createSocket	2475
6.491.3.2 createSocket	2475
6.491.3.3 createSocket	2476
6.491.3.4 createSocket	2476
6.491.3.5 createSocket	2477
6.491.3.6 getDefault	2478
6.492decaf::internal::net::SocketFileDescriptor Class Reference	2478
6.492.1 Detailed Description	2478
6.492.2 Constructor & Destructor Documentation	2479
6.492.2.1 SocketFileDescriptor	2479
6.492.2.2 ~SocketFileDescriptor	2479
6.492.3 Member Function Documentation	2479
6.492.3.1 getValue	2479
6.493decaf::net::SocketImpl Class Reference	2479
6.493.1 Detailed Description	2481
6.493.2 Constructor & Destructor Documentation	2481
6.493.2.1 SocketImpl	2481
6.493.2.2 ~SocketImpl	2481
6.493.3 Member Function Documentation	2481
6.493.3.1 accept	2481
6.493.3.2 available	2481
6.493.3.3 bind	2482
6.493.3.4 close	2482
6.493.3.5 connect	2482
6.493.3.6 create	2483
6.493.3.7 getFileDescriptor	2483
6.493.3.8 getInetAddress	2483
6.493.3.9 getInputStream	2483
6.493.3.10getLocalAddress	2484
6.493.3.11getLocalPort	2484

6.493.3.12	getOption	2484
6.493.3.13	getOutputStream	2484
6.493.3.14	getPort	2485
6.493.3.15	listen	2485
6.493.3.16	sendUrgentData	2485
6.493.3.17	setOption	2486
6.493.3.18	shutdownInput	2486
6.493.3.19	shutdownOutput	2486
6.493.3.20	supportsUrgentData	2487
6.493.3.21	toString	2487
6.493.4	Field Documentation	2487
6.493.4.1	address	2487
6.493.4.2	fd	2487
6.493.4.3	localPort	2487
6.493.4.4	port	2487
6.494	decaf::net::SocketImplFactory Class Reference	2487
6.494.1	Detailed Description	2488
6.494.2	Constructor & Destructor Documentation	2488
6.494.2.1	~SocketImplFactory	2488
6.494.3	Member Function Documentation	2488
6.494.3.1	createSocketImpl	2488
6.495	decaf::net::SocketOptions Class Reference	2488
6.495.1	Detailed Description	2490
6.495.2	Constructor & Destructor Documentation	2490
6.495.2.1	~SocketOptions	2490
6.495.3	Field Documentation	2490
6.495.3.1	SOCKET_OPTION_BINDADDR	2490
6.495.3.2	SOCKET_OPTION_BROADCAST	2490
6.495.3.3	SOCKET_OPTION_IP_MULTICAST_IF	2490
6.495.3.4	SOCKET_OPTION_IP_MULTICAST_IF2	2490
6.495.3.5	SOCKET_OPTION_IP_MULTICAST_LOOP	2491
6.495.3.6	SOCKET_OPTION_IP_TOS	2491
6.495.3.7	SOCKET_OPTION_KEEPALIVE	2491
6.495.3.8	SOCKET_OPTION_LINGER	2491

6.495.3.9	SOCKET_OPTION_OOBLINE	2492
6.495.3.10	SOCKET_OPTION_RCVBUF	2492
6.495.3.11	SOCKET_OPTION_REUSEADDR	2492
6.495.3.12	SOCKET_OPTION_SNDBUF	2492
6.495.3.13	SOCKET_OPTION_TCP_NODELAY	2492
6.495.3.14	SOCKET_OPTION_TIMEOUT	2493
6.496	decaf::net::SocketTimeoutException Class Reference	2493
6.496.1	Constructor & Destructor Documentation	2493
6.496.1.1	SocketTimeoutException	2493
6.496.1.2	SocketTimeoutException	2494
6.496.1.3	SocketTimeoutException	2494
6.496.1.4	SocketTimeoutException	2494
6.496.1.5	SocketTimeoutException	2494
6.496.1.6	SocketTimeoutException	2495
6.496.1.7	~SocketTimeoutException	2495
6.496.2	Member Function Documentation	2495
6.496.2.1	clone	2495
6.497	decaf::net::ssl::SSLContext Class Reference	2495
6.497.1	Detailed Description	2496
6.497.2	Constructor & Destructor Documentation	2496
6.497.2.1	SSLContext	2496
6.497.2.2	~SSLContext	2496
6.497.3	Member Function Documentation	2496
6.497.3.1	getDefault	2496
6.497.3.2	getDefaultSSLParameters	2497
6.497.3.3	getServerSocketFactory	2497
6.497.3.4	getSocketFactory	2497
6.497.3.5	getSupportedSSLParameters	2498
6.497.3.6	setDefault	2498
6.498	decaf::net::ssl::SSLContextSpi Class Reference	2498
6.498.1	Detailed Description	2499
6.498.2	Constructor & Destructor Documentation	2499
6.498.2.1	~SSLContextSpi	2499
6.498.3	Member Function Documentation	2499

6.498.3.1 providerGetDefaultSSLParameters	2499
6.498.3.2 providerGetServerSocketFactory	2500
6.498.3.3 providerGetSocketFactory	2500
6.498.3.4 providerGetSupportedSSLParameters	2500
6.498.3.5 providerInit	2501
6.499decaf::net::ssl::SSLParameters Class Reference	2501
6.499.1 Constructor & Destructor Documentation	2502
6.499.1.1 SSLParameters	2502
6.499.1.2 SSLParameters	2502
6.499.1.3 SSLParameters	2502
6.499.1.4 ~SSLParameters	2503
6.499.2 Member Function Documentation	2503
6.499.2.1 getCipherSuites	2503
6.499.2.2 getNeedClientAuth	2503
6.499.2.3 getProtocols	2503
6.499.2.4 getWantClientAuth	2503
6.499.2.5 setCipherSuites	2503
6.499.2.6 setNeedClientAuth	2504
6.499.2.7 setProtocols	2504
6.499.2.8 setWantClientAuth	2504
6.500decaf::net::ssl::SSLServerSocket Class Reference	2504
6.500.1 Detailed Description	2506
6.500.2 Constructor & Destructor Documentation	2506
6.500.2.1 SSLServerSocket	2506
6.500.2.2 SSLServerSocket	2506
6.500.2.3 SSLServerSocket	2506
6.500.2.4 SSLServerSocket	2507
6.500.2.5 ~SSLServerSocket	2507
6.500.3 Member Function Documentation	2508
6.500.3.1 getEnabledCipherSuites	2508
6.500.3.2 getEnabledProtocols	2508
6.500.3.3 getNeedClientAuth	2508
6.500.3.4 getSupportedCipherSuites	2508
6.500.3.5 getSupportedProtocols	2509

6.500.3.6	getWantClientAuth	2509
6.500.3.7	setEnabledCipherSuites	2509
6.500.3.8	setEnabledProtocols	2509
6.500.3.9	setNeedClientAuth	2510
6.500.3.10	setWantClientAuth	2510
6.501	decaf::net::ssl::SSLServerSocketFactory Class Reference	2510
6.501.1	Detailed Description	2511
6.501.2	Constructor & Destructor Documentation	2511
6.501.2.1	SSLServerSocketFactory	2511
6.501.2.2	~SSLServerSocketFactory	2511
6.501.3	Member Function Documentation	2511
6.501.3.1	getDefault	2512
6.501.3.2	getDefaultCipherSuites	2512
6.501.3.3	getSupportedCipherSuites	2512
6.502	decaf::net::ssl::SSLSocket Class Reference	2513
6.502.1	Detailed Description	2514
6.502.2	Constructor & Destructor Documentation	2515
6.502.2.1	SSLSocket	2515
6.502.2.2	SSLSocket	2515
6.502.2.3	SSLSocket	2515
6.502.2.4	SSLSocket	2516
6.502.2.5	SSLSocket	2516
6.502.2.6	~SSLSocket	2517
6.502.3	Member Function Documentation	2517
6.502.3.1	setEnabledCipherSuites	2517
6.502.3.2	setEnabledProtocols	2517
6.502.3.3	getNeedClientAuth	2517
6.502.3.4	getSSLParameters	2517
6.502.3.5	getSupportedCipherSuites	2518
6.502.3.6	getSupportedProtocols	2518
6.502.3.7	getUseClientMode	2518
6.502.3.8	getWantClientAuth	2519
6.502.3.9	setEnabledCipherSuites	2519
6.502.3.10	setEnabledProtocols	2519

6.502.3.11	setNeedClientAuth	2520
6.502.3.12	setSSLParameters	2520
6.502.3.13	setUseClientMode	2520
6.502.3.14	setWantClientAuth	2521
6.502.3.15	startHandshake	2521
6.503	decaf::net::ssl::SSLSocketFactory Class Reference	2522
6.503.1	Detailed Description	2522
6.503.2	Constructor & Destructor Documentation	2523
6.503.2.1	SSLSocketFactory	2523
6.503.2.2	~SSLSocketFactory	2523
6.503.3	Member Function Documentation	2523
6.503.3.1	createSocket	2523
6.503.3.2	getDefault	2523
6.503.3.3	getDefaultCipherSuites	2524
6.503.3.4	getSupportedCipherSuites	2524
6.504	activemq::transport::tcp::SslTransport Class Reference	2525
6.504.1	Detailed Description	2525
6.504.2	Constructor & Destructor Documentation	2526
6.504.2.1	SslTransport	2526
6.504.2.2	~SslTransport	2526
6.504.3	Member Function Documentation	2526
6.504.3.1	configureSocket	2526
6.504.3.2	createSocket	2526
6.505	activemq::transport::tcp::SslTransportFactory Class Reference	2527
6.505.1	Constructor & Destructor Documentation	2527
6.505.1.1	~SslTransportFactory	2527
6.505.2	Member Function Documentation	2527
6.505.2.1	doCreateComposite	2527
6.506	activemq::commands::BrokerError::StackTraceElement Struct Reference	2527
6.506.1	Constructor & Destructor Documentation	2528
6.506.1.1	StackTraceElement	2528
6.506.2	Field Documentation	2528
6.506.2.1	ClassName	2528
6.506.2.2	FileName	2528

6.506.2.3	LineNumber	2528
6.506.2.4	MethodName	2528
6.507	decaf::internal::io::StandardOutputStream Class Reference	2528
6.507.1	Detailed Description	2529
6.507.2	Constructor & Destructor Documentation	2529
6.507.2.1	StandardOutputStream	2529
6.507.2.2	~StandardOutputStream	2529
6.507.3	Member Function Documentation	2529
6.507.3.1	close	2529
6.507.3.2	doWriteArrayBounded	2530
6.507.3.3	doWriteByte	2530
6.507.3.4	flush	2530
6.508	decaf::internal::io::StandardInputStream Class Reference	2530
6.508.1	Constructor & Destructor Documentation	2531
6.508.1.1	StandardInputStream	2531
6.508.1.2	~StandardInputStream	2531
6.508.2	Member Function Documentation	2531
6.508.2.1	available	2531
6.508.2.2	doReadByte	2531
6.509	decaf::internal::io::StandardOutputStream Class Reference	2532
6.509.1	Constructor & Destructor Documentation	2532
6.509.1.1	StandardOutputStream	2532
6.509.1.2	~StandardOutputStream	2532
6.509.2	Member Function Documentation	2532
6.509.2.1	close	2533
6.509.2.2	doWriteArrayBounded	2533
6.509.2.3	doWriteByte	2533
6.509.2.4	flush	2533
6.510	cms::Startable Class Reference	2534
6.510.1	Detailed Description	2534
6.510.2	Constructor & Destructor Documentation	2534
6.510.2.1	~Startable	2534
6.510.3	Member Function Documentation	2534
6.510.3.1	start	2534

6.511	decaf::lang::STATIC_CAST_TOKEN Struct Reference	2535
6.512	activemq::core::ActiveMQConstants::StaticInitializer Class Reference . .	2535
6.512.1	Constructor & Destructor Documentation	2535
6.512.1.1	StaticInitializer	2535
6.512.1.2	~StaticInitializer	2535
6.512.2	Field Documentation	2535
6.512.2.1	destOptionMap	2535
6.512.2.2	destOptions	2536
6.512.2.3	uriParams	2536
6.512.2.4	uriParamsMap	2536
6.513	decaf::util::StlList< E > Class Template Reference	2536
6.513.1	Detailed Description	2541
6.513.2	Constructor & Destructor Documentation	2541
6.513.2.1	StlList	2541
6.513.2.2	StlList	2541
6.513.2.3	StlList	2541
6.513.3	Member Function Documentation	2542
6.513.3.1	add	2542
6.513.3.2	add	2542
6.513.3.3	addAll	2543
6.513.3.4	addAll	2544
6.513.3.5	clear	2545
6.513.3.6	contains	2545
6.513.3.7	copy	2546
6.513.3.8	equals	2546
6.513.3.9	get	2547
6.513.3.10	indexOf	2547
6.513.3.11	isEmpty	2548
6.513.3.12	iterator	2548
6.513.3.13	iterator	2548
6.513.3.14	lastIndexOf	2548
6.513.3.15	listIterator	2549
6.513.3.16	listIterator	2549
6.513.3.17	listIterator	2549

6.513.3.18	istliterator	2550
6.513.3.19	remove	2550
6.513.3.20	removeAt	2550
6.513.3.21	set	2551
6.513.3.22	size	2552
6.514	decaf::util::StlMap< K, V, COMPARATOR > Class Template Reference	2552
6.514.1	Detailed Description	2556
6.514.2	Constructor & Destructor Documentation	2556
6.514.2.1	StlMap	2556
6.514.2.2	StlMap	2556
6.514.2.3	StlMap	2556
6.514.2.4	~StlMap	2556
6.514.3	Member Function Documentation	2556
6.514.3.1	clear	2557
6.514.3.2	containsKey	2557
6.514.3.3	containsValue	2557
6.514.3.4	copy	2558
6.514.3.5	copy	2558
6.514.3.6	equals	2558
6.514.3.7	equals	2558
6.514.3.8	get	2559
6.514.3.9	get	2559
6.514.3.10	isEmpty	2560
6.514.3.11	keySet	2560
6.514.3.12	lock	2560
6.514.3.13	notify	2560
6.514.3.14	notifyAll	2561
6.514.3.15	put	2561
6.514.3.16	putAll	2562
6.514.3.17	putAll	2562
6.514.3.18	remove	2562
6.514.3.19	size	2563
6.514.3.20	tryLock	2563
6.514.3.21	unlock	2563

6.514.3.22	values	2563
6.514.3.23	wait	2564
6.514.3.24	wait	2564
6.514.3.25	wait	2565
6.515	decaf::util::StlQueue< T > Class Template Reference	2565
6.515.1	Detailed Description	2567
6.515.2	Constructor & Destructor Documentation	2567
6.515.2.1	StlQueue	2567
6.515.2.2	~StlQueue	2567
6.515.3	Member Function Documentation	2567
6.515.3.1	back	2567
6.515.3.2	back	2568
6.515.3.3	clear	2568
6.515.3.4	empty	2568
6.515.3.5	enqueueFront	2568
6.515.3.6	front	2568
6.515.3.7	front	2569
6.515.3.8	getSafeValue	2569
6.515.3.9	iterator	2569
6.515.3.10	lock	2569
6.515.3.11	notify	2570
6.515.3.12	notifyAll	2570
6.515.3.13	pop	2570
6.515.3.14	push	2571
6.515.3.15	reverse	2571
6.515.3.16	size	2571
6.515.3.17	toArray	2571
6.515.3.18	tryLock	2571
6.515.3.19	unlock	2572
6.515.3.20	wait	2572
6.515.3.21	wait	2572
6.515.3.22	wait	2573
6.516	decaf::util::StlSet< E > Class Template Reference	2574
6.516.1	Detailed Description	2576

6.516.2 Constructor & Destructor Documentation	2576
6.516.2.1 StlSet	2577
6.516.2.2 StlSet	2577
6.516.2.3 StlSet	2577
6.516.2.4 ~StlSet	2577
6.516.3 Member Function Documentation	2577
6.516.3.1 add	2577
6.516.3.2 clear	2578
6.516.3.3 contains	2578
6.516.3.4 copy	2579
6.516.3.5 equals	2580
6.516.3.6 isEmpty	2580
6.516.3.7 iterator	2580
6.516.3.8 iterator	2580
6.516.3.9 remove	2580
6.516.3.10size	2581
6.517activemq::wireformat::stomp::StompCommandConstants Class Reference	2581
6.517.1 Field Documentation	2583
6.517.1.1 ABORT	2583
6.517.1.2 ACK	2583
6.517.1.3 ACK_AUTO	2583
6.517.1.4 ACK_CLIENT	2583
6.517.1.5 ACK_INDIVIDUAL	2583
6.517.1.6 BEGIN	2583
6.517.1.7 BYTES	2583
6.517.1.8 COMMIT	2583
6.517.1.9 CONNECT	2583
6.517.1.10CONNECTED	2583
6.517.1.11DISCONNECT	2584
6.517.1.12ERROR_CMD	2584
6.517.1.13HEADER_ACK	2584
6.517.1.14HEADER_CLIENT_ID	2584
6.517.1.15HEADER_CONSUMERPRIORITY	2584
6.517.1.16HEADER_CONTENTLENGTH	2584

6.517.1.17	HEADER_CORRELATIONID	2584
6.517.1.18	HEADER_DESTINATION	2584
6.517.1.19	HEADER_DISPATCH_ASYNC	2584
6.517.1.20	HEADER_EXCLUSIVE	2584
6.517.1.21	HEADER_EXPIRES	2584
6.517.1.22	HEADER_ID	2584
6.517.1.23	HEADER_JMSPRIORITY	2584
6.517.1.24	HEADER_LOGIN	2585
6.517.1.25	HEADER_MAXPENDINGMSGLIMIT	2585
6.517.1.26	HEADER_MESSAGE	2585
6.517.1.27	HEADER_MESSAGEID	2585
6.517.1.28	HEADER_NOLOCAL	2585
6.517.1.29	HEADER_OLDSUBSCRIPTIONNAME	2585
6.517.1.30	HEADER_PASSWORD	2585
6.517.1.31	HEADER_PERSISTENT	2585
6.517.1.32	HEADER_PREFETCHSIZE	2585
6.517.1.33	HEADER_RECEIPT_REQUIRED	2585
6.517.1.34	HEADER_RECEIPTID	2585
6.517.1.35	HEADER_REDELIVERED	2585
6.517.1.36	HEADER_REDELIVERYCOUNT	2586
6.517.1.37	HEADER_REPLYTO	2586
6.517.1.38	HEADER_REQUESTID	2586
6.517.1.39	HEADER_RESPONSEID	2586
6.517.1.40	HEADER_RETROACTIVE	2586
6.517.1.41	HEADER_SELECTOR	2586
6.517.1.42	HEADER_SESSIONID	2586
6.517.1.43	HEADER_SUBSCRIPTION	2586
6.517.1.44	HEADER_SUBSCRIPTIONNAME	2586
6.517.1.45	HEADER_TIMESTAMP	2586
6.517.1.46	HEADER_TRANSACTIONID	2586
6.517.1.47	HEADER_TRANSFORMATION	2586
6.517.1.48	HEADER_TRANSFORMATION_ERROR	2587
6.517.1.49	HEADER_TYPE	2587
6.517.1.50	MESSAGE	2587

6.517.1.51	QUEUE_PREFIX	2587
6.517.1.52	RECEIPT	2587
6.517.1.53	SEND	2587
6.517.1.54	SUBSCRIBE	2587
6.517.1.55	TEMPQUEUE_PREFIX	2587
6.517.1.56	TEMPTOPIC_PREFIX	2587
6.517.1.57	TEXT	2587
6.517.1.58	TOPIC_PREFIX	2587
6.517.1.59	UNSUBSCRIBE	2587
6.518	activemq::wireformat::stomp::StompFrame Class Reference	2587
6.518.1	Detailed Description	2589
6.518.2	Constructor & Destructor Documentation	2589
6.518.2.1	StompFrame	2589
6.518.2.2	~StompFrame	2589
6.518.3	Member Function Documentation	2589
6.518.3.1	clone	2589
6.518.3.2	copy	2589
6.518.3.3	fromStream	2589
6.518.3.4	getBody	2590
6.518.3.5	getBody	2590
6.518.3.6	getBodyLength	2590
6.518.3.7	getCommand	2590
6.518.3.8	getProperties	2590
6.518.3.9	getProperties	2591
6.518.3.10	getProperty	2591
6.518.3.11	hasProperty	2591
6.518.3.12	removeProperty	2591
6.518.3.13	setBody	2591
6.518.3.14	setCommand	2592
6.518.3.15	setProperty	2592
6.518.3.16	toStream	2592
6.519	activemq::wireformat::stomp::StompHelper Class Reference	2592
6.519.1	Detailed Description	2593
6.519.2	Constructor & Destructor Documentation	2594

6.519.2.1 StompHelper	2594
6.519.2.2 ~StompHelper	2594
6.519.3 Member Function Documentation	2594
6.519.3.1 convertConsumerId	2594
6.519.3.2 convertConsumerId	2594
6.519.3.3 convertDestination	2594
6.519.3.4 convertDestination	2595
6.519.3.5 convertMessageId	2595
6.519.3.6 convertMessageId	2595
6.519.3.7 convertProducerId	2596
6.519.3.8 convertProducerId	2596
6.519.3.9 convertProperties	2596
6.519.3.10convertProperties	2596
6.519.3.11convertTransactionId	2597
6.519.3.12convertTransactionId	2597
6.520activemq::wireformat::stomp::StompWireFormat Class Reference	2597
6.520.1 Constructor & Destructor Documentation	2598
6.520.1.1 StompWireFormat	2598
6.520.1.2 ~StompWireFormat	2598
6.520.2 Member Function Documentation	2598
6.520.2.1 createNegotiator	2598
6.520.2.2 getVersion	2599
6.520.2.3 hasNegotiator	2599
6.520.2.4 inReceive	2599
6.520.2.5 marshal	2599
6.520.2.6 setVersion	2600
6.520.2.7 unmarshal	2600
6.521activemq::wireformat::stomp::StompWireFormatFactory Class Reference	2601
6.521.1 Detailed Description	2601
6.521.2 Constructor & Destructor Documentation	2601
6.521.2.1 StompWireFormatFactory	2601
6.521.2.2 ~StompWireFormatFactory	2601
6.521.3 Member Function Documentation	2601
6.521.3.1 createWireFormat	2601

6.522cms::Stoppable Class Reference	2602
6.522.1 Detailed Description	2602
6.522.2 Constructor & Destructor Documentation	2602
6.522.2.1 ~Stoppable	2602
6.522.3 Member Function Documentation	2602
6.522.3.1 stop	2602
6.523decaf::util::logging::StreamHandler Class Reference	2603
6.523.1 Detailed Description	2604
6.523.2 Constructor & Destructor Documentation	2604
6.523.2.1 StreamHandler	2604
6.523.2.2 StreamHandler	2604
6.523.2.3 ~StreamHandler	2604
6.523.3 Member Function Documentation	2604
6.523.3.1 close	2604
6.523.3.2 close	2605
6.523.3.3 flush	2605
6.523.3.4 isLoggable	2605
6.523.3.5 publish	2605
6.523.3.6 setOutputStream	2606
6.524cms::StreamMessage Class Reference	2606
6.524.1 Detailed Description	2608
6.524.2 Constructor & Destructor Documentation	2608
6.524.2.1 ~StreamMessage	2608
6.524.3 Member Function Documentation	2608
6.524.3.1 readBoolean	2608
6.524.3.2 readByte	2609
6.524.3.3 readBytes	2609
6.524.3.4 readBytes	2610
6.524.3.5 readChar	2611
6.524.3.6 readDouble	2612
6.524.3.7 readFloat	2612
6.524.3.8 readInt	2613
6.524.3.9 readLong	2613
6.524.3.10readShort	2614

6.524.3.11	readString	2614
6.524.3.12	readUnsignedShort	2615
6.524.3.13	writeBoolean	2615
6.524.3.14	writeByte	2616
6.524.3.15	writeBytes	2616
6.524.3.16	writeBytes	2616
6.524.3.17	writeChar	2617
6.524.3.18	writeDouble	2617
6.524.3.19	writeFloat	2618
6.524.3.20	writeInt	2618
6.524.3.21	writeLong	2619
6.524.3.22	writeShort	2619
6.524.3.23	writeString	2619
6.524.3.24	writeUnsignedShort	2620
6.525	decaf::lang::String Class Reference	2620
6.525.1	Detailed Description	2622
6.525.2	Constructor & Destructor Documentation	2622
6.525.2.1	String	2622
6.525.2.2	String	2622
6.525.2.3	String	2623
6.525.2.4	String	2623
6.525.2.5	String	2623
6.525.2.6	~String	2624
6.525.3	Member Function Documentation	2624
6.525.3.1	charAt	2624
6.525.3.2	isEmpty	2624
6.525.3.3	length	2624
6.525.3.4	operator=	2624
6.525.3.5	operator=	2624
6.525.3.6	subSequence	2625
6.525.3.7	toString	2625
6.525.3.8	valueOf	2625
6.525.3.9	valueOf	2625
6.525.3.10	valueOf	2626

6.525.3.11valueOf	2626
6.525.3.12valueOf	2626
6.525.3.13valueOf	2627
6.525.3.14valueOf	2627
6.526decaf::util::StringTokenizer Class Reference	2627
6.526.1 Detailed Description	2628
6.526.2 Constructor & Destructor Documentation	2628
6.526.2.1 StringTokenizer	2628
6.526.2.2 ~StringTokenizer	2628
6.526.3 Member Function Documentation	2628
6.526.3.1 countTokens	2629
6.526.3.2 hasMoreTokens	2629
6.526.3.3 nextToken	2629
6.526.3.4 nextToken	2629
6.526.3.5 reset	2630
6.526.3.6 toArray	2630
6.527activemq::commands::SubscriptionInfo Class Reference	2631
6.527.1 Constructor & Destructor Documentation	2632
6.527.1.1 SubscriptionInfo	2632
6.527.1.2 ~SubscriptionInfo	2632
6.527.2 Member Function Documentation	2632
6.527.2.1 cloneDataStructure	2632
6.527.2.2 copyDataStructure	2632
6.527.2.3 equals	2633
6.527.2.4 getClientId	2633
6.527.2.5 getClientId	2633
6.527.2.6 getDataStructureType	2633
6.527.2.7 getDestination	2633
6.527.2.8 getDestination	2633
6.527.2.9 getSelector	2633
6.527.2.10getSelector	2633
6.527.2.11getSubscriptionName	2633
6.527.2.12getSubscriptionName	2634
6.527.2.13getSubscribedDestination	2634

6.527.2.14	getSubscribedDestination	2634
6.527.2.15	setClientId	2634
6.527.2.16	setDestination	2634
6.527.2.17	setSelector	2634
6.527.2.18	setSubscriptionName	2634
6.527.2.19	setSubscribedDestination	2634
6.527.2.20	toString	2634
6.527.3	Field Documentation	2634
6.527.3.1	clientId	2634
6.527.3.2	destination	2634
6.527.3.3	ID_SUBSCRIPTIONINFO	2635
6.527.3.4	selector	2635
6.527.3.5	subscriptionName	2635
6.527.3.6	subscribedDestination	2635
6.528	activemq::wireformat::openwire::marshal::generated::SubscriptionInfo-	
	Marshaller Class Reference	2635
6.528.1	Detailed Description	2636
6.528.2	Constructor & Destructor Documentation	2636
6.528.2.1	SubscriptionInfoMarshaller	2636
6.528.2.2	~SubscriptionInfoMarshaller	2636
6.528.3	Member Function Documentation	2636
6.528.3.1	createObject	2636
6.528.3.2	getDataStructureType	2636
6.528.3.3	looseMarshal	2637
6.528.3.4	looseUnmarshal	2637
6.528.3.5	tightMarshal1	2638
6.528.3.6	tightMarshal2	2638
6.528.3.7	tightUnmarshal	2638
6.529	decaf::util::concurrent::Synchronizable Class Reference	2639
6.529.1	Detailed Description	2640
6.529.2	Constructor & Destructor Documentation	2640
6.529.2.1	~Synchronizable	2640
6.529.3	Member Function Documentation	2640
6.529.3.1	lock	2640

6.529.3.2 notify	2641
6.529.3.3 notifyAll	2642
6.529.3.4 tryLock	2643
6.529.3.5 unlock	2645
6.529.3.6 wait	2646
6.529.3.7 wait	2647
6.529.3.8 wait	2648
6.530decaf::internal::util::concurrent::SynchronizableImpl Class Reference	2650
6.530.1 Detailed Description	2651
6.530.2 Constructor & Destructor Documentation	2651
6.530.2.1 SynchronizableImpl	2651
6.530.2.2 ~SynchronizableImpl	2651
6.530.3 Member Function Documentation	2651
6.530.3.1 lock	2651
6.530.3.2 notify	2651
6.530.3.3 notifyAll	2652
6.530.3.4 tryLock	2652
6.530.3.5 unlock	2652
6.530.3.6 wait	2652
6.530.3.7 wait	2653
6.530.3.8 wait	2653
6.531activemq::core::Synchronization Class Reference	2654
6.531.1 Detailed Description	2654
6.531.2 Constructor & Destructor Documentation	2654
6.531.2.1 ~Synchronization	2654
6.531.3 Member Function Documentation	2654
6.531.3.1 afterCommit	2654
6.531.3.2 afterRollback	2654
6.531.3.3 beforeEnd	2655
6.532decaf::util::concurrent::SynchronousQueue< E > Class Template - Reference	2655
6.532.1 Detailed Description	2657
6.532.2 Constructor & Destructor Documentation	2657
6.532.2.1 SynchronousQueue	2657

6.532.2.2 ~SynchronousQueue	2657
6.532.3 Member Function Documentation	2657
6.532.3.1 clear	2658
6.532.3.2 contains	2658
6.532.3.3 containsAll	2658
6.532.3.4 drainTo	2659
6.532.3.5 drainTo	2659
6.532.3.6 equals	2660
6.532.3.7 isEmpty	2660
6.532.3.8 iterator	2661
6.532.3.9 iterator	2661
6.532.3.10offer	2661
6.532.3.11offer	2662
6.532.3.12peek	2662
6.532.3.13poll	2662
6.532.3.14poll	2663
6.532.3.15put	2663
6.532.3.16remainingCapacity	2664
6.532.3.17remove	2664
6.532.3.18removeAll	2664
6.532.3.19retainAll	2664
6.532.3.20size	2664
6.532.3.21take	2664
6.532.3.22toArray	2665
6.533decaf::lang::System Class Reference	2665
6.533.1 Detailed Description	2667
6.533.2 Constructor & Destructor Documentation	2667
6.533.2.1 System	2667
6.533.2.2 ~System	2667
6.533.3 Member Function Documentation	2668
6.533.3.1 arraycopy	2668
6.533.3.2 arraycopy	2668
6.533.3.3 arraycopy	2669
6.533.3.4 arraycopy	2669

6.533.3.5 arraycopy	2670
6.533.3.6 arraycopy	2670
6.533.3.7 arraycopy	2670
6.533.3.8 arraycopy	2671
6.533.3.9 availableProcessors	2671
6.533.3.10clearProperty	2672
6.533.3.11currentTimeMillis	2672
6.533.3.12getenv	2672
6.533.3.13getenv	2673
6.533.3.14getProperties	2673
6.533.3.15getProperty	2673
6.533.3.16getProperty	2674
6.533.3.17nanoTime	2674
6.533.3.18setenv	2675
6.533.3.19setProperty	2675
6.533.3.20unsetenv	2675
6.533.4 Friends And Related Function Documentation	2676
6.533.4.1 decaf::lang::Runtime	2676
6.534activemq::threads::Task Class Reference	2676
6.534.1 Detailed Description	2676
6.534.2 Constructor & Destructor Documentation	2676
6.534.2.1 ~Task	2676
6.534.3 Member Function Documentation	2676
6.534.3.1 iterate	2676
6.535activemq::threads::TaskRunner Class Reference	2677
6.535.1 Constructor & Destructor Documentation	2677
6.535.1.1 ~TaskRunner	2677
6.535.2 Member Function Documentation	2677
6.535.2.1 shutdown	2677
6.535.2.2 shutdown	2678
6.535.2.3 wakeup	2678
6.536decaf::internal::net::tcp::TcpSocket Class Reference	2678
6.536.1 Detailed Description	2681
6.536.2 Constructor & Destructor Documentation	2681

6.536.2.1	TcpSocket	2681
6.536.2.2	~TcpSocket	2682
6.536.3	Member Function Documentation	2682
6.536.3.1	accept	2682
6.536.3.2	available	2682
6.536.3.3	bind	2682
6.536.3.4	checkResult	2682
6.536.3.5	close	2683
6.536.3.6	connect	2683
6.536.3.7	create	2683
6.536.3.8	getInputStream	2684
6.536.3.9	getLocalAddress	2684
6.536.3.10	getOption	2684
6.536.3.11	getOutputStream	2685
6.536.3.12	getSocketHandle	2685
6.536.3.13	isClosed	2685
6.536.3.14	isConnected	2685
6.536.3.15	listen	2685
6.536.3.16	read	2686
6.536.3.17	setOption	2686
6.536.3.18	shutdownInput	2687
6.536.3.19	shutdownOutput	2687
6.536.3.20	write	2687
6.537	decaf::internal::net::tcp::TcpSocketInputStream Class Reference	2688
6.537.1	Detailed Description	2689
6.537.2	Constructor & Destructor Documentation	2689
6.537.2.1	TcpSocketInputStream	2689
6.537.2.2	~TcpSocketInputStream	2689
6.537.3	Member Function Documentation	2689
6.537.3.1	available	2689
6.537.3.2	close	2690
6.537.3.3	doReadArrayBounded	2690
6.537.3.4	doReadByte	2690
6.537.3.5	skip	2690

6.538	decaf::internal::net::tcp::TcpSocketOutputStream Class Reference . . .	2691
6.538.1	Detailed Description	2692
6.538.2	Constructor & Destructor Documentation	2692
6.538.2.1	TcpSocketOutputStream	2692
6.538.2.2	~TcpSocketOutputStream	2692
6.538.3	Member Function Documentation	2692
6.538.3.1	close	2692
6.538.3.2	doWriteArrayBounded	2692
6.538.3.3	doWriteByte	2693
6.539	activemq::transport::tcp::TcpTransport Class Reference	2693
6.539.1	Detailed Description	2694
6.539.2	Constructor & Destructor Documentation	2694
6.539.2.1	TcpTransport	2694
6.539.2.2	~TcpTransport	2694
6.539.3	Member Function Documentation	2694
6.539.3.1	close	2694
6.539.3.2	configureSocket	2695
6.539.3.3	connect	2695
6.539.3.4	createSocket	2695
6.539.3.5	isClosed	2696
6.539.3.6	isConnected	2696
6.539.3.7	isFaultTolerant	2696
6.540	activemq::transport::tcp::TcpTransportFactory Class Reference	2696
6.540.1	Detailed Description	2697
6.540.2	Constructor & Destructor Documentation	2697
6.540.2.1	~TcpTransportFactory	2697
6.540.3	Member Function Documentation	2697
6.540.3.1	create	2697
6.540.3.2	createComposite	2698
6.540.3.3	doCreateComposite	2698
6.541	cms::TemporaryQueue Class Reference	2698
6.541.1	Detailed Description	2699
6.541.2	Constructor & Destructor Documentation	2699
6.541.2.1	~TemporaryQueue	2699

6.541.3 Member Function Documentation	2699
6.541.3.1 destroy	2699
6.541.3.2 getQueueName	2699
6.542cms::TemporaryTopic Class Reference	2700
6.542.1 Detailed Description	2700
6.542.2 Constructor & Destructor Documentation	2700
6.542.2.1 ~TemporaryTopic	2701
6.542.3 Member Function Documentation	2701
6.542.3.1 destroy	2701
6.542.3.2 getTopicName	2701
6.543cms::TextMessage Class Reference	2701
6.543.1 Detailed Description	2702
6.543.2 Constructor & Destructor Documentation	2702
6.543.2.1 ~TextMessage	2702
6.543.3 Member Function Documentation	2702
6.543.3.1 getText	2702
6.543.3.2 setText	2703
6.543.3.3 setText	2703
6.544decaf::lang::Thread Class Reference	2703
6.544.1 Detailed Description	2706
6.544.2 Member Enumeration Documentation	2707
6.544.2.1 State	2707
6.544.3 Constructor & Destructor Documentation	2707
6.544.3.1 Thread	2707
6.544.3.2 Thread	2707
6.544.3.3 Thread	2707
6.544.3.4 Thread	2708
6.544.3.5 ~Thread	2708
6.544.4 Member Function Documentation	2708
6.544.4.1 currentThread	2708
6.544.4.2 getId	2708
6.544.4.3 getName	2708
6.544.4.4 getPriority	2709
6.544.4.5 getState	2709

6.544.4.6	getUncaughtExceptionHandler	2709
6.544.4.7	isAlive	2709
6.544.4.8	isDaemon	2709
6.544.4.9	join	2710
6.544.4.10	join	2710
6.544.4.11	join	2710
6.544.4.12	run	2710
6.544.4.13	setDaemon	2711
6.544.4.14	setName	2711
6.544.4.15	setPriority	2711
6.544.4.16	setUncaughtExceptionHandler	2711
6.544.4.17	sleep	2712
6.544.4.18	sleep	2712
6.544.4.19	start	2712
6.544.4.20	toString	2713
6.544.4.21	yield	2713
6.544.5	Friends And Related Function Documentation	2713
6.544.5.1	decaf::lang::Runtime	2713
6.544.5.2	decaf::util::concurrent::locks::LockSupport	2713
6.544.6	Field Documentation	2713
6.544.6.1	MAX_PRIORITY	2713
6.544.6.2	MIN_PRIORITY	2713
6.544.6.3	NORM_PRIORITY	2713
6.545	decaf::util::concurrent::ThreadFactory Class Reference	2714
6.545.1	Detailed Description	2714
6.545.2	Constructor & Destructor Documentation	2714
6.545.2.1	~ThreadFactory	2714
6.545.3	Member Function Documentation	2714
6.545.3.1	newThread	2714
6.546	decaf::lang::ThreadGroup Class Reference	2715
6.546.1	Detailed Description	2715
6.546.2	Constructor & Destructor Documentation	2715
6.546.2.1	ThreadGroup	2715
6.546.2.2	~ThreadGroup	2715

6.547decaf::util::concurrent::ThreadPoolExecutor Class Reference	2715
6.547.1 Detailed Description	2719
6.547.2 Constructor & Destructor Documentation	2719
6.547.2.1 ThreadPoolExecutor	2719
6.547.2.2 ThreadPoolExecutor	2720
6.547.2.3 ThreadPoolExecutor	2720
6.547.2.4 ThreadPoolExecutor	2721
6.547.2.5 ~ThreadPoolExecutor	2722
6.547.3 Member Function Documentation	2722
6.547.3.1 afterExecute	2722
6.547.3.2 allowCoreThreadTimeout	2722
6.547.3.3 allowsCoreThreadTimeout	2723
6.547.3.4 awaitTermination	2723
6.547.3.5 beforeExecute	2724
6.547.3.6 execute	2724
6.547.3.7 getActiveCount	2724
6.547.3.8 getCompletedTaskCount	2725
6.547.3.9 getCorePoolSize	2725
6.547.3.10getKeepAliveTime	2725
6.547.3.11getLargestPoolSize	2725
6.547.3.12getMaximumPoolSize	2726
6.547.3.13getPoolSize	2726
6.547.3.14getQueue	2726
6.547.3.15getRejectedExecutionHandler	2726
6.547.3.16getTaskCount	2727
6.547.3.17getThreadFactory	2727
6.547.3.18sShutdown	2727
6.547.3.19sTerminated	2727
6.547.3.20sTerminating	2728
6.547.3.21prestartAllCoreThreads	2728
6.547.3.22prestartCoreThread	2728
6.547.3.23purge	2728
6.547.3.24remove	2729
6.547.3.25setCorePoolSize	2729

6.547.3.26	setKeepAliveTime	2729
6.547.3.27	setMaximumPoolSize	2730
6.547.3.28	setRejectedExecutionHandler	2730
6.547.3.29	setThreadFactory	2731
6.547.3.30	shutdown	2731
6.547.3.31	shutdownNow	2731
6.547.3.32	terminated	2732
6.547.4	Friends And Related Function Documentation	2732
6.547.4.1	ExecutorKernel	2732
6.548	decaf::lang::Throwable Class Reference	2732
6.548.1	Detailed Description	2733
6.548.2	Constructor & Destructor Documentation	2733
6.548.2.1	Throwable	2733
6.548.2.2	~Throwable	2733
6.548.3	Member Function Documentation	2733
6.548.3.1	clone	2733
6.548.3.2	getCause	2734
6.548.3.3	getMessage	2735
6.548.3.4	getStackTrace	2735
6.548.3.5	getStackTraceString	2735
6.548.3.6	initCause	2736
6.548.3.7	printStackTrace	2736
6.548.3.8	printStackTrace	2736
6.548.3.9	setMark	2736
6.549	decaf::util::concurrent::TimeoutException Class Reference	2737
6.549.1	Constructor & Destructor Documentation	2737
6.549.1.1	TimeoutException	2737
6.549.1.2	TimeoutException	2737
6.549.1.3	TimeoutException	2738
6.549.1.4	TimeoutException	2738
6.549.1.5	TimeoutException	2738
6.549.1.6	TimeoutException	2738
6.549.1.7	~TimeoutException	2739
6.549.2	Member Function Documentation	2739

6.549.2.1 clone	2739
6.550decaf::util::Timer Class Reference	2739
6.550.1 Detailed Description	2740
6.550.2 Constructor & Destructor Documentation	2741
6.550.2.1 Timer	2741
6.550.2.2 Timer	2741
6.550.2.3 ~Timer	2741
6.550.3 Member Function Documentation	2741
6.550.3.1 cancel	2741
6.550.3.2 purge	2742
6.550.3.3 schedule	2742
6.550.3.4 schedule	2742
6.550.3.5 schedule	2743
6.550.3.6 schedule	2743
6.550.3.7 schedule	2744
6.550.3.8 schedule	2745
6.550.3.9 schedule	2745
6.550.3.10schedule	2746
6.550.3.11scheduleAtFixedRate	2747
6.550.3.12scheduleAtFixedRate	2748
6.550.3.13scheduleAtFixedRate	2749
6.550.3.14scheduleAtFixedRate	2750
6.551decaf::util::TimerTask Class Reference	2751
6.551.1 Detailed Description	2751
6.551.2 Constructor & Destructor Documentation	2751
6.551.2.1 TimerTask	2751
6.551.2.2 ~TimerTask	2752
6.551.3 Member Function Documentation	2752
6.551.3.1 cancel	2752
6.551.3.2 getWhen	2752
6.551.3.3 isScheduled	2752
6.551.3.4 scheduledExecutionTime	2752
6.551.3.5 setScheduledTime	2753
6.551.4 Friends And Related Function Documentation	2753

6.551.4.1	decaf::internal::util::TimerTaskHeap	2753
6.551.4.2	Timer	2753
6.551.4.3	TimerImpl	2753
6.552	decaf::internal::util::TimerTaskHeap Class Reference	2753
6.552.1	Detailed Description	2754
6.552.2	Constructor & Destructor Documentation	2754
6.552.2.1	TimerTaskHeap	2754
6.552.2.2	~TimerTaskHeap	2754
6.552.3	Member Function Documentation	2754
6.552.3.1	adjustMinimum	2754
6.552.3.2	deleteIfCancelled	2754
6.552.3.3	find	2754
6.552.3.4	insert	2755
6.552.3.5	isEmpty	2755
6.552.3.6	peek	2755
6.552.3.7	remove	2755
6.552.3.8	reset	2755
6.552.3.9	size	2756
6.553	decaf::util::concurrent::TimeUnit Class Reference	2756
6.553.1	Detailed Description	2757
6.553.2	Constructor & Destructor Documentation	2758
6.553.2.1	TimeUnit	2758
6.553.2.2	~TimeUnit	2758
6.553.3	Member Function Documentation	2758
6.553.3.1	compareTo	2758
6.553.3.2	convert	2758
6.553.3.3	equals	2759
6.553.3.4	operator<	2759
6.553.3.5	operator==	2759
6.553.3.6	sleep	2759
6.553.3.7	timedJoin	2759
6.553.3.8	timedWait	2760
6.553.3.9	toDays	2760
6.553.3.10	toHours	2761

6.553.3.1 toMicros	2761
6.553.3.2 toMillis	2761
6.553.3.3 toMinutes	2762
6.553.3.4 toNanos	2762
6.553.3.5 toSeconds	2763
6.553.3.6 toString	2763
6.553.3.7 valueOf	2763
6.553.4 Field Documentation	2764
6.553.4.1 DAYS	2764
6.553.4.2 HOURS	2764
6.553.4.3 MICROSECONDS	2764
6.553.4.4 MILLISECONDS	2764
6.553.4.5 MINUTES	2764
6.553.4.6 NANOSECONDS	2764
6.553.4.7 SECONDS	2764
6.553.4.8 values	2764
6.554 cms::Topic Class Reference	2764
6.554.1 Detailed Description	2765
6.554.2 Constructor & Destructor Documentation	2765
6.554.2.1 ~Topic	2765
6.554.3 Member Function Documentation	2765
6.554.3.1 getTopicName	2765
6.555 activemq::state::Tracked Class Reference	2765
6.555.1 Constructor & Destructor Documentation	2766
6.555.1.1 Tracked	2766
6.555.1.2 Tracked	2766
6.555.1.3 ~Tracked	2766
6.555.2 Member Function Documentation	2766
6.555.2.1 isWaitingForResponse	2766
6.555.2.2 onResponse	2766
6.556 activemq::commands::TransactionId Class Reference	2766
6.556.1 Member Typedef Documentation	2767
6.556.1.1 COMPARATOR	2767
6.556.2 Constructor & Destructor Documentation	2767

6.556.2.1 TransactionId	2767
6.556.2.2 TransactionId	2767
6.556.2.3 ~TransactionId	2768
6.556.3 Member Function Documentation	2768
6.556.3.1 cloneDataStructure	2768
6.556.3.2 compareTo	2768
6.556.3.3 copyDataStructure	2768
6.556.3.4 equals	2768
6.556.3.5 equals	2769
6.556.3.6 getDataStructureType	2769
6.556.3.7 isLocalTransactionId	2769
6.556.3.8 isXATransactionId	2769
6.556.3.9 operator<	2769
6.556.3.10operator=	2769
6.556.3.11operator==	2769
6.556.3.12toString	2770
6.556.4 Field Documentation	2770
6.556.4.1 ID_TRANSACTIONID	2770
6.557activemq::wireformat::openwire::marshal::generated::TransactionId- Marshaller Class Reference	2770
6.557.1 Detailed Description	2771
6.557.2 Constructor & Destructor Documentation	2771
6.557.2.1 TransactionIdMarshaller	2771
6.557.2.2 ~TransactionIdMarshaller	2771
6.557.3 Member Function Documentation	2771
6.557.3.1 looseMarshal	2771
6.557.3.2 looseUnmarshal	2772
6.557.3.3 tightMarshal1	2772
6.557.3.4 tightMarshal2	2773
6.557.3.5 tightUnmarshal	2773
6.558activemq::commands::TransactionInfo Class Reference	2774
6.558.1 Constructor & Destructor Documentation	2775
6.558.1.1 TransactionInfo	2775
6.558.1.2 ~TransactionInfo	2775

6.558.2 Member Function Documentation	2775
6.558.2.1 cloneDataStructure	2775
6.558.2.2 copyDataStructure	2775
6.558.2.3 equals	2776
6.558.2.4 getConnectionId	2776
6.558.2.5 getConnectionId	2776
6.558.2.6 getDataStructureType	2776
6.558.2.7 getTransactionId	2776
6.558.2.8 getTransactionId	2776
6.558.2.9 getType	2776
6.558.2.10sTransactionInfo	2777
6.558.2.11setConnectionId	2777
6.558.2.12setTransactionId	2777
6.558.2.13setType	2777
6.558.2.14toString	2777
6.558.2.15visit	2777
6.558.3 Field Documentation	2777
6.558.3.1 connectionId	2778
6.558.3.2 ID_TRANSACTIONINFO	2778
6.558.3.3 transactionId	2778
6.558.3.4 type	2778
6.559activemq::wireformat::openwire::marshal::generated::TransactionInfo- Marshaller Class Reference	2778
6.559.1 Detailed Description	2779
6.559.2 Constructor & Destructor Documentation	2779
6.559.2.1 TransactionInfoMarshaller	2779
6.559.2.2 ~TransactionInfoMarshaller	2779
6.559.3 Member Function Documentation	2779
6.559.3.1 createObject	2779
6.559.3.2 getDataStructureType	2779
6.559.3.3 looseMarshal	2780
6.559.3.4 looseUnmarshal	2780
6.559.3.5 tightMarshal1	2781
6.559.3.6 tightMarshal2	2781

6.559.3.7 tightUnmarshal	2781
6.560cms::TransactionInProgressException Class Reference	2782
6.560.1 Detailed Description	2782
6.560.2 Constructor & Destructor Documentation	2783
6.560.2.1 TransactionInProgressException	2783
6.560.2.2 TransactionInProgressException	2783
6.560.2.3 TransactionInProgressException	2783
6.560.2.4 TransactionInProgressException	2783
6.560.2.5 TransactionInProgressException	2783
6.560.2.6 ~TransactionInProgressException	2783
6.561cms::TransactionRolledBackException Class Reference	2783
6.561.1 Detailed Description	2784
6.561.2 Constructor & Destructor Documentation	2784
6.561.2.1 TransactionRolledBackException	2784
6.561.2.2 TransactionRolledBackException	2784
6.561.2.3 TransactionRolledBackException	2784
6.561.2.4 TransactionRolledBackException	2784
6.561.2.5 TransactionRolledBackException	2784
6.561.2.6 ~TransactionRolledBackException	2784
6.562activemq::state::TransactionState Class Reference	2785
6.562.1 Constructor & Destructor Documentation	2785
6.562.1.1 TransactionState	2785
6.562.1.2 ~TransactionState	2785
6.562.2 Member Function Documentation	2785
6.562.2.1 addCommand	2785
6.562.2.2 addProducerState	2785
6.562.2.3 checkShutdown	2785
6.562.2.4 getCommands	2785
6.562.2.5 getld	2786
6.562.2.6 getPreparedResult	2786
6.562.2.7 getProducerStates	2786
6.562.2.8 isPrepared	2786
6.562.2.9 setPrepared	2786
6.562.2.10setPreparedResult	2786

6.562.2.1 shutdown	2786
6.562.2.2 toString	2786
6.563 decaf::internal::util::concurrent::Transferer< E > Class Template - Reference	2786
6.563.1 Detailed Description	2786
6.564 decaf::internal::util::concurrent::TransferQueue< E > Class Template - Reference	2787
6.564.1 Detailed Description	2787
6.564.2 Constructor & Destructor Documentation	2787
6.564.2.1 TransferQueue	2787
6.564.2.2 ~TransferQueue	2788
6.564.3 Member Function Documentation	2788
6.564.3.1 transfer	2788
6.564.3.2 transfer	2788
6.565 decaf::internal::util::concurrent::TransferStack< E > Class Template - Reference	2789
6.565.1 Constructor & Destructor Documentation	2789
6.565.1.1 TransferStack	2789
6.565.1.2 ~TransferStack	2789
6.565.2 Member Function Documentation	2789
6.565.2.1 transfer	2789
6.565.2.2 transfer	2790
6.566 activemq::transport::Transport Class Reference	2790
6.566.1 Detailed Description	2792
6.566.2 Constructor & Destructor Documentation	2792
6.566.2.1 ~Transport	2792
6.566.3 Member Function Documentation	2792
6.566.3.1 getRemoteAddress	2792
6.566.3.2 getTransportListener	2792
6.566.3.3 getWireFormat	2792
6.566.3.4 isClosed	2793
6.566.3.5 isConnected	2793
6.566.3.6 isFaultTolerant	2793
6.566.3.7 isReconnectSupported	2794
6.566.3.8 isUpdateURIsSupported	2794

6.566.3.9 narrow	2794
6.566.3.10 oneway	2795
6.566.3.11 reconnect	2795
6.566.3.12 request	2795
6.566.3.13 request	2796
6.566.3.14 setTransportListener	2797
6.566.3.15 setWireFormat	2797
6.566.3.16 start	2797
6.566.3.17 stop	2797
6.566.3.18 updateURIs	2798
6.567activemq::transport::TransportFactory Class Reference	2798
6.567.1 Detailed Description	2799
6.567.2 Constructor & Destructor Documentation	2799
6.567.2.1 ~TransportFactory	2799
6.567.3 Member Function Documentation	2799
6.567.3.1 create	2799
6.567.3.2 createComposite	2800
6.568activemq::transport::TransportFilter Class Reference	2800
6.568.1 Detailed Description	2802
6.568.2 Constructor & Destructor Documentation	2802
6.568.2.1 TransportFilter	2802
6.568.2.2 ~TransportFilter	2802
6.568.3 Member Function Documentation	2802
6.568.3.1 close	2802
6.568.3.2 fire	2803
6.568.3.3 fire	2803
6.568.3.4 getRemoteAddress	2803
6.568.3.5 getTransportListener	2803
6.568.3.6 getWireFormat	2804
6.568.3.7 isClosed	2804
6.568.3.8 isConnected	2804
6.568.3.9 isFaultTolerant	2804
6.568.3.10 isReconnectSupported	2805
6.568.3.11 isUpdateURIsSupported	2805

6.568.3.12	narrow	2805
6.568.3.13	onCommand	2805
6.568.3.14	oneway	2806
6.568.3.15	onException	2806
6.568.3.16	reconnect	2807
6.568.3.17	request	2807
6.568.3.18	request	2807
6.568.3.19	setTransportListener	2808
6.568.3.20	setWireFormat	2808
6.568.3.21	start	2808
6.568.3.22	stop	2809
6.568.3.23	transportInterrupted	2809
6.568.3.24	transportResumed	2809
6.568.3.25	updateURIs	2809
6.568.4	Field Documentation	2810
6.568.4.1	listener	2810
6.568.4.2	next	2810
6.569	activemq::transport::TransportListener Class Reference	2810
6.569.1	Detailed Description	2811
6.569.2	Constructor & Destructor Documentation	2811
6.569.2.1	~TransportListener	2811
6.569.3	Member Function Documentation	2811
6.569.3.1	onCommand	2811
6.569.3.2	onException	2811
6.569.3.3	transportInterrupted	2812
6.569.3.4	transportResumed	2812
6.570	activemq::transport::TransportRegistry Class Reference	2812
6.570.1	Detailed Description	2813
6.570.2	Constructor & Destructor Documentation	2813
6.570.2.1	~TransportRegistry	2813
6.570.3	Member Function Documentation	2813
6.570.3.1	findFactory	2813
6.570.3.2	getInstance	2813
6.570.3.3	getTransportNames	2814

6.570.3.4 registerFactory	2814
6.570.3.5 unregisterAllFactories	2814
6.570.3.6 unregisterFactory	2814
6.571tree_desc_s Struct Reference	2815
6.571.1 Field Documentation	2815
6.571.1.1 dyn_tree	2815
6.571.1.2 max_code	2815
6.571.1.3 stat_desc	2815
6.572decaf::lang::Thread::UncaughtExceptionHandler Class Reference	2815
6.572.1 Detailed Description	2816
6.572.2 Constructor & Destructor Documentation	2816
6.572.2.1 ~UncaughtExceptionHandler	2816
6.572.3 Member Function Documentation	2816
6.572.3.1 uncaughtException	2816
6.573decaf::net::UnknownHostException Class Reference	2816
6.573.1 Constructor & Destructor Documentation	2817
6.573.1.1 UnknownHostException	2817
6.573.1.2 UnknownHostException	2817
6.573.1.3 UnknownHostException	2817
6.573.1.4 UnknownHostException	2817
6.573.1.5 UnknownHostException	2818
6.573.1.6 UnknownHostException	2818
6.573.1.7 ~UnknownHostException	2818
6.573.2 Member Function Documentation	2818
6.573.2.1 clone	2818
6.574decaf::net::UnknownServiceException Class Reference	2819
6.574.1 Constructor & Destructor Documentation	2819
6.574.1.1 UnknownServiceException	2819
6.574.1.2 UnknownServiceException	2820
6.574.1.3 UnknownServiceException	2820
6.574.1.4 UnknownServiceException	2820
6.574.1.5 UnknownServiceException	2820
6.574.1.6 UnknownServiceException	2821
6.574.1.7 ~UnknownServiceException	2821

6.574.2 Member Function Documentation	2821
6.574.2.1 clone	2821
6.575decaf::io::UnsupportedEncodingException Class Reference	2821
6.575.1 Detailed Description	2822
6.575.2 Constructor & Destructor Documentation	2822
6.575.2.1 UnsupportedEncodingException	2822
6.575.2.2 UnsupportedEncodingException	2822
6.575.2.3 UnsupportedEncodingException	2823
6.575.2.4 UnsupportedEncodingException	2823
6.575.2.5 UnsupportedEncodingException	2823
6.575.2.6 UnsupportedEncodingException	2824
6.575.2.7 ~UnsupportedEncodingException	2824
6.575.3 Member Function Documentation	2824
6.575.3.1 clone	2824
6.576decaf::lang::exceptions::UnsupportedOperationException Class - Reference	2824
6.576.1 Constructor & Destructor Documentation	2825
6.576.1.1 UnsupportedOperationException	2825
6.576.1.2 UnsupportedOperationException	2825
6.576.1.3 UnsupportedOperationException	2825
6.576.1.4 UnsupportedOperationException	2826
6.576.1.5 UnsupportedOperationException	2826
6.576.1.6 UnsupportedOperationException	2826
6.576.1.7 ~UnsupportedOperationException	2827
6.576.2 Member Function Documentation	2827
6.576.2.1 clone	2827
6.577cms::UnsupportedOperationException Class Reference	2827
6.577.1 Detailed Description	2828
6.577.2 Constructor & Destructor Documentation	2828
6.577.2.1 UnsupportedOperationException	2828
6.577.2.2 UnsupportedOperationException	2828
6.577.2.3 UnsupportedOperationException	2828
6.577.2.4 UnsupportedOperationException	2828
6.577.2.5 UnsupportedOperationException	2828

6.577.2.6 ~UnsupportedOperationException	2828
6.578decaf::net::URI Class Reference	2828
6.578.1 Detailed Description	2830
6.578.2 Constructor & Destructor Documentation	2830
6.578.2.1 URI	2830
6.578.2.2 URI	2831
6.578.2.3 URI	2831
6.578.2.4 URI	2831
6.578.2.5 URI	2832
6.578.2.6 URI	2832
6.578.2.7 URI	2832
6.578.2.8 ~URI	2833
6.578.3 Member Function Documentation	2833
6.578.3.1 compareTo	2833
6.578.3.2 create	2833
6.578.3.3 equals	2834
6.578.3.4 getAuthority	2834
6.578.3.5 getFragment	2834
6.578.3.6 getHost	2834
6.578.3.7 getPath	2834
6.578.3.8 getPort	2834
6.578.3.9 getQuery	2834
6.578.3.10getRawAuthority	2835
6.578.3.11getRawFragment	2835
6.578.3.12getRawPath	2835
6.578.3.13getRawQuery	2835
6.578.3.14getRawSchemeSpecificPart	2836
6.578.3.15getRawUserInfo	2836
6.578.3.16getScheme	2836
6.578.3.17getSchemeSpecificPart	2836
6.578.3.18getUserInfo	2836
6.578.3.19sAbsolute	2837
6.578.3.20sOpaque	2837
6.578.3.21normalize	2837

6.578.3.22operator<	2838
6.578.3.23operator==	2838
6.578.3.24parseServerAuthority	2838
6.578.3.25relativize	2839
6.578.3.26resolve	2839
6.578.3.27resolve	2839
6.578.3.28toString	2840
6.578.3.29toURL	2840
6.579decaf::internal::net::URLEncoderDecoder Class Reference	2841
6.579.1 Constructor & Destructor Documentation	2842
6.579.1.1 URLEncoderDecoder	2842
6.579.1.2 ~URLEncoderDecoder	2842
6.579.2 Member Function Documentation	2842
6.579.2.1 decode	2842
6.579.2.2 encodeOthers	2842
6.579.2.3 quoteIllegal	2843
6.579.2.4 validate	2843
6.579.2.5 validateSimple	2843
6.580decaf::internal::net::URIHelper Class Reference	2844
6.580.1 Detailed Description	2845
6.580.2 Constructor & Destructor Documentation	2845
6.580.2.1 URIHelper	2845
6.580.2.2 URIHelper	2846
6.580.2.3 ~URIHelper	2846
6.580.3 Member Function Documentation	2846
6.580.3.1 isValidDomainName	2846
6.580.3.2 isValidHexChar	2846
6.580.3.3 isValidHost	2846
6.580.3.4 isValidIP4Word	2847
6.580.3.5 isValidIP6Address	2847
6.580.3.6 isValidIPv4Address	2847
6.580.3.7 parseAuthority	2848
6.580.3.8 parseURI	2848
6.580.3.9 validateAuthority	2849

6.580.3.10validateFragment	2849
6.580.3.11validatePath	2849
6.580.3.12validateQuery	2850
6.580.3.13validateScheme	2850
6.580.3.14validateSsp	2850
6.580.3.15validateUserinfo	2851
6.581activemq::transport::failover::URIPool Class Reference	2851
6.581.1 Constructor & Destructor Documentation	2852
6.581.1.1 URIPool	2852
6.581.1.2 URIPool	2852
6.581.1.3 ~URIPool	2852
6.581.2 Member Function Documentation	2852
6.581.2.1 addURI	2852
6.581.2.2 addURIs	2852
6.581.2.3 getURI	2853
6.581.2.4 isRandomize	2853
6.581.2.5 removeURI	2853
6.581.2.6 setRandomize	2853
6.582activemq::util::URISupport Class Reference	2854
6.582.1 Member Function Documentation	2854
6.582.1.1 createQueryString	2854
6.582.1.2 parseComposite	2855
6.582.1.3 parseQuery	2855
6.582.1.4 parseQuery	2856
6.582.1.5 parseURL	2856
6.583decaf::net::URISyntaxException Class Reference	2856
6.583.1 Constructor & Destructor Documentation	2857
6.583.1.1 URISyntaxException	2857
6.583.1.2 URISyntaxException	2857
6.583.1.3 URISyntaxException	2858
6.583.1.4 URISyntaxException	2858
6.583.1.5 URISyntaxException	2858
6.583.1.6 URISyntaxException	2858
6.583.1.7 URISyntaxException	2859

6.583.1.8 URISyntaxException	2859
6.583.1.9 ~URISyntaxException	2859
6.583.2 Member Function Documentation	2859
6.583.2.1 clone	2859
6.583.2.2 getIndex	2860
6.583.2.3 getInput	2860
6.583.2.4 getReason	2860
6.584decaf::internal::net::URIType Class Reference	2860
6.584.1 Detailed Description	2862
6.584.2 Constructor & Destructor Documentation	2862
6.584.2.1 URIType	2862
6.584.2.2 URIType	2862
6.584.2.3 ~URIType	2862
6.584.3 Member Function Documentation	2862
6.584.3.1 getAuthority	2862
6.584.3.2 getFragment	2862
6.584.3.3 getHost	2863
6.584.3.4 getPath	2863
6.584.3.5 getPort	2863
6.584.3.6 getQuery	2863
6.584.3.7 getScheme	2863
6.584.3.8 getSchemeSpecificPart	2864
6.584.3.9 getSource	2864
6.584.3.10getUserInfo	2864
6.584.3.11isAbsolute	2864
6.584.3.12sOpaque	2864
6.584.3.13sServerAuthority	2865
6.584.3.14sValid	2865
6.584.3.15setAbsolute	2865
6.584.3.16setAuthority	2865
6.584.3.17setFragment	2865
6.584.3.18setHost	2866
6.584.3.19setOpaque	2866
6.584.3.20setPath	2866

6.584.3.21	setPort	2866
6.584.3.22	setQuery	2866
6.584.3.23	setScheme	2867
6.584.3.24	setSchemeSpecificPart	2867
6.584.3.25	setServerAuthority	2867
6.584.3.26	setSource	2867
6.584.3.27	setUserInfo	2867
6.584.3.28	setValid	2868
6.585	decaf::net::URL Class Reference	2868
6.585.1	Detailed Description	2868
6.585.2	Constructor & Destructor Documentation	2870
6.585.2.1	URL	2870
6.585.2.2	URL	2870
6.585.2.3	~URL	2870
6.586	decaf::net::URLDecoder Class Reference	2870
6.586.1	Constructor & Destructor Documentation	2870
6.586.1.1	~URLDecoder	2870
6.586.2	Member Function Documentation	2870
6.586.2.1	decode	2871
6.587	decaf::net::URLEncoder Class Reference	2871
6.587.1	Constructor & Destructor Documentation	2871
6.587.1.1	~URLEncoder	2871
6.587.2	Member Function Documentation	2871
6.587.2.1	encode	2872
6.588	activemq::util::Usage Class Reference	2872
6.588.1	Constructor & Destructor Documentation	2873
6.588.1.1	~Usage	2873
6.588.2	Member Function Documentation	2873
6.588.2.1	decreaseUsage	2873
6.588.2.2	enqueueUsage	2873
6.588.2.3	increaseUsage	2873
6.588.2.4	isFull	2873
6.588.2.5	waitForSpace	2874
6.588.2.6	waitForSpace	2874

6.589decaf::io::UTFDataFormatException Class Reference	2874
6.589.1 Detailed Description	2875
6.589.2 Constructor & Destructor Documentation	2875
6.589.2.1 UTFDataFormatException	2875
6.589.2.2 UTFDataFormatException	2875
6.589.2.3 UTFDataFormatException	2875
6.589.2.4 UTFDataFormatException	2876
6.589.2.5 UTFDataFormatException	2876
6.589.2.6 UTFDataFormatException	2876
6.589.2.7 ~UTFDataFormatException	2876
6.589.3 Member Function Documentation	2876
6.589.3.1 clone	2877
6.590decaf::util::UUID Class Reference	2877
6.590.1 Detailed Description	2878
6.590.2 Constructor & Destructor Documentation	2879
6.590.2.1 UUID	2879
6.590.2.2 ~UUID	2879
6.590.3 Member Function Documentation	2879
6.590.3.1 clockSequence	2879
6.590.3.2 compareTo	2880
6.590.3.3 equals	2880
6.590.3.4 fromString	2880
6.590.3.5 getLeastSignificantBits	2880
6.590.3.6 getMostSignificantBits	2881
6.590.3.7 nameUUIDFromBytes	2881
6.590.3.8 nameUUIDFromBytes	2881
6.590.3.9 node	2881
6.590.3.10operator<	2882
6.590.3.11operator==	2882
6.590.3.12randomUUID	2882
6.590.3.13timestamp	2883
6.590.3.14toString	2883
6.590.3.15variant	2883
6.590.3.16version	2884

6.591activemq::wireformat::WireFormat Class Reference	2884
6.591.1 Detailed Description	2885
6.591.2 Constructor & Destructor Documentation	2885
6.591.2.1 ~WireFormat	2885
6.591.3 Member Function Documentation	2885
6.591.3.1 createNegotiator	2886
6.591.3.2 getVersion	2886
6.591.3.3 hasNegotiator	2886
6.591.3.4 inReceive	2887
6.591.3.5 marshal	2887
6.591.3.6 setVersion	2887
6.591.3.7 unmarshal	2888
6.592activemq::wireformat::WireFormatFactory Class Reference	2888
6.592.1 Detailed Description	2889
6.592.2 Constructor & Destructor Documentation	2889
6.592.2.1 ~WireFormatFactory	2889
6.592.3 Member Function Documentation	2889
6.592.3.1 createWireFormat	2889
6.593activemq::commands::WireFormatInfo Class Reference	2889
6.593.1 Constructor & Destructor Documentation	2892
6.593.1.1 WireFormatInfo	2892
6.593.1.2 ~WireFormatInfo	2892
6.593.2 Member Function Documentation	2892
6.593.2.1 afterUnmarshal	2892
6.593.2.2 beforeMarshal	2892
6.593.2.3 cloneDataStructure	2892
6.593.2.4 copyDataStructure	2892
6.593.2.5 equals	2892
6.593.2.6 getCacheSize	2893
6.593.2.7 getDataStructureType	2893
6.593.2.8 getMagic	2893
6.593.2.9 getMarshaledProperties	2893
6.593.2.10getMaxInactivityDuration	2894
6.593.2.11getMaxInactivityDurationInitalDelay	2894

6.593.2.12	getProperties	2894
6.593.2.13	getProperties	2894
6.593.2.14	getVersion	2894
6.593.2.15	sCacheEnabled	2895
6.593.2.16	sMarshalAware	2895
6.593.2.17	sSizePrefixDisabled	2895
6.593.2.18	sStackTraceEnabled	2895
6.593.2.19	sTcpNoDelayEnabled	2895
6.593.2.20	sTightEncodingEnabled	2896
6.593.2.21	isValid	2896
6.593.2.22	sWireFormatInfo	2896
6.593.2.23	setCacheEnabled	2896
6.593.2.24	setCacheSize	2896
6.593.2.25	setMagic	2897
6.593.2.26	setMarshaledProperties	2897
6.593.2.27	setMaxInactivityDuration	2897
6.593.2.28	setMaxInactivityDurationInitalDelay	2897
6.593.2.29	setProperties	2898
6.593.2.30	setSizePrefixDisabled	2898
6.593.2.31	setStackTraceEnabled	2898
6.593.2.32	setTcpNoDelayEnabled	2898
6.593.2.33	setTightEncodingEnabled	2898
6.593.2.34	setVersion	2899
6.593.2.35	toString	2899
6.593.2.36	visit	2899
6.593.3	Field Documentation	2899
6.593.3.1	ID_WIREFORMATINFO	2899
6.594	activemq::wireformat::openwire::marshal::generated::WireFormatInfo- Marshaller Class Reference	2900
6.594.1	Detailed Description	2901
6.594.2	Constructor & Destructor Documentation	2901
6.594.2.1	WireFormatInfoMarshaller	2901
6.594.2.2	~WireFormatInfoMarshaller	2901
6.594.3	Member Function Documentation	2901

6.594.3.1	createObject	2901
6.594.3.2	getDataStructureType	2901
6.594.3.3	looseMarshal	2902
6.594.3.4	looseUnmarshal	2902
6.594.3.5	tightMarshal1	2902
6.594.3.6	tightMarshal2	2903
6.594.3.7	tightUnmarshal	2903
6.595	activemq::wireformat::WireFormatNegotiator Class Reference	2904
6.595.1	Detailed Description	2904
6.595.2	Constructor & Destructor Documentation	2904
6.595.2.1	WireFormatNegotiator	2904
6.595.2.2	~WireFormatNegotiator	2905
6.596	activemq::wireformat::WireFormatRegistry Class Reference	2905
6.596.1	Detailed Description	2905
6.596.2	Constructor & Destructor Documentation	2906
6.596.2.1	~WireFormatRegistry	2906
6.596.3	Member Function Documentation	2906
6.596.3.1	findFactory	2906
6.596.3.2	getInstance	2906
6.596.3.3	getWireFormatNames	2906
6.596.3.4	registerFactory	2907
6.596.3.5	unregisterAllFactories	2907
6.596.3.6	unregisterFactory	2907
6.597	activemq::transport::inactivity::WriteChecker Class Reference	2907
6.597.1	Detailed Description	2908
6.597.2	Constructor & Destructor Documentation	2908
6.597.2.1	WriteChecker	2908
6.597.2.2	~WriteChecker	2908
6.597.3	Member Function Documentation	2908
6.597.3.1	run	2908
6.598	decaf::io::Writer Class Reference	2908
6.598.1	Constructor & Destructor Documentation	2910
6.598.1.1	Writer	2910
6.598.1.2	~Writer	2910

6.598.2 Member Function Documentation	2910
6.598.2.1 append	2910
6.598.2.2 append	2910
6.598.2.3 append	2911
6.598.2.4 doAppendChar	2911
6.598.2.5 doAppendCharSequence	2911
6.598.2.6 doAppendCharSequenceStartEnd	2911
6.598.2.7 doWriteArray	2911
6.598.2.8 doWriteArrayBounded	2911
6.598.2.9 doWriteChar	2912
6.598.2.10doWriteString	2912
6.598.2.11doWriteStringBounded	2912
6.598.2.12doWriteVector	2912
6.598.2.13write	2912
6.598.2.14write	2912
6.598.2.15write	2912
6.598.2.16write	2913
6.598.2.17write	2913
6.598.2.18write	2913
6.599decaf::security::auth::x500::X500Principal Class Reference	2914
6.599.1 Constructor & Destructor Documentation	2914
6.599.1.1 ~X500Principal	2914
6.599.2 Member Function Documentation	2914
6.599.2.1 getEncoded	2914
6.599.2.2 getName	2914
6.599.2.3 hashCode	2915
6.600decaf::security::cert::X509Certificate Class Reference	2915
6.600.1 Detailed Description	2915
6.600.2 Constructor & Destructor Documentation	2916
6.600.2.1 ~X509Certificate	2916
6.600.3 Member Function Documentation	2916
6.600.3.1 checkValidity	2916
6.600.3.2 checkValidity	2916
6.600.3.3 getBasicConstraints	2916

6.600.3.4	getIssuerUniqueID	2916
6.600.3.5	getIssuerX500Principal	2916
6.600.3.6	getKeyUsage	2916
6.600.3.7	getNotAfter	2916
6.600.3.8	getNotBefore	2916
6.600.3.9	getSigAlgName	2916
6.600.3.10	getSigAlgOID	2916
6.600.3.11	getSigAlgParams	2916
6.600.3.12	getSignature	2916
6.600.3.13	getSubjectUniqueID	2916
6.600.3.14	getSubjectX500Principal	2917
6.600.3.15	getTBSertificate	2917
6.600.3.16	getVersion	2917
6.601	cms::XAConnection Class Reference	2917
6.601.1	Detailed Description	2917
6.601.2	Constructor & Destructor Documentation	2917
6.601.2.1	~XAConnection	2918
6.601.3	Member Function Documentation	2918
6.601.3.1	createXASession	2918
6.602	cms::XAConnectionFactory Class Reference	2918
6.602.1	Detailed Description	2919
6.602.2	Constructor & Destructor Documentation	2919
6.602.2.1	~XAConnectionFactory	2919
6.602.3	Member Function Documentation	2919
6.602.3.1	createCMSXAConnectionFactory	2919
6.602.3.2	createXAConnection	2920
6.602.3.3	createXAConnection	2920
6.603	cms::XAException Class Reference	2921
6.603.1	Detailed Description	2923
6.603.2	Constructor & Destructor Documentation	2923
6.603.2.1	XAException	2923
6.603.2.2	XAException	2923
6.603.2.3	XAException	2923
6.603.2.4	XAException	2923

6.603.2.5 XAException	2923
6.603.2.6 XAException	2923
6.603.2.7 ~XAException	2923
6.603.3 Member Function Documentation	2923
6.603.3.1 getErrorCode	2923
6.603.3.2 setErrorCode	2924
6.603.4 Field Documentation	2924
6.603.4.1 XA_HEURCOM	2924
6.603.4.2 XA_HEURHAZ	2924
6.603.4.3 XA_HEURMIX	2924
6.603.4.4 XA_HEURRB	2924
6.603.4.5 XA_NOMIGRATE	2924
6.603.4.6 XA_RBBASE	2924
6.603.4.7 XA_RBCOMMFAIL	2924
6.603.4.8 XA_RBDEADLOCK	2925
6.603.4.9 XA_RBEND	2925
6.603.4.10XA_RBINTEGRITY	2925
6.603.4.11XA_RBOTHER	2925
6.603.4.12XA_RBPROTO	2925
6.603.4.13XA_RBROLLBACK	2925
6.603.4.14XA_RBTIMEOUT	2925
6.603.4.15XA_RBTRANSIENT	2925
6.603.4.16XA_RDONLY	2925
6.603.4.17XA_RETRY	2925
6.603.4.18XAER_ASYNC	2926
6.603.4.19XAER_DUPID	2926
6.603.4.20XAER_INVALID	2926
6.603.4.21XAER_NOTA	2926
6.603.4.22XAER_OUTSIDE	2926
6.603.4.23XAER_PROTO	2926
6.603.4.24XAER_RMERR	2926
6.603.4.25XAER_RMFAIL	2926
6.604cms::XAResource Class Reference	2926
6.604.1 Detailed Description	2928

6.604.2 Constructor & Destructor Documentation	2928
6.604.2.1 ~XAResource	2928
6.604.3 Member Function Documentation	2928
6.604.3.1 commit	2928
6.604.3.2 end	2929
6.604.3.3 forget	2930
6.604.3.4 getTransactionTimeout	2930
6.604.3.5 isSameRM	2930
6.604.3.6 prepare	2931
6.604.3.7 recover	2931
6.604.3.8 rollback	2932
6.604.3.9 setTransactionTimeout	2932
6.604.3.10start	2933
6.604.4 Field Documentation	2933
6.604.4.1 TMENDRSCAN	2933
6.604.4.2 TMFAIL	2933
6.604.4.3 TMJOIN	2933
6.604.4.4 TMNOFLAGS	2934
6.604.4.5 TMONEPHASE	2934
6.604.4.6 TMRESUME	2934
6.604.4.7 TMSTARTRSCAN	2934
6.604.4.8 TMSUCCESS	2934
6.604.4.9 TMSUSPEND	2934
6.604.4.10XA_OK	2934
6.604.4.11XA_RDONLY	2934
6.605cms::XASession Class Reference	2935
6.605.1 Detailed Description	2935
6.605.2 Constructor & Destructor Documentation	2935
6.605.2.1 ~XASession	2936
6.605.3 Member Function Documentation	2936
6.605.3.1 getXAResource	2936
6.606activemq::commands::XATransactionId Class Reference	2936
6.606.1 Member Typedef Documentation	2938
6.606.1.1 COMPARATOR	2938

6.606.2 Constructor & Destructor Documentation	2938
6.606.2.1 XATransactionId	2938
6.606.2.2 XATransactionId	2938
6.606.2.3 XATransactionId	2938
6.606.2.4 ~XATransactionId	2938
6.606.3 Member Function Documentation	2938
6.606.3.1 clone	2938
6.606.3.2 cloneDataStructure	2938
6.606.3.3 compareTo	2939
6.606.3.4 copyDataStructure	2939
6.606.3.5 equals	2939
6.606.3.6 equals	2939
6.606.3.7 equals	2939
6.606.3.8 getBranchQualifier	2939
6.606.3.9 getBranchQualifier	2940
6.606.3.10getBranchQualifier	2940
6.606.3.11getDataStructureType	2940
6.606.3.12getFormatId	2940
6.606.3.13getGlobalTransactionId	2940
6.606.3.14getGlobalTransactionId	2941
6.606.3.15getGlobalTransactionId	2941
6.606.3.16sXATransactionId	2941
6.606.3.17operator<	2941
6.606.3.18operator=	2941
6.606.3.19operator==	2941
6.606.3.20setBranchQualifier	2941
6.606.3.21setFormatId	2941
6.606.3.22setGlobalTransactionId	2942
6.606.3.23toString	2942
6.606.4 Field Documentation	2942
6.606.4.1 branchQualifier	2942
6.606.4.2 formatId	2942
6.606.4.3 globalTransactionId	2942
6.606.4.4 ID_XATRANSACTIONID	2942

6.607activemq::wireformat::openwire::marshal::generated::XATransactionId-	
Marshaller Class Reference	2942
6.607.1 Detailed Description	2943
6.607.2 Constructor & Destructor Documentation	2943
6.607.2.1 XATransactionIdMarshaller	2943
6.607.2.2 ~XATransactionIdMarshaller	2943
6.607.3 Member Function Documentation	2943
6.607.3.1 createObject	2944
6.607.3.2 getDataStructureType	2944
6.607.3.3 looseMarshal	2944
6.607.3.4 looseUnmarshal	2945
6.607.3.5 tightMarshal1	2945
6.607.3.6 tightMarshal2	2945
6.607.3.7 tightUnmarshal	2946
6.608cms::Xid Class Reference	2946
6.608.1 Detailed Description	2947
6.608.2 Constructor & Destructor Documentation	2948
6.608.2.1 Xid	2948
6.608.2.2 ~Xid	2948
6.608.3 Member Function Documentation	2948
6.608.3.1 clone	2948
6.608.3.2 equals	2948
6.608.3.3 getBranchQualifier	2948
6.608.3.4 getFormatId	2949
6.608.3.5 getGlobalTransactionId	2949
6.608.4 Field Documentation	2949
6.608.4.1 MAXBQUALSIZE	2949
6.608.4.2 MAXGTRIDSIZE	2950
6.609decaf::util::logging::XMLFormatter Class Reference	2950
6.609.1 Detailed Description	2950
6.609.2 Constructor & Destructor Documentation	2950
6.609.2.1 XMLFormatter	2950
6.609.2.2 ~XMLFormatter	2951
6.609.3 Member Function Documentation	2951

6.609.3.1	format	2951
6.609.3.2	getHead	2951
6.609.3.3	getTail	2951
6.610z_stream_s	Struct Reference	2952
6.610.1	Field Documentation	2952
6.610.1.1	adler	2952
6.610.1.2	avail_in	2952
6.610.1.3	avail_out	2952
6.610.1.4	data_type	2952
6.610.1.5	msg	2952
6.610.1.6	next_in	2952
6.610.1.7	next_out	2952
6.610.1.8	opaque	2952
6.610.1.9	reserved	2952
6.610.1.10	state	2952
6.610.1.11	total_in	2953
6.610.1.12	total_out	2953
6.610.1.13	zalloc	2953
6.610.1.14	zfree	2953
6.611decaf::util::zip::ZipException	Class Reference	2953
6.611.1	Constructor & Destructor Documentation	2953
6.611.1.1	ZipException	2953
6.611.1.2	ZipException	2954
6.611.1.3	ZipException	2954
6.611.1.4	ZipException	2954
6.611.1.5	ZipException	2954
6.611.1.6	ZipException	2955
6.611.1.7	~ZipException	2955
6.611.2	Member Function Documentation	2955
6.611.2.1	clone	2955
7	File Documentation	2957
7.1	src/main/activemq/cmsutil/CachedConsumer.h File Reference	2957
7.2	src/main/activemq/cmsutil/CachedProducer.h File Reference	2957

7.3	src/main/activemq/cmsutil/CmsAccessor.h File Reference	2958
7.4	src/main/activemq/cmsutil/CmsDestinationAccessor.h File Reference	2958
7.5	src/main/activemq/cmsutil/CmsTemplate.h File Reference	2959
7.6	src/main/activemq/cmsutil/DestinationResolver.h File Reference	2959
7.7	src/main/activemq/cmsutil/DynamicDestinationResolver.h File Reference	2960
7.8	src/main/activemq/cmsutil/MessageCreator.h File Reference	2960
7.9	src/main/activemq/cmsutil/PooledSession.h File Reference	2961
7.10	src/main/activemq/cmsutil/ProducerCallback.h File Reference	2961
7.11	src/main/activemq/cmsutil/ResourceLifecycleManager.h File Reference	2962
7.12	src/main/decaf/internal/util/ResourceLifecycleManager.h File Reference	2962
7.13	src/main/activemq/cmsutil/SessionCallback.h File Reference	2962
7.14	src/main/activemq/cmsutil/SessionPool.h File Reference	2963
7.15	src/main/activemq/commands/ActiveMQBlobMessage.h File Reference	2963
7.16	src/main/activemq/commands/ActiveMQBytesMessage.h File Reference	2964
7.17	src/main/activemq/commands/ActiveMQDestination.h File Reference	2964
7.18	src/main/activemq/commands/ActiveMQMapMessage.h File Reference	2965
7.19	src/main/activemq/commands/ActiveMQMessage.h File Reference	2965
7.20	src/main/activemq/commands/ActiveMQMessageTemplate.h File Reference	2966
7.21	src/main/activemq/commands/ActiveMQObjectMessage.h File Reference	2966
7.22	src/main/activemq/commands/ActiveMQQueue.h File Reference	2967
7.23	src/main/activemq/commands/ActiveMQStreamMessage.h File Reference	2967
7.24	src/main/activemq/commands/ActiveMQTempDestination.h File Reference	2968
7.25	src/main/activemq/commands/ActiveMQTempQueue.h File Reference	2968
7.26	src/main/activemq/commands/ActiveMQTempTopic.h File Reference	2969
7.27	src/main/activemq/commands/ActiveMQTextMessage.h File Reference	2969
7.28	src/main/activemq/commands/ActiveMQTopic.h File Reference	2970
7.29	src/main/activemq/commands/BaseCommand.h File Reference	2970
7.30	src/main/activemq/commands/BaseDataStructure.h File Reference	2971
7.31	src/main/activemq/commands/BooleanExpression.h File Reference	2971
7.32	src/main/activemq/commands/BrokerError.h File Reference	2971
7.33	src/main/activemq/commands/BrokerId.h File Reference	2972
7.34	src/main/activemq/commands/BrokerInfo.h File Reference	2972

7.35	src/main/activemq/commands/Command.h File Reference	2973
7.36	src/main/activemq/commands/ConnectionControl.h File Reference	2973
7.37	src/main/activemq/commands/ConnectionError.h File Reference	2974
7.38	src/main/activemq/commands/ConnectionId.h File Reference	2974
7.39	src/main/activemq/commands/ConnectionInfo.h File Reference	2974
7.40	src/main/activemq/commands/ConsumerControl.h File Reference	2975
7.41	src/main/activemq/commands/ConsumerId.h File Reference	2975
7.42	src/main/activemq/commands/ConsumerInfo.h File Reference	2976
7.43	src/main/activemq/commands/ControlCommand.h File Reference	2976
7.44	src/main/activemq/commands/DataArrayResponse.h File Reference	2977
7.45	src/main/activemq/commands/DataResponse.h File Reference	2977
7.46	src/main/activemq/commands/DataSet.h File Reference	2978
7.47	src/main/activemq/commands/DestinationInfo.h File Reference	2978
7.48	src/main/activemq/commands/DiscoveryEvent.h File Reference	2978
7.49	src/main/activemq/commands/ExceptionResponse.h File Reference	2979
7.50	src/main/activemq/commands/FlushCommand.h File Reference	2979
7.51	src/main/activemq/commands/IntegerResponse.h File Reference	2980
7.52	src/main/activemq/commands/JournalQueueAck.h File Reference	2980
7.53	src/main/activemq/commands/JournalTopicAck.h File Reference	2981
7.54	src/main/activemq/commands/JournalTrace.h File Reference	2981
7.55	src/main/activemq/commands/JournalTransaction.h File Reference	2982
7.56	src/main/activemq/commands/KeepAliveInfo.h File Reference	2982
7.57	src/main/activemq/commands/LastPartialCommand.h File Reference	2982
7.58	src/main/activemq/commands/LocalTransactionId.h File Reference	2983
7.59	src/main/activemq/commands/Message.h File Reference	2983
7.60	src/main/cms/Message.h File Reference	2984
7.61	src/main/activemq/commands/MessageAck.h File Reference	2984
7.62	src/main/activemq/commands/MessageDispatch.h File Reference	2985
7.63	src/main/activemq/commands/MessageDispatchNotification.h File Reference	2985
7.64	src/main/activemq/commands/MessageId.h File Reference	2986
7.65	src/main/activemq/commands/MessagePull.h File Reference	2986
7.66	src/main/activemq/commands/NetworkBridgeFilter.h File Reference	2987
7.67	src/main/activemq/commands/PartialCommand.h File Reference	2987

7.68	src/main/activemq/commands/ProducerAck.h File Reference	2988
7.69	src/main/activemq/commands/ProducerId.h File Reference	2988
7.70	src/main/activemq/commands/ProducerInfo.h File Reference	2988
7.71	src/main/activemq/commands/RemoveInfo.h File Reference	2989
7.72	src/main/activemq/commands/RemoveSubscriptionInfo.h File Reference	2989
7.73	src/main/activemq/commands/ReplayCommand.h File Reference	2990
7.74	src/main/activemq/commands/Response.h File Reference	2990
7.75	src/main/activemq/commands/SessionId.h File Reference	2991
7.76	src/main/activemq/commands/SessionInfo.h File Reference	2991
7.77	src/main/activemq/commands/ShutdownInfo.h File Reference	2992
7.78	src/main/activemq/commands/SubscriptionInfo.h File Reference	2992
7.79	src/main/activemq/commands/TransactionId.h File Reference	2992
7.80	src/main/activemq/commands/TransactionInfo.h File Reference	2993
7.81	src/main/activemq/commands/WireFormatInfo.h File Reference	2993
7.82	src/main/activemq/commands/XATransactionId.h File Reference	2994
7.83	src/main/activemq/core/ActiveMQAckHandler.h File Reference	2994
7.84	src/main/activemq/core/ActiveMQConnection.h File Reference	2995
7.85	src/main/activemq/core/ActiveMQConnectionFactory.h File Reference . .	2995
7.86	src/main/activemq/core/ActiveMQConnectionMetaData.h File Reference	2996
7.87	src/main/activemq/core/ActiveMQConstants.h File Reference	2996
7.88	src/main/activemq/core/ActiveMQConsumer.h File Reference	2997
7.89	src/main/activemq/core/ActiveMQProducer.h File Reference	2997
7.90	src/main/activemq/core/ActiveMQQueueBrowser.h File Reference	2998
7.91	src/main/activemq/core/ActiveMQSession.h File Reference	2998
7.92	src/main/activemq/core/ActiveMQSessionExecutor.h File Reference . . .	2999
7.93	src/main/activemq/core/ActiveMQTransactionContext.h File Reference .	3000
7.94	src/main/activemq/core/ActiveMQXAConnection.h File Reference	3000
7.95	src/main/activemq/core/ActiveMQXAConnectionFactory.h File Reference	3001
7.96	src/main/activemq/core/ActiveMQXASession.h File Reference	3001
7.97	src/main/activemq/core/DispatchData.h File Reference	3001
7.98	src/main/activemq/core/Dispatcher.h File Reference	3002
7.99	src/main/activemq/core/FifoMessageDispatchChannel.h File Reference .	3002
7.100	src/main/activemq/core/MessageDispatchChannel.h File Reference . . .	3003
7.101	src/main/activemq/core/policies/DefaultPrefetchPolicy.h File Reference .	3003

7.102src/main/activemq/core/policies/DefaultRedeliveryPolicy.h File Reference	3004
7.103src/main/activemq/core/PrefetchPolicy.h File Reference	3004
7.104src/main/activemq/core/RedeliveryPolicy.h File Reference	3005
7.105src/main/activemq/core/SimplePriorityMessageDispatchChannel.h File Reference	3005
7.106src/main/activemq/core/Synchronization.h File Reference	3005
7.107src/main/activemq/exceptions/ActiveMQException.h File Reference	3006
7.108src/main/activemq/exceptions/BrokerException.h File Reference	3006
7.109src/main/activemq/exceptions/ConnectionFailedException.h File Reference	3007
7.110src/main/activemq/exceptions/ExceptionDefines.h File Reference	3007
7.110.1 Define Documentation	3008
7.110.1.1 AMQ_CATCH_EXCEPTION_CONVERT	3008
7.110.1.2 AMQ_CATCH_NOTHROW	3008
7.110.1.3 AMQ_CATCH_RETHROW	3008
7.110.1.4 AMQ_CATCHALL_NOTHROW	3009
7.110.1.5 AMQ_CATCHALL_THROW	3009
7.111src/main/decaf/lang/exceptions/ExceptionDefines.h File Reference	3009
7.111.1 Define Documentation	3010
7.111.1.1 DECAF_CATCH_EXCEPTION_CONVERT	3010
7.111.1.2 DECAF_CATCH_NOTHROW	3010
7.111.1.3 DECAF_CATCH_RETHROW	3011
7.111.1.4 DECAF_CATCHALL_NOTHROW	3011
7.111.1.5 DECAF_CATCHALL_THROW	3011
7.112src/main/activemq/io/LoggingInputStream.h File Reference	3012
7.113src/main/activemq/io/LoggingOutputStream.h File Reference	3012
7.114src/main/activemq/library/ActiveMQCPP.h File Reference	3012
7.115src/main/activemq/state/CommandVisitor.h File Reference	3013
7.116src/main/activemq/state/CommandVisitorAdapter.h File Reference	3013
7.117src/main/activemq/state/ConnectionState.h File Reference	3014
7.118src/main/activemq/state/ConnectionStateTracker.h File Reference	3015
7.119src/main/activemq/state/ConsumerState.h File Reference	3015
7.120src/main/activemq/state/ProducerState.h File Reference	3016
7.121src/main/activemq/state/SessionState.h File Reference	3016

7.122src/main/activemq/state/Tracked.h File Reference	3017
7.123src/main/activemq/state/TransactionState.h File Reference	3017
7.124src/main/activemq/threads/CompositeTask.h File Reference	3018
7.125src/main/activemq/threads/CompositeTaskRunner.h File Reference . . .	3018
7.126src/main/activemq/threads/DedicatedTaskRunner.h File Reference . . .	3019
7.127src/main/activemq/threads/Scheduler.h File Reference	3019
7.128src/main/activemq/threads/SchedulerTimerTask.h File Reference	3020
7.129src/main/activemq/threads/Task.h File Reference	3020
7.130src/main/activemq/threads/TaskRunner.h File Reference	3020
7.131src/main/activemq/transport/AbstractTransportFactory.h File Reference .	3021
7.132src/main/activemq/transport/CompositeTransport.h File Reference	3021
7.133src/main/activemq/transport/correlator/FutureResponse.h File Reference	3022
7.134src/main/activemq/transport/correlator/ResponseCorrelator.h File - Reference	3022
7.135src/main/activemq/transport/DefaultTransportListener.h File Reference .	3023
7.136src/main/activemq/transport/failover/BackupTransport.h File Reference .	3024
7.137src/main/activemq/transport/failover/BackupTransportPool.h File - Reference	3024
7.138src/main/activemq/transport/failover/CloseTransportsTask.h File - Reference	3025
7.139src/main/activemq/transport/failover/FailoverTransport.h File Reference .	3025
7.140src/main/activemq/transport/failover/FailoverTransportFactory.h File - Reference	3026
7.141src/main/activemq/transport/failover/FailoverTransportListener.h File Reference	3026
7.142src/main/activemq/transport/failover/URIPool.h File Reference	3027
7.143src/main/activemq/transport/inactivity/InactivityMonitor.h File Reference .	3027
7.144src/main/activemq/transport/inactivity/ReadChecker.h File Reference . .	3028
7.145src/main/activemq/transport/inactivity/WriteChecker.h File Reference . .	3028
7.146src/main/activemq/transport/IOTransport.h File Reference	3029
7.147src/main/activemq/transport/logging/LoggingTransport.h File Reference .	3029
7.148src/main/activemq/transport/mock/InternalCommandListener.h File - Reference	3030
7.149src/main/activemq/transport/mock/MockTransport.h File Reference . . .	3030
7.150src/main/activemq/transport/mock/MockTransportFactory.h File Reference	3031
7.151src/main/activemq/transport/mock/ResponseBuilder.h File Reference . .	3032

7.152src/main/activemq/transport/tcp/SslTransport.h File Reference	3032
7.153src/main/activemq/transport/tcp/SslTransportFactory.h File Reference	3033
7.154src/main/activemq/transport/tcp/TcpTransport.h File Reference	3033
7.155src/main/activemq/transport/tcp/TcpTransportFactory.h File Reference	3034
7.156src/main/activemq/transport/Transport.h File Reference	3034
7.157src/main/activemq/transport/TransportFactory.h File Reference	3035
7.158src/main/activemq/transport/TransportFilter.h File Reference	3035
7.159src/main/activemq/transport/TransportListener.h File Reference	3036
7.160src/main/activemq/transport/TransportRegistry.h File Reference	3036
7.161src/main/activemq/util/ActiveMQProperties.h File Reference	3037
7.162src/main/activemq/util/CMSExceptionSupport.h File Reference	3037
7.162.1 Define Documentation	3038
7.162.1.1 AMQ_CATCH_ALL_THROW_CMSEXCEPTION	3038
7.163src/main/activemq/util/CompositeData.h File Reference	3039
7.164src/main/activemq/util/Config.h File Reference	3039
7.164.1 Define Documentation	3039
7.164.1.1 AMQCPP_API	3039
7.164.1.2 HAVE_PTHREAD_H	3039
7.164.1.3 HAVE_UUID_T	3039
7.164.1.4 HAVE_UUID_UUID_H	3039
7.165src/main/cms/Config.h File Reference	3039
7.165.1 Define Documentation	3040
7.165.1.1 CMS_API	3040
7.166src/main/decaf/util/Config.h File Reference	3040
7.166.1 Define Documentation	3040
7.166.1.1 DECAF_API	3040
7.166.1.2 DECAF_UNUSED	3040
7.166.1.3 HAVE_PTHREAD_H	3040
7.166.1.4 HAVE_UUID_T	3040
7.166.1.5 HAVE_UUID_UUID_H	3040
7.167src/main/activemq/util/IdGenerator.h File Reference	3040
7.168src/main/activemq/util/LongSequenceGenerator.h File Reference	3041
7.169src/main/activemq/util/MarshallingSupport.h File Reference	3041
7.170src/main/activemq/util/MemoryUsage.h File Reference	3041

7.171src/main/activemq/util/PrimitiveList.h File Reference	3042
7.172src/main/activemq/util/PrimitiveMap.h File Reference	3042
7.173src/main/activemq/util/PrimitiveValueConverter.h File Reference	3043
7.174src/main/activemq/util/PrimitiveValueNode.h File Reference	3043
7.175src/main/activemq/util/Service.h File Reference	3044
7.176src/main/activemq/util/ServiceListener.h File Reference	3044
7.177src/main/activemq/util/ServiceStopper.h File Reference	3045
7.178src/main/activemq/util/ServiceSupport.h File Reference	3045
7.178.1 Define Documentation	3046
7.178.1.1 SERVICESUPPORT_H_	3046
7.179src/main/activemq/util/URISupport.h File Reference	3046
7.180src/main/activemq/util/Usage.h File Reference	3046
7.181src/main/activemq/wireformat/MarshalAware.h File Reference	3046
7.182src/main/activemq/wireformat/openwire/marshal/BaseDataStream- Marshaller.h File Reference	3047
7.183src/main/activemq/wireformat/openwire/marshal/DataStreamMarshaller.h File Reference	3047
7.184src/main/activemq/wireformat/openwire/marshal/generated/ActiveMQ- BlobMessageMarshaller.h File Reference	3048
7.185src/main/activemq/wireformat/openwire/marshal/generated/ActiveMQ- BytesMessageMarshaller.h File Reference	3049
7.186src/main/activemq/wireformat/openwire/marshal/generated/ActiveMQ- DestinationMarshaller.h File Reference	3049
7.187src/main/activemq/wireformat/openwire/marshal/generated/ActiveMQ- MapMessageMarshaller.h File Reference	3050
7.188src/main/activemq/wireformat/openwire/marshal/generated/ActiveMQ- MessageMarshaller.h File Reference	3051
7.189src/main/activemq/wireformat/openwire/marshal/generated/ActiveMQ- ObjectMessageMarshaller.h File Reference	3051
7.190src/main/activemq/wireformat/openwire/marshal/generated/ActiveMQ- QueueMarshaller.h File Reference	3052
7.191src/main/activemq/wireformat/openwire/marshal/generated/ActiveMQ- StreamMessageMarshaller.h File Reference	3052
7.192src/main/activemq/wireformat/openwire/marshal/generated/ActiveMQ- TempDestinationMarshaller.h File Reference	3053
7.193src/main/activemq/wireformat/openwire/marshal/generated/ActiveMQ- TempQueueMarshaller.h File Reference	3054

7.194	src/main/activemq/wireformat/openwire/marshal/generated/ActiveMQ- TempTopicMarshaller.h File Reference	3054
7.195	src/main/activemq/wireformat/openwire/marshal/generated/ActiveMQ- TextMessageMarshaller.h File Reference	3055
7.196	src/main/activemq/wireformat/openwire/marshal/generated/ActiveMQ- TopicMarshaller.h File Reference	3056
7.197	src/main/activemq/wireformat/openwire/marshal/generated/Base- CommandMarshaller.h File Reference	3056
7.198	src/main/activemq/wireformat/openwire/marshal/generated/BrokerId- Marshaller.h File Reference	3057
7.199	src/main/activemq/wireformat/openwire/marshal/generated/BrokerInfo- Marshaller.h File Reference	3058
7.200	src/main/activemq/wireformat/openwire/marshal/generated/Connection- ControlMarshaller.h File Reference	3058
7.201	src/main/activemq/wireformat/openwire/marshal/generated/Connection- ErrorMarshaller.h File Reference	3059
7.202	src/main/activemq/wireformat/openwire/marshal/generated/Connection- IdMarshaller.h File Reference	3060
7.203	src/main/activemq/wireformat/openwire/marshal/generated/Connection- InfoMarshaller.h File Reference	3060
7.204	src/main/activemq/wireformat/openwire/marshal/generated/Consumer- ControlMarshaller.h File Reference	3061
7.205	src/main/activemq/wireformat/openwire/marshal/generated/Consumer- IdMarshaller.h File Reference	3061
7.206	src/main/activemq/wireformat/openwire/marshal/generated/Consumer- InfoMarshaller.h File Reference	3062
7.207	src/main/activemq/wireformat/openwire/marshal/generated/Control- CommandMarshaller.h File Reference	3063
7.208	src/main/activemq/wireformat/openwire/marshal/generated/DataArray- ResponseMarshaller.h File Reference	3063
7.209	src/main/activemq/wireformat/openwire/marshal/generated/Data- ResponseMarshaller.h File Reference	3064
7.210	src/main/activemq/wireformat/openwire/marshal/generated/Destination- InfoMarshaller.h File Reference	3065
7.211	src/main/activemq/wireformat/openwire/marshal/generated/Discovery- EventMarshaller.h File Reference	3065
7.212	src/main/activemq/wireformat/openwire/marshal/generated/Exception- ResponseMarshaller.h File Reference	3066
7.213	src/main/activemq/wireformat/openwire/marshal/generated/Flush- CommandMarshaller.h File Reference	3066

7.214src/main/activemq/wireformat/openwire/marshal/generated/Integer-ResponseMarshaller.h File Reference	3067
7.215src/main/activemq/wireformat/openwire/marshal/generated/QueueAckMarshaller.h File Reference	3068
7.216src/main/activemq/wireformat/openwire/marshal/generated/QueueTopicAckMarshaller.h File Reference	3068
7.217src/main/activemq/wireformat/openwire/marshal/generated/QueueTraceMarshaller.h File Reference	3069
7.218src/main/activemq/wireformat/openwire/marshal/generated/QueueTransactionMarshaller.h File Reference	3070
7.219src/main/activemq/wireformat/openwire/marshal/generated/KeepAlive-InfoMarshaller.h File Reference	3070
7.220src/main/activemq/wireformat/openwire/marshal/generated/LastPartial-CommandMarshaller.h File Reference	3071
7.221src/main/activemq/wireformat/openwire/marshal/generated/Local-TransactionIdMarshaller.h File Reference	3071
7.222src/main/activemq/wireformat/openwire/marshal/generated/Marshaller-Factory.h File Reference	3072
7.223src/main/activemq/wireformat/openwire/marshal/generated/Message-AckMarshaller.h File Reference	3073
7.224src/main/activemq/wireformat/openwire/marshal/generated/Message-DispatchMarshaller.h File Reference	3073
7.225src/main/activemq/wireformat/openwire/marshal/generated/Message-DispatchNotificationMarshaller.h File Reference	3074
7.226src/main/activemq/wireformat/openwire/marshal/generated/MessageId-Marshaller.h File Reference	3074
7.227src/main/activemq/wireformat/openwire/marshal/generated/Message-Marshaller.h File Reference	3075
7.228src/main/activemq/wireformat/openwire/marshal/generated/Message-PullMarshaller.h File Reference	3076
7.229src/main/activemq/wireformat/openwire/marshal/generated/Network-BridgeFilterMarshaller.h File Reference	3076
7.230src/main/activemq/wireformat/openwire/marshal/generated/Partial-CommandMarshaller.h File Reference	3077
7.231src/main/activemq/wireformat/openwire/marshal/generated/Producer-AckMarshaller.h File Reference	3078
7.232src/main/activemq/wireformat/openwire/marshal/generated/ProducerId-Marshaller.h File Reference	3078
7.233src/main/activemq/wireformat/openwire/marshal/generated/Producer-InfoMarshaller.h File Reference	3079

7.234src/main/activemq/wireformat/openwire/marshal/generated/Remove- InfoMarshaller.h File Reference	3079
7.235src/main/activemq/wireformat/openwire/marshal/generated/Remove- SubscriptionInfoMarshaller.h File Reference	3080
7.236src/main/activemq/wireformat/openwire/marshal/generated/Replay- CommandMarshaller.h File Reference	3081
7.237src/main/activemq/wireformat/openwire/marshal/generated/Response- Marshaller.h File Reference	3081
7.238src/main/activemq/wireformat/openwire/marshal/generated/SessionId- Marshaller.h File Reference	3082
7.239src/main/activemq/wireformat/openwire/marshal/generated/Session- InfoMarshaller.h File Reference	3083
7.240src/main/activemq/wireformat/openwire/marshal/generated/Shutdown- InfoMarshaller.h File Reference	3083
7.241src/main/activemq/wireformat/openwire/marshal/generated/Subscription- InfoMarshaller.h File Reference	3084
7.242src/main/activemq/wireformat/openwire/marshal/generated/Transaction- IdMarshaller.h File Reference	3084
7.243src/main/activemq/wireformat/openwire/marshal/generated/Transaction- InfoMarshaller.h File Reference	3085
7.244src/main/activemq/wireformat/openwire/marshal/generated/Wire- FormatInfoMarshaller.h File Reference	3086
7.245src/main/activemq/wireformat/openwire/marshal/generated/XATransaction- IdMarshaller.h File Reference	3086
7.246src/main/activemq/wireformat/openwire/marshal/PrimitiveTypesMarshaller.h File Reference	3087
7.247src/main/activemq/wireformat/openwire/OpenWireFormat.h File Reference	3088
7.248src/main/activemq/wireformat/openwire/OpenWireFormatFactory.h File Reference	3088
7.249src/main/activemq/wireformat/openwire/OpenWireFormatNegotiator.h - File Reference	3089
7.250src/main/activemq/wireformat/openwire/OpenWireResponseBuilder.h File Reference	3089
7.251src/main/activemq/wireformat/openwire/utils/BooleanStream.h File - Reference	3090
7.252src/main/activemq/wireformat/openwire/utils/HexTable.h File Reference .	3090
7.253src/main/activemq/wireformat/openwire/utils/MessagePropertyInterceptor.h File Reference	3091

7.254src/main/activemq/wireformat/stomp/StompCommandConstants.h File - Reference	3091
7.255src/main/activemq/wireformat/stomp/StompFrame.h File Reference . . .	3092
7.256src/main/activemq/wireformat/stomp/StompHelper.h File Reference . . .	3092
7.257src/main/activemq/wireformat/stomp/StompWireFormat.h File Reference	3093
7.258src/main/activemq/wireformat/stomp/StompWireFormatFactory.h File Reference	3093
7.259src/main/activemq/wireformat/WireFormat.h File Reference	3094
7.260src/main/activemq/wireformat/WireFormatFactory.h File Reference . . .	3094
7.261src/main/activemq/wireformat/WireFormatNegotiator.h File Reference . .	3095
7.262src/main/activemq/wireformat/WireFormatRegistry.h File Reference . . .	3095
7.263src/main/cms/BytesMessage.h File Reference	3096
7.264src/main/cms/Closeable.h File Reference	3096
7.265src/main/decaf/io/Closeable.h File Reference	3097
7.266src/main/cms/CMSException.h File Reference	3097
7.267src/main/cms/CMSProperties.h File Reference	3098
7.268src/main/cms/CMSSecurityException.h File Reference	3098
7.269src/main/cms/Connection.h File Reference	3098
7.270src/main/cms/ConnectionFactory.h File Reference	3099
7.271src/main/cms/ConnectionMetaData.h File Reference	3099
7.272src/main/cms/DeliveryMode.h File Reference	3100
7.273src/main/cms/Destination.h File Reference	3100
7.274src/main/cms/ExceptionListener.h File Reference	3100
7.275src/main/cms/IllegalStateException.h File Reference	3101
7.276src/main/decaf/lang/exceptions/IllegalStateException.h File Reference .	3101
7.277src/main/cms/InvalidClientIdException.h File Reference	3102
7.278src/main/cms/InvalidDestinationException.h File Reference	3102
7.279src/main/cms/InvalidSelectorException.h File Reference	3103
7.280src/main/cms/MapMessage.h File Reference	3103
7.281src/main/cms/MessageConsumer.h File Reference	3103
7.282src/main/cms/MessageEnumeration.h File Reference	3104
7.283src/main/cms/MessageEOFException.h File Reference	3104
7.284src/main/cms/MessageFormatException.h File Reference	3105
7.285src/main/cms/MessageListener.h File Reference	3105

7.286src/main/cms/MessageNotReadableException.h File Reference	3105
7.287src/main/cms/MessageNotWriteableException.h File Reference	3106
7.288src/main/cms/MessageProducer.h File Reference	3106
7.289src/main/cms/ObjectMessage.h File Reference	3107
7.290src/main/cms/Queue.h File Reference	3107
7.291src/main/decaf/util/Queue.h File Reference	3108
7.292src/main/cms/QueueBrowser.h File Reference	3108
7.293src/main/cms/Session.h File Reference	3109
7.294src/main/cms/Startable.h File Reference	3109
7.295src/main/cms/Stopable.h File Reference	3109
7.296src/main/cms/StreamMessage.h File Reference	3110
7.297src/main/cms/TemporaryQueue.h File Reference	3110
7.298src/main/cms/TemporaryTopic.h File Reference	3111
7.299src/main/cms/TextMessage.h File Reference	3111
7.300src/main/cms/Topic.h File Reference	3112
7.301src/main/cms/TransactionInProgressException.h File Reference	3112
7.302src/main/cms/TransactionRolledBackException.h File Reference	3112
7.303src/main/cms/UnsupportedOperationException.h File Reference	3113
7.304src/main/decaf/lang/exceptions/UnsupportedOperationException.h File Reference	3113
7.305src/main/cms/XAConnection.h File Reference	3114
7.306src/main/cms/XAConnectionFactory.h File Reference	3114
7.307src/main/cms/XAException.h File Reference	3115
7.308src/main/cms/XAResource.h File Reference	3115
7.309src/main/cms/XASession.h File Reference	3115
7.310src/main/cms/Xid.h File Reference	3116
7.311src/main/decaf/internal/AprPool.h File Reference	3116
7.312src/main/decaf/internal/DecafRuntime.h File Reference	3117
7.313src/main/decaf/internal/io/StandardErrorOutputStream.h File Reference	3117
7.314src/main/decaf/internal/io/StandardInputStream.h File Reference	3118
7.315src/main/decaf/internal/io/StandardOutputStream.h File Reference	3118
7.316src/main/decaf/internal/net/DefaultServerSocketFactory.h File Reference	3119
7.317src/main/decaf/internal/net/DefaultSocketFactory.h File Reference	3119
7.318src/main/decaf/internal/net/Network.h File Reference	3120

7.319src/main/decaf/internal/net/SocketFileDescriptor.h File Reference	3120
7.320src/main/decaf/internal/net/ssl/DefaultSSLContext.h File Reference . . .	3121
7.321src/main/decaf/internal/net/ssl/DefaultSSLServerSocketFactory.h File - Reference	3121
7.322src/main/decaf/internal/net/ssl/DefaultSSLSocketFactory.h File Reference	3122
7.323src/main/decaf/internal/net/ssl/openssl/OpenSSLContextSpi.h File - Reference	3122
7.324src/main/decaf/internal/net/ssl/openssl/OpenSSLParameters.h File - Reference	3123
7.325src/main/decaf/internal/net/ssl/openssl/OpenSSLServerSocket.h File Reference	3123
7.326src/main/decaf/internal/net/ssl/openssl/OpenSSLServerSocketFactory.h File Reference	3124
7.327src/main/decaf/internal/net/ssl/openssl/OpenSSLSocket.h File Reference	3124
7.328src/main/decaf/internal/net/ssl/openssl/OpenSSLSocketException.h - File Reference	3125
7.329src/main/decaf/internal/net/ssl/openssl/OpenSSLSocketFactory.h File - Reference	3125
7.330src/main/decaf/internal/net/ssl/openssl/OpenSSLSocketInputStream.h - File Reference	3126
7.331src/main/decaf/internal/net/ssl/openssl/OpenSSLSocketOutputStream.h File Reference	3126
7.332src/main/decaf/internal/net/tcp/TcpSocket.h File Reference	3127
7.333src/main/decaf/internal/net/tcp/TcpSocketInputStream.h File Reference .	3127
7.334src/main/decaf/internal/net/tcp/TcpSocketOutputStream.h File Reference	3128
7.335src/main/decaf/internal/net/URIEncoderDecoder.h File Reference	3128
7.336src/main/decaf/internal/net/URIHelper.h File Reference	3129
7.337src/main/decaf/internal/net/URIType.h File Reference	3129
7.338src/main/decaf/internal/nio/BufferFactory.h File Reference	3129
7.339src/main/decaf/internal/nio/ByteBuffer.h File Reference	3130
7.340src/main/decaf/internal/nio/CharArrayBuffer.h File Reference	3131
7.341src/main/decaf/internal/nio/DoubleArrayBuffer.h File Reference	3131
7.342src/main/decaf/internal/nio/FloatArrayBuffer.h File Reference	3132
7.343src/main/decaf/internal/nio/IntArrayBuffer.h File Reference	3132
7.344src/main/decaf/internal/nio/LongArrayBuffer.h File Reference	3133
7.345src/main/decaf/internal/nio/ShortArrayBuffer.h File Reference	3133
7.346src/main/decaf/internal/security/unix/SecureRandomImpl.h File Reference	3134

7.347src/main/decaf/internal/security/windows/SecureRandomImpl.h File - Reference	3134
7.348src/main/decaf/internal/util/ByteArrayAdapter.h File Reference	3135
7.349src/main/decaf/internal/util/concurrent/ConditionImpl.h File Reference . .	3135
7.350src/main/decaf/internal/util/concurrent/MutexImpl.h File Reference	3136
7.351src/main/decaf/internal/util/concurrent/SynchronizableImpl.h File - Reference	3136
7.352src/main/decaf/internal/util/concurrent/Transferer.h File Reference	3137
7.353src/main/decaf/internal/util/concurrent/TransferQueue.h File Reference .	3137
7.354src/main/decaf/internal/util/concurrent/TransferStack.h File Reference . .	3138
7.355src/main/decaf/internal/util/concurrent/unix/ConditionHandle.h File - Reference	3138
7.356src/main/decaf/internal/util/concurrent/windows/ConditionHandle.h File Reference	3139
7.357src/main/decaf/internal/util/concurrent/unix/MutexHandle.h File Reference	3139
7.358src/main/decaf/internal/util/concurrent/windows/MutexHandle.h File - Reference	3140
7.359src/main/decaf/internal/util/GenericResource.h File Reference	3140
7.360src/main/decaf/internal/util/HexStringParser.h File Reference	3141
7.361src/main/decaf/internal/util/Resource.h File Reference	3141
7.362src/main/decaf/internal/util/TimerTaskHeap.h File Reference	3141
7.363src/main/decaf/internal/util/zip/crc32.h File Reference	3142
7.363.1 Variable Documentation	3142
7.363.1.1 crc_table	3142
7.364src/main/decaf/internal/util/zip/deflate.h File Reference	3142
7.364.1 Define Documentation	3144
7.364.1.1 _tr_tally_dist	3144
7.364.1.2 _tr_tally_lit	3144
7.364.1.3 BL_CODES	3144
7.364.1.4 BUSY_STATE	3144
7.364.1.5 Code	3144
7.364.1.6 COMMENT_STATE	3144
7.364.1.7 d_code	3144
7.364.1.8 D_CODES	3144
7.364.1.9 Dad	3144

7.364.1.10	EXTRA_STATE	3144
7.364.1.11	FINISH_STATE	3144
7.364.1.12	Freq	3144
7.364.1.13	GZIP	3144
7.364.1.14	HCRC_STATE	3145
7.364.1.15	HEAP_SIZE	3145
7.364.1.16	INIT_STATE	3145
7.364.1.17	L_CODES	3145
7.364.1.18	Len	3145
7.364.1.19	LENGTH_CODES	3145
7.364.1.20	LITERALS	3145
7.364.1.21	MAX_BITS	3145
7.364.1.22	MAX_DIST	3145
7.364.1.23	max_insert_length	3145
7.364.1.24	MIN_LOOKAHEAD	3145
7.364.1.25	NAME_STATE	3145
7.364.1.26	put_byte	3145
7.364.1.27	WIN_INIT	3145
7.364.2	Typedef Documentation	3145
7.364.2.1	ct_data	3145
7.364.2.2	deflate_state	3145
7.364.2.3	IPos	3145
7.364.2.4	Pos	3145
7.364.2.5	Posf	3145
7.364.2.6	static_tree_desc	3145
7.364.2.7	tree_desc	3145
7.364.3	Function Documentation	3145
7.364.3.1	OF	3145
7.364.3.2	OF	3146
7.364.3.3	OF	3146
7.364.4	Variable Documentation	3146
7.364.4.1	_dist_code	3146
7.364.4.2	_length_code	3146
7.365	src/main/decaf/internal/util/zip/gzguts.h File Reference	3146

7.365.1 Define Documentation	3147
7.365.1.1 COPY	3147
7.365.1.2 GT_OFF	3147
7.365.1.3 GZ_APPEND	3147
7.365.1.4 GZ_NONE	3147
7.365.1.5 GZ_READ	3147
7.365.1.6 GZ_WRITE	3147
7.365.1.7 GZBUFSIZE	3147
7.365.1.8 GZIP	3147
7.365.1.9 local	3147
7.365.1.10 LOOK	3147
7.365.1.11 ZLIB_INTERNAL	3147
7.365.1.12 zstrerror	3147
7.365.2 Typedef Documentation	3147
7.365.2.1 gz_statep	3147
7.365.3 Function Documentation	3147
7.365.3.1 OF	3147
7.365.3.2 OF	3147
7.365.3.3 OF	3147
7.365.3.4 OF	3147
7.365.3.5 OF	3147
7.365.3.6 OF	3148
7.365.3.7 OF	3148
7.366src/main/decaf/internal/util/zip/inffast.h File Reference	3148
7.366.1 Function Documentation	3148
7.366.1.1 OF	3148
7.367src/main/decaf/internal/util/zip/inffixed.h File Reference	3148
7.368src/main/decaf/internal/util/zip/inflate.h File Reference	3148
7.368.1 Define Documentation	3148
7.368.1.1 GUNZIP	3148
7.368.2 Enumeration Type Documentation	3148
7.368.2.1 inflate_mode	3148
7.369src/main/decaf/internal/util/zip/inftrees.h File Reference	3149
7.369.1 Define Documentation	3150

7.369.1.1 ENOUGH	3150
7.369.1.2 ENOUGH_DISTs	3150
7.369.1.3 ENOUGH_LENS	3150
7.369.2 Enumeration Type Documentation	3150
7.369.2.1 codetype	3150
7.369.3 Function Documentation	3150
7.369.3.1 OF	3150
7.370src/main/decaf/internal/util/zip/trees.h File Reference	3150
7.370.1 Variable Documentation	3151
7.370.1.1 _dist_code	3151
7.370.1.2 _length_code	3151
7.370.1.3 base_dist	3152
7.370.1.4 base_length	3152
7.370.1.5 static_dtree	3152
7.370.1.6 static_ltree	3152
7.371src/main/decaf/internal/util/zip/zconf.h File Reference	3152
7.371.1 Define Documentation	3153
7.371.1.1 z_off64_t	3153
7.371.2 Typedef Documentation	3153
7.371.2.1 Byte	3153
7.371.2.2 Bytef	3153
7.371.2.3 charf	3153
7.371.2.4 intf	3153
7.371.2.5 ulnt	3153
7.371.2.6 ulntf	3153
7.371.2.7 uLong	3153
7.371.2.8 uLongf	3153
7.371.2.9 voidp	3153
7.371.2.10voidpc	3153
7.371.2.11voidpf	3153
7.372src/main/decaf/internal/util/zip/zlib.h File Reference	3153
7.372.1 Define Documentation	3156
7.372.1.1 deflateInit	3156
7.372.1.2 deflateInit2	3156

7.372.1.3 inflateBackInit	3156
7.372.1.4 inflateInit	3157
7.372.1.5 inflateInit2	3157
7.372.1.6 Z_ASCII	3157
7.372.1.7 Z_BEST_COMPRESSION	3157
7.372.1.8 Z_BEST_SPEED	3157
7.372.1.9 Z_BINARY	3157
7.372.1.10 Z_BLOCK	3157
7.372.1.11 Z_BUF_ERROR	3157
7.372.1.12 Z_DATA_ERROR	3157
7.372.1.13 Z_DEFAULT_COMPRESSION	3157
7.372.1.14 Z_DEFAULT_STRATEGY	3157
7.372.1.15 Z_DEFLATED	3157
7.372.1.16 Z_ERRNO	3157
7.372.1.17 Z_FILTERED	3157
7.372.1.18 Z_FINISH	3157
7.372.1.19 Z_FIXED	3157
7.372.1.20 Z_FULL_FLUSH	3157
7.372.1.21 Z_HUFFMAN_ONLY	3157
7.372.1.22 Z_MEM_ERROR	3157
7.372.1.23 Z_NEED_DICT	3157
7.372.1.24 Z_NO_COMPRESSION	3157
7.372.1.25 Z_NO_FLUSH	3157
7.372.1.26 Z_NULL	3157
7.372.1.27 Z_OK	3158
7.372.1.28 Z_PARTIAL_FLUSH	3158
7.372.1.29 Z_RLE	3158
7.372.1.30 Z_STREAM_END	3158
7.372.1.31 Z_STREAM_ERROR	3158
7.372.1.32 Z_SYNC_FLUSH	3158
7.372.1.33 Z_TEXT	3158
7.372.1.34 Z_TREES	3158
7.372.1.35 Z_UNKNOWN	3158
7.372.1.36 Z_VERSION_ERROR	3158

7.372.1.37ZLIB_VER_MAJOR	3158
7.372.1.38ZLIB_VER_MINOR	3158
7.372.1.39ZLIB_VER_REVISION	3158
7.372.1.40ZLIB_VER_SUBREVISION	3158
7.372.1.41ZLIB_VERNUM	3158
7.372.1.42ZLIB_VERSION	3158
7.372.1.43zlib_version	3158
7.372.2 Typedef Documentation	3158
7.372.2.1 gz_header	3158
7.372.2.2 gz_headerp	3158
7.372.2.3 gzFile	3158
7.372.2.4 OF	3158
7.372.2.5 z_stream	3158
7.372.2.6 z_streamp	3159
7.372.3 Function Documentation	3159
7.372.3.1 OF	3159
7.372.3.2 OF	3159
7.372.3.3 OF	3159
7.372.3.4 OF	3159
7.372.3.5 OF	3159
7.372.3.6 OF	3159
7.372.3.7 OF	3159
7.372.3.8 OF	3159
7.372.3.9 OF	3159
7.372.3.10OF	3159
7.372.3.11OF	3159
7.372.3.12OF	3159
7.372.3.13OF	3159
7.372.3.14OF	3159
7.372.3.15OF	3159
7.372.3.16OF	3159
7.372.3.17OF	3159
7.372.3.18OF	3159
7.372.3.19OF	3160

7.372.3.20OF	3160
7.372.3.21OF	3160
7.372.3.22OF	3160
7.372.3.23OF	3160
7.372.3.24OF	3160
7.372.3.25OF	3160
7.372.3.26OF	3160
7.372.3.27OF	3160
7.372.3.28OF	3160
7.372.3.29OF	3160
7.372.3.30OF	3160
7.372.3.31OF	3160
7.372.3.32OF	3160
7.372.3.33OF	3160
7.372.3.34OF	3160
7.372.3.35OF	3160
7.372.3.36OF	3160
7.372.3.37OF	3160
7.372.3.38OF	3160
7.372.3.39OF	3160
7.372.3.40OF	3161
7.372.3.41OF	3161
7.372.3.42OF	3161
7.373src/main/decaf/internal/util/zip/zutil.h File Reference	3161
7.373.1 Define Documentation	3162
7.373.1.1 Assert	3162
7.373.1.2 DEF_MEM_LEVEL	3162
7.373.1.3 DYN_TREES	3162
7.373.1.4 ERR_MSG	3162
7.373.1.5 ERR_RETURN	3162
7.373.1.6 MAX_MATCH	3162
7.373.1.7 MIN_MATCH	3162
7.373.1.8 PRESET_DICT	3162
7.373.1.9 STATIC_TREES	3162

7.373.1.10	STORED_BLOCK	3162
7.373.1.11	Trace	3162
7.373.1.12	Tracec	3162
7.373.1.13	Tracecv	3162
7.373.1.14	Tracev	3162
7.373.1.15	Tracevv	3163
7.373.1.16	TRY_FREE	3163
7.373.1.17	ZALLOC	3163
7.373.1.18	ZFREE	3163
7.373.1.19	ZLIB_INTERNAL	3163
7.373.2	Typedef Documentation	3163
7.373.2.1	uch	3163
7.373.2.2	uchf	3163
7.373.2.3	ulg	3163
7.373.2.4	ush	3163
7.373.2.5	ushf	3163
7.373.3	Function Documentation	3163
7.373.3.1	OF	3163
7.373.3.2	OF	3163
7.373.3.3	OF	3163
7.373.3.4	OF	3163
7.373.3.5	OF	3163
7.373.3.6	OF	3163
7.373.4	Variable Documentation	3163
7.373.4.1	z_errmsg	3163
7.374	src/main/decaf/io/BlockingByteArrayInputStream.h File Reference	3164
7.375	src/main/decaf/io/BufferedInputStream.h File Reference	3164
7.376	src/main/decaf/io/BufferedOutputStream.h File Reference	3164
7.377	src/main/decaf/io/ByteArrayInputStream.h File Reference	3165
7.378	src/main/decaf/io/ByteArrayOutputStream.h File Reference	3165
7.379	src/main/decaf/io/DataInput.h File Reference	3166
7.380	src/main/decaf/io/DataInputStream.h File Reference	3166
7.381	src/main/decaf/io/DataOutput.h File Reference	3167
7.382	src/main/decaf/io/DataOutputStream.h File Reference	3167

7.383src/main/decaf/io/EOFException.h File Reference	3168
7.384src/main/decaf/io/FileDescriptor.h File Reference	3168
7.385src/main/decaf/io/FilterInputStream.h File Reference	3169
7.386src/main/decaf/io/FilterOutputStream.h File Reference	3169
7.387src/main/decaf/io/Flushable.h File Reference	3170
7.388src/main/decaf/io/InputStream.h File Reference	3170
7.389src/main/decaf/io/InputStreamReader.h File Reference	3171
7.390src/main/decaf/io/InterruptedIOException.h File Reference	3171
7.391src/main/decaf/io/IOException.h File Reference	3171
7.392src/main/decaf/io/OutputStream.h File Reference	3172
7.393src/main/decaf/io/OutputStreamWriter.h File Reference	3172
7.394src/main/decaf/io/PushbackInputStream.h File Reference	3173
7.395src/main/decaf/io/Reader.h File Reference	3173
7.396src/main/decaf/io/UnsupportedEncodingException.h File Reference . . .	3174
7.397src/main/decaf/io/UTFDataFormatException.h File Reference	3174
7.398src/main/decaf/io/Writer.h File Reference	3174
7.399src/main/decaf/lang/Appendable.h File Reference	3175
7.400src/main/decaf/lang/ArrayPointer.h File Reference	3175
7.401src/main/decaf/lang/Boolean.h File Reference	3176
7.402src/main/decaf/lang/Byte.h File Reference	3177
7.403src/main/decaf/lang/Character.h File Reference	3177
7.404src/main/decaf/lang/CharSequence.h File Reference	3177
7.405src/main/decaf/lang/Comparable.h File Reference	3178
7.406src/main/decaf/lang/Double.h File Reference	3178
7.407src/main/decaf/lang/Exception.h File Reference	3179
7.408src/main/decaf/lang/exceptions/ClassCastException.h File Reference . .	3179
7.409src/main/decaf/lang/exceptions/IllegalArgumentException.h File - Reference	3180
7.410src/main/decaf/lang/exceptions/IllegalMonitorStateException.h File - Reference	3180
7.411src/main/decaf/lang/exceptions/IllegalThreadStateException.h File - Reference	3180
7.412src/main/decaf/lang/exceptions/IndexOutOfBoundsException.h File - Reference	3181
7.413src/main/decaf/lang/exceptions/InterruptedException.h File Reference .	3181

7.414src/main/decaf/lang/exceptions/InvalidStateException.h File Reference .	3182
7.415src/main/decaf/lang/exceptions/NullPointerException.h File Reference .	3182
7.416src/main/decaf/lang/exceptions/NumberFormatException.h File Reference	3183
7.417src/main/decaf/lang/exceptions/RuntimeException.h File Reference . . .	3183
7.418src/main/decaf/lang/Float.h File Reference	3183
7.419src/main/decaf/lang/Integer.h File Reference	3184
7.420src/main/decaf/lang/Iterable.h File Reference	3184
7.421src/main/decaf/lang/Long.h File Reference	3185
7.422src/main/decaf/lang/Math.h File Reference	3185
7.423src/main/decaf/lang/Number.h File Reference	3186
7.424src/main/decaf/lang/Pointer.h File Reference	3186
7.425src/main/decaf/lang/Readable.h File Reference	3187
7.426src/main/decaf/lang/Runnable.h File Reference	3187
7.427src/main/decaf/lang/Runtime.h File Reference	3188
7.428src/main/decaf/lang/Short.h File Reference	3188
7.429src/main/decaf/lang/String.h File Reference	3189
7.430src/main/decaf/lang/System.h File Reference	3189
7.431src/main/decaf/lang/Thread.h File Reference	3190
7.432src/main/decaf/lang/ThreadGroup.h File Reference	3190
7.433src/main/decaf/lang/Throwable.h File Reference	3191
7.434src/main/decaf/net/BindException.h File Reference	3191
7.435src/main/decaf/net/ConnectException.h File Reference	3192
7.436src/main/decaf/net/DatagramPacket.h File Reference	3192
7.437src/main/decaf/net/HttpRetryException.h File Reference	3192
7.438src/main/decaf/net/Inet4Address.h File Reference	3193
7.439src/main/decaf/net/Inet6Address.h File Reference	3193
7.440src/main/decaf/net/InetAddress.h File Reference	3194
7.441src/main/decaf/net/InetSocketAddress.h File Reference	3194
7.442src/main/decaf/net/MalformedURLException.h File Reference	3194
7.443src/main/decaf/net/NoRouteToHostException.h File Reference	3195
7.444src/main/decaf/net/PortUnreachableException.h File Reference	3195
7.445src/main/decaf/net/ProtocolException.h File Reference	3196
7.446src/main/decaf/net/ServerSocket.h File Reference	3196
7.447src/main/decaf/net/ServerSocketFactory.h File Reference	3196

7.448src/main/decaf/net/Socket.h File Reference	3197
7.449src/main/decaf/net/SocketAddress.h File Reference	3197
7.450src/main/decaf/net/SocketError.h File Reference	3198
7.451src/main/decaf/net/SocketException.h File Reference	3198
7.452src/main/decaf/net/SocketFactory.h File Reference	3199
7.453src/main/decaf/net/SocketImpl.h File Reference	3199
7.454src/main/decaf/net/SocketImplFactory.h File Reference	3200
7.455src/main/decaf/net/SocketOptions.h File Reference	3200
7.456src/main/decaf/net/SocketTimeoutException.h File Reference	3200
7.457src/main/decaf/net/ssl/SSLContext.h File Reference	3201
7.458src/main/decaf/net/ssl/SSLContextSpi.h File Reference	3201
7.459src/main/decaf/net/ssl/SSLParameters.h File Reference	3202
7.460src/main/decaf/net/ssl/SSLServerSocket.h File Reference	3202
7.461src/main/decaf/net/ssl/SSLServerSocketFactory.h File Reference	3203
7.462src/main/decaf/net/ssl/SSLSocket.h File Reference	3203
7.463src/main/decaf/net/ssl/SSLSocketFactory.h File Reference	3203
7.464src/main/decaf/net/UnknownHostException.h File Reference	3204
7.465src/main/decaf/net/UnknownServiceException.h File Reference	3204
7.466src/main/decaf/net/URI.h File Reference	3205
7.467src/main/decaf/net/URISyntaxException.h File Reference	3205
7.468src/main/decaf/net/URL.h File Reference	3206
7.469src/main/decaf/net/URLDecoder.h File Reference	3206
7.470src/main/decaf/net/URLEncoder.h File Reference	3206
7.471src/main/decaf/nio/Buffer.h File Reference	3207
7.472src/main/decaf/nio/BufferOverflowException.h File Reference	3207
7.473src/main/decaf/nio/BufferUnderflowException.h File Reference	3208
7.474src/main/decaf/nio/ByteBuffer.h File Reference	3208
7.475src/main/decaf/nio/CharBuffer.h File Reference	3208
7.476src/main/decaf/nio/DoubleBuffer.h File Reference	3209
7.477src/main/decaf/nio/FloatBuffer.h File Reference	3210
7.478src/main/decaf/nio/IntBuffer.h File Reference	3210
7.479src/main/decaf/nio/InvalidMarkException.h File Reference	3211
7.480src/main/decaf/nio/LongBuffer.h File Reference	3211
7.481src/main/decaf/nio/ReadOnlyBufferException.h File Reference	3211

7.482src/main/decaf/nio/ShortBuffer.h File Reference	3212
7.483src/main/decaf/security/auth/x500/X500Principal.h File Reference	3212
7.484src/main/decaf/security/cert/Certificate.h File Reference	3213
7.485src/main/decaf/security/cert/CertificateEncodingException.h File - Reference	3213
7.486src/main/decaf/security/cert/CertificateException.h File Reference	3214
7.487src/main/decaf/security/cert/CertificateExpiredException.h File Reference	3214
7.488src/main/decaf/security/cert/CertificateNotYetValidException.h File - Reference	3215
7.489src/main/decaf/security/cert/CertificateParsingException.h File Reference	3215
7.490src/main/decaf/security/cert/X509Certificate.h File Reference	3216
7.491src/main/decaf/security/GeneralSecurityException.h File Reference	3216
7.492src/main/decaf/security/InvalidKeyException.h File Reference	3216
7.493src/main/decaf/security/Key.h File Reference	3217
7.494src/main/decaf/security/KeyException.h File Reference	3217
7.495src/main/decaf/security/KeyManagementException.h File Reference	3218
7.496src/main/decaf/security/NoSuchAlgorithmException.h File Reference	3218
7.497src/main/decaf/security/NoSuchProviderException.h File Reference	3218
7.498src/main/decaf/security/Principal.h File Reference	3219
7.499src/main/decaf/security/PublicKey.h File Reference	3219
7.500src/main/decaf/security/SecureRandom.h File Reference	3220
7.501src/main/decaf/security/SecureRandomSpi.h File Reference	3220
7.502src/main/decaf/security/SignatureException.h File Reference	3221
7.503src/main/decaf/util/AbstractCollection.h File Reference	3221
7.504src/main/decaf/util/AbstractList.h File Reference	3222
7.505src/main/decaf/util/AbstractMap.h File Reference	3222
7.506src/main/decaf/util/AbstractQueue.h File Reference	3223
7.507src/main/decaf/util/AbstractSequentialList.h File Reference	3223
7.508src/main/decaf/util/AbstractSet.h File Reference	3224
7.509src/main/decaf/util/ArrayList.h File Reference	3224
7.510src/main/decaf/util/Arrays.h File Reference	3225
7.511src/main/decaf/util/Collection.h File Reference	3225
7.512src/main/decaf/util/Comparator.h File Reference	3226
7.513src/main/decaf/util/comparators/Less.h File Reference	3226

7.514src/main/decaf/util/concurrent/AbstractExecutorService.h File Reference	3227
7.515src/main/decaf/util/concurrent/atomic/AtomicBoolean.h File Reference	3227
7.516src/main/decaf/util/concurrent/atomic/AtomicInteger.h File Reference	3228
7.517src/main/decaf/util/concurrent/atomic/AtomicRefCounter.h File Reference	3228
7.518src/main/decaf/util/concurrent/atomic/AtomicReference.h File Reference	3229
7.519src/main/decaf/util/concurrent/BlockingQueue.h File Reference	3229
7.520src/main/decaf/util/concurrent/BrokenBarrierException.h File Reference	3230
7.521src/main/decaf/util/concurrent/Callable.h File Reference	3230
7.522src/main/decaf/util/concurrent/CancellationException.h File Reference	3230
7.523src/main/decaf/util/concurrent/Concurrent.h File Reference	3231
7.523.1 Define Documentation	3231
7.523.1.1 synchronized	3231
7.523.1.2 WAIT_INFINITE	3232
7.524src/main/decaf/util/concurrent/ConcurrentMap.h File Reference	3232
7.525src/main/decaf/util/concurrent/ConcurrentStlMap.h File Reference	3232
7.526src/main/decaf/util/concurrent/CopyOnWriteArrayList.h File Reference	3233
7.527src/main/decaf/util/concurrent/CopyOnWriteArraySet.h File Reference	3234
7.528src/main/decaf/util/concurrent/CountDownLatch.h File Reference	3234
7.529src/main/decaf/util/concurrent/Delayed.h File Reference	3235
7.530src/main/decaf/util/concurrent/ExecutionException.h File Reference	3235
7.531src/main/decaf/util/concurrent/Executor.h File Reference	3236
7.532src/main/decaf/util/concurrent/Executors.h File Reference	3236
7.533src/main/decaf/util/concurrent/ExecutorService.h File Reference	3237
7.534src/main/decaf/util/concurrent/Future.h File Reference	3237
7.535src/main/decaf/util/concurrent/LinkedBlockingQueue.h File Reference	3238
7.536src/main/decaf/util/concurrent/Lock.h File Reference	3238
7.537src/main/decaf/util/concurrent/locks/Lock.h File Reference	3239
7.538src/main/decaf/util/concurrent/locks/AbstractOwnableSynchronizer.h File Reference	3239
7.539src/main/decaf/util/concurrent/locks/Condition.h File Reference	3240
7.540src/main/decaf/util/concurrent/locks/LockSupport.h File Reference	3240
7.541src/main/decaf/util/concurrent/locks/ReadWriteLock.h File Reference	3241
7.542src/main/decaf/util/concurrent/locks/ReentrantLock.h File Reference	3241
7.543src/main/decaf/util/concurrent/Mutex.h File Reference	3242

7.544src/main/decaf/util/concurrent/RejectedExecutionException.h File	-
Reference	3242
7.545src/main/decaf/util/concurrent/RejectedExecutionHandler.h File Reference	3243
7.546src/main/decaf/util/concurrent/Semaphore.h File Reference	3243
7.547src/main/decaf/util/concurrent/Synchronizable.h File Reference	3244
7.548src/main/decaf/util/concurrent/SynchronousQueue.h File Reference . . .	3244
7.549src/main/decaf/util/concurrent/ThreadFactory.h File Reference	3245
7.550src/main/decaf/util/concurrent/ThreadPoolExecutor.h File Reference . . .	3245
7.551src/main/decaf/util/concurrent/TimeoutException.h File Reference	3246
7.552src/main/decaf/util/concurrent/TimeUnit.h File Reference	3247
7.553src/main/decaf/util/ConcurrentModificationException.h File Reference . .	3247
7.554src/main/decaf/util/Date.h File Reference	3248
7.555src/main/decaf/util/Deque.h File Reference	3248
7.556src/main/decaf/util/Iterator.h File Reference	3248
7.557src/main/decaf/util/LinkedList.h File Reference	3249
7.558src/main/decaf/util/List.h File Reference	3250
7.559src/main/decaf/util/ListIterator.h File Reference	3250
7.560src/main/decaf/util/logging/ConsoleHandler.h File Reference	3251
7.561src/main/decaf/util/logging/ErrorHandler.h File Reference	3251
7.562src/main/decaf/util/logging/Filter.h File Reference	3251
7.563src/main/decaf/util/logging/Formatter.h File Reference	3252
7.564src/main/decaf/util/logging/Handler.h File Reference	3252
7.565src/main/decaf/util/logging/Level.h File Reference	3253
7.566src/main/decaf/util/logging/Logger.h File Reference	3253
7.567src/main/decaf/util/logging/LoggerCommon.h File Reference	3254
7.568src/main/decaf/util/logging/LoggerDefines.h File Reference	3254
7.568.1 Define Documentation	3255
7.568.1.1 LOGDECAF_DEBUG	3255
7.568.1.2 LOGDECAF_DEBUG_1	3255
7.568.1.3 LOGDECAF_DECLARE	3255
7.568.1.4 LOGDECAF_DECLARE_LOCAL	3255
7.568.1.5 LOGDECAF_ERROR	3255
7.568.1.6 LOGDECAF_FATAL	3255
7.568.1.7 LOGDECAF_INFO	3255

7.568.1.8 LOGDECAF_INITIALIZE	3255
7.568.1.9 LOGDECAF_WARN	3255
7.569src/main/decaf/util/logging/LoggerHierarchy.h File Reference	3255
7.570src/main/decaf/util/logging/LogManager.h File Reference	3256
7.571src/main/decaf/util/logging/LogRecord.h File Reference	3256
7.572src/main/decaf/util/logging/LogWriter.h File Reference	3257
7.573src/main/decaf/util/logging/MarkBlockLogger.h File Reference	3257
7.574src/main/decaf/util/logging/PropertiesChangeListener.h File Reference	3258
7.575src/main/decaf/util/logging/SimpleFormatter.h File Reference	3258
7.576src/main/decaf/util/logging/SimpleLogger.h File Reference	3259
7.577src/main/decaf/util/logging/StreamHandler.h File Reference	3259
7.578src/main/decaf/util/logging/XMLFormatter.h File Reference	3260
7.579src/main/decaf/util/Map.h File Reference	3260
7.580src/main/decaf/util/NoSuchElementException.h File Reference	3260
7.581src/main/decaf/util/PriorityQueue.h File Reference	3261
7.582src/main/decaf/util/Properties.h File Reference	3261
7.583src/main/decaf/util/Random.h File Reference	3262
7.584src/main/decaf/util/Set.h File Reference	3262
7.585src/main/decaf/util/StIList.h File Reference	3263
7.586src/main/decaf/util/StIMap.h File Reference	3263
7.587src/main/decaf/util/StIQueue.h File Reference	3264
7.588src/main/decaf/util/StISet.h File Reference	3264
7.589src/main/decaf/util/StringTokenizer.h File Reference	3265
7.590src/main/decaf/util/Timer.h File Reference	3265
7.591src/main/decaf/util/TimerTask.h File Reference	3266
7.592src/main/decaf/util/UUID.h File Reference	3266
7.593src/main/decaf/util/zip/Adler32.h File Reference	3267
7.594src/main/decaf/util/zip/CheckedInputStream.h File Reference	3267
7.595src/main/decaf/util/zip/CheckedOutputStream.h File Reference	3268
7.596src/main/decaf/util/zip/Checksum.h File Reference	3268
7.597src/main/decaf/util/zip/CRC32.h File Reference	3269
7.598src/main/decaf/util/zip/DataFormatException.h File Reference	3269
7.599src/main/decaf/util/zip/Deflater.h File Reference	3270
7.600src/main/decaf/util/zip/DeflaterOutputStream.h File Reference	3270

7.601src/main/decaf/util/zip/Inflater.h File Reference	3271
7.602src/main/decaf/util/zip/InflaterInputStream.h File Reference	3271
7.603src/main/decaf/util/zip/ZipException.h File Reference	3272

Chapter 1

Namespace Index

1.1 Namespace List

Here is a list of all namespaces with brief descriptions:

activemq	
Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements	65
activemq::cmsutil	66
activemq::commands	67
activemq::core	68
activemq::core::policies	69
activemq::exceptions	70
activemq::io	70
activemq::library	70
activemq::state	70
activemq::threads	71
activemq::transport	71
activemq::transport::correlator	72
activemq::transport::failover	72
activemq::transport::inactivity	73
activemq::transport::logging	73
activemq::transport::mock	73
activemq::transport::tcp	73
activemq::util	74
activemq::wireformat	75
activemq::wireformat::openwire	75
activemq::wireformat::openwire::marshal	75
activemq::wireformat::openwire::marshal::generated	76
activemq::wireformat::openwire::utils	79
activemq::wireformat::stomp	79
cms	
Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements	80

decaf	
Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements	83
decaf::internal	83
decaf::internal::io	84
decaf::internal::net	84
decaf::internal::net::ssl	85
decaf::internal::net::ssl::openssl	85
decaf::internal::net::tcp	86
decaf::internal::nio	86
decaf::internal::security	87
decaf::internal::util	87
decaf::internal::util::concurrent	87
decaf::io	88
decaf::lang	89
decaf::lang::exceptions	92
decaf::net	92
decaf::net::ssl	94
decaf::nio	94
decaf::security	95
decaf::security::auth	95
decaf::security::auth::x500	95
decaf::security::cert	96
decaf::util	96
decaf::util::comparators	98
decaf::util::concurrent	98
decaf::util::concurrent::atomic	100
decaf::util::concurrent::locks	100
decaf::util::logging	101
decaf::util::zip	102
std	103

Chapter 2

Data Structure Index

2.1 Class Hierarchy

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

decaf::util::concurrent::locks::AbstractOwnableSynchronizer	135
activemq::core::ActiveMQAckHandler	156
activemq::core::ActiveMQConstants	231
activemq::library::ActiveMQCPP	244
decaf::lang::Appendable	442
decaf::io::Writer	2908
decaf::io::OutputStreamWriter	2074
decaf::nio::CharBuffer	785
decaf::internal::nio::CharArrayBuffer	773
decaf::internal::AprPool	444
decaf::lang::ArrayPointer< T, REFCOUNTER >	462
decaf::util::Arrays	471
decaf::util::concurrent::atomic::AtomicBoolean	473
decaf::util::concurrent::atomic::AtomicRefCounter	481
decaf::util::concurrent::atomic::AtomicReference< T >	484
activemq::wireformat::openwire::utils::BooleanStream	552
decaf::nio::Buffer	582
decaf::nio::ByteBuffer	690
decaf::internal::nio::ByteBuffer	646
decaf::nio::CharBuffer	785
decaf::nio::DoubleBuffer	1259
decaf::internal::nio::DoubleArrayBuffer	1248
decaf::nio::FloatBuffer	1368
decaf::internal::nio::FloatArrayBuffer	1357
decaf::nio::IntBuffer	1488
decaf::internal::nio::IntArrayBuffer	1477
decaf::nio::LongBuffer	1752
decaf::internal::nio::LongArrayBuffer	1742

decaf::nio::ShortBuffer	2419
decaf::internal::nio::ShortArrayBuffer	2408
decaf::internal::nio::BufferFactory	597
decaf::internal::util::ByteArrayAdapter	624
decaf::util::concurrent::Callable< V >	746
decaf::security::cert::Certificate	751
decaf::security::cert::X509Certificate	2915
decaf::lang::CharSequence	803
decaf::lang::String	2620
decaf::nio::CharBuffer	785
decaf::util::zip::Checksum	810
decaf::util::zip::Adler32	439
decaf::util::zip::CRC32	1065
cms::Closeable	815
activemq::commands::ActiveMQTempDestination	379
activemq::commands::ActiveMQTempQueue	386
activemq::commands::ActiveMQTempTopic	396
cms::Connection	933
activemq::core::ActiveMQConnection	187
activemq::core::ActiveMQXAConnection	432
cms::XAConnection	2917
activemq::core::ActiveMQXAConnection	432
cms::MessageConsumer	1877
activemq::cmsutil::CachedConsumer	734
activemq::core::ActiveMQConsumer	234
cms::MessageProducer	1924
activemq::cmsutil::CachedProducer	738
activemq::core::ActiveMQProducer	308
cms::QueueBrowser	2227
activemq::core::ActiveMQQueueBrowser	326
cms::Session	2361
activemq::cmsutil::PooledSession	2092
activemq::core::ActiveMQSession	333
activemq::core::ActiveMQXASession	436
cms::XASession	2935
activemq::core::ActiveMQXASession	436
decaf::io::Closeable	816
activemq::transport::Transport	2790
activemq::transport::CompositeTransport	896
activemq::transport::failover::FailoverTransport	1305
activemq::transport::IOTransport	1548
activemq::transport::mock::MockTransport	1948
activemq::transport::TransportFilter	2800
activemq::transport::correlator::ResponseCorrelator	2303
activemq::transport::inactivity::InactivityMonitor	1425
activemq::transport::logging::LoggingTransport	1709
activemq::transport::tcp::TcpTransport	2693

activemq::transport::tcp::SslTransport	2525
activemq::wireformat::WireFormatNegotiator	2904
activemq::wireformat::openwire::OpenWireFormatNegotiator	2060
decaf::io::InputStream	1464
decaf::internal::io::StandardInputStream	2530
decaf::internal::net::ssl::openssl::OpenSSLSocketInputStream	2040
decaf::internal::net::tcp::TcpSocketInputStream	2688
decaf::io::BlockingByteArrayInputStream	534
decaf::io::ByteArrayInputStream	679
decaf::io::FilterInputStream	1334
activemq::io::LoggingInputStream	1706
decaf::io::BufferedInputStream	589
decaf::io::DataInputStream	1094
decaf::io::PushbackInputStream	2214
decaf::util::zip::CheckedInputStream	805
decaf::util::zip::InflaterInputStream	1456
decaf::io::OutputStream	2066
decaf::internal::io::StandardErrorOutputStream	2528
decaf::internal::io::StandardOutputStream	2532
decaf::internal::net::ssl::openssl::OpenSSLSocketOutputStream	2043
decaf::internal::net::tcp::TcpSocketOutputStream	2691
decaf::io::ByteArrayOutputStream	687
decaf::io::FilterOutputStream	1341
activemq::io::LoggingOutputStream	1707
decaf::io::BufferedOutputStream	595
decaf::io::DataOutputStream	1109
decaf::util::zip::CheckedOutputStream	808
decaf::util::zip::DeflaterOutputStream	1189
decaf::io::Reader	2237
decaf::io::InputStreamReader	1475
decaf::io::Writer	2908
decaf::net::Socket	2452
decaf::net::ssl::SSLSocket	2513
decaf::internal::net::ssl::openssl::OpenSSLSocket	2014
decaf::util::logging::Handler	1401
decaf::util::logging::StreamHandler	2603
decaf::util::logging::ConsoleHandler	990
activemq::cmsutil::CmsAccessor	819
activemq::cmsutil::CmsDestinationAccessor	823
activemq::cmsutil::CmsTemplate	836
cms::CMSException	826
cms::CMSSecurityException	835
cms::IllegalStateException	1419
cms::InvalidClientIdException	1534
cms::InvalidDestinationException	1535
cms::InvalidSelectorException	1541
cms::MessageEOFException	1905
cms::MessageFormatException	1906

cms::MessageNotReadableException	1922
cms::MessageNotWriteableException	1923
cms::TransactionInProgressException	2782
cms::TransactionRolledBackException	2783
cms::UnsupportedOperationException	2827
cms::XAException	2921
activemq::util::CMSExceptionSupport	829
cms::CMSProperties	830
activemq::util::ActiveMQProperties	316
code	851
activemq::state::CommandVisitor	872
activemq::state::CommandVisitorAdapter	878
activemq::state::ConnectionStateTracker	984
decaf::lang::Comparable< T >	885
decaf::lang::Comparable< bool >	885
decaf::lang::Boolean	545
decaf::lang::Comparable< Boolean >	885
decaf::lang::Boolean	545
decaf::lang::Comparable< BrokerId >	885
activemq::commands::BrokerId	564
decaf::lang::Comparable< Byte >	885
decaf::lang::Byte	614
decaf::lang::Comparable< ByteBuffer >	885
decaf::nio::ByteBuffer	690
decaf::lang::Comparable< char >	885
decaf::lang::Character	765
decaf::lang::Comparable< Character >	885
decaf::lang::Character	765
decaf::lang::Comparable< CharBuffer >	885
decaf::nio::CharBuffer	785
decaf::lang::Comparable< ConnectionId >	885
activemq::commands::ConnectionId	960
decaf::lang::Comparable< ConsumerId >	885
activemq::commands::ConsumerId	1000
decaf::lang::Comparable< Date >	885
decaf::util::Date	1139
decaf::lang::Comparable< Delayed >	885
decaf::util::concurrent::Delayed	1194
decaf::lang::Comparable< Double >	885
decaf::lang::Double	1235
decaf::lang::Comparable< double >	885
decaf::lang::Double	1235
decaf::lang::Comparable< DoubleBuffer >	885
decaf::nio::DoubleBuffer	1259

decaf::lang::Comparable< Float >	885
decaf::lang::Float	1344
decaf::lang::Comparable< float >	885
decaf::lang::Float	1344
decaf::lang::Comparable< FloatBuffer >	885
decaf::nio::FloatBuffer	1368
decaf::lang::Comparable< int >	885
decaf::lang::Integer	1500
decaf::lang::Comparable< IntBuffer >	885
decaf::nio::IntBuffer	1488
decaf::lang::Comparable< Integer >	885
decaf::lang::Integer	1500
decaf::lang::Comparable< Level >	885
decaf::util::logging::Level	1616
decaf::lang::Comparable< LocalTransactionId >	885
activemq::commands::LocalTransactionId	1674
decaf::lang::Comparable< Long >	885
decaf::lang::Long	1726
decaf::lang::Comparable< long long >	885
decaf::lang::Long	1726
decaf::lang::Comparable< LongBuffer >	885
decaf::nio::LongBuffer	1752
decaf::lang::Comparable< MessageId >	885
activemq::commands::MessageId	1908
decaf::lang::Comparable< ProducerId >	885
activemq::commands::ProducerId	2182
decaf::lang::Comparable< SessionId >	885
activemq::commands::SessionId	2378
decaf::lang::Comparable< Short >	885
decaf::lang::Short	2398
decaf::lang::Comparable< short >	885
decaf::lang::Short	2398
decaf::lang::Comparable< ShortBuffer >	885
decaf::nio::ShortBuffer	2419
decaf::lang::Comparable< TimeUnit >	885
decaf::util::concurrent::TimeUnit	2756
decaf::lang::Comparable< TransactionId >	885
activemq::commands::TransactionId	2766
activemq::commands::LocalTransactionId	1674
activemq::commands::XATransactionId	2936
decaf::lang::Comparable< unsigned char >	885
decaf::lang::Byte	614
decaf::lang::Comparable< URI >	885

decaf::net::URI	2828
decaf::lang::Comparable< UUID >	885
decaf::util::UUID	2877
decaf::lang::Comparable< XATransactionId >	885
activemq::commands::XATransactionId	2936
decaf::util::Comparator< T >	888
decaf::util::Comparator< ArrayPointer< T, R > >	888
decaf::lang::ArrayPointerComparator< T, R >	470
decaf::util::Comparator< E >	888
decaf::util::comparators::Less< E >	1612
decaf::util::Comparator< Pointer< T, R > >	888
decaf::lang::PointerComparator< T, R >	2091
activemq::util::CompositeData	890
decaf::util::concurrent::locks::Condition	921
decaf::util::concurrent::ConditionHandle	928
decaf::internal::util::concurrent::ConditionImpl	929
cms::ConnectionFactory	955
activemq::core::ActiveMQConnectionFactory	213
activemq::core::ActiveMQXAConnectionFactory	433
cms::ConnectionMetaData	978
activemq::core::ActiveMQConnectionMetaData	227
activemq::state::ConnectionState	982
activemq::state::ConsumerState	1022
decaf::util::concurrent::CountDownLatch	1062
ct_data_s	1068
decaf::net::DatagramPacket	1078
decaf::io::DataInput	1086
decaf::io::DataOutput	1103
activemq::wireformat::openwire::marshal::DataStreamMarshaller	1119
activemq::wireformat::openwire::marshal::BaseDataStreamMarshaller	507
activemq::wireformat::openwire::marshal::generated::ActiveMQ-DestinationMarshaller	258
activemq::wireformat::openwire::marshal::generated::ActiveMQ-QueueMarshaller	329
activemq::wireformat::openwire::marshal::generated::ActiveMQ-TempDestinationMarshaller	382
activemq::wireformat::openwire::marshal::generated::ActiveMQ-QTempQueueMarshaller	391
activemq::wireformat::openwire::marshal::generated::ActiveMQ-QTempTopicMarshaller	401
activemq::wireformat::openwire::marshal::generated::ActiveMQ-TopicMarshaller	419
activemq::wireformat::openwire::marshal::generated::BaseCommand-Marshaller	499
activemq::wireformat::openwire::marshal::generated::BrokerInfo-Marshaller	578

activemq::wireformat::openwire::marshal::generated::Connection- ControlMarshaller	943
activemq::wireformat::openwire::marshal::generated::Connection- ErrorMarshaller	951
activemq::wireformat::openwire::marshal::generated::Connection- InfoMarshaller	974
activemq::wireformat::openwire::marshal::generated::Consumer- ControlMarshaller	996
activemq::wireformat::openwire::marshal::generated::Consumer- InfoMarshaller	1017
activemq::wireformat::openwire::marshal::generated::Control- CommandMarshaller	1025
activemq::wireformat::openwire::marshal::generated::Destination- InfoMarshaller	1218
activemq::wireformat::openwire::marshal::generated::Flush- CommandMarshaller	1383
activemq::wireformat::openwire::marshal::generated::KeepAlive- InfoMarshaller	1594
activemq::wireformat::openwire::marshal::generated::Message- AckMarshaller	1873
activemq::wireformat::openwire::marshal::generated::Message- DispatchMarshaller	1890
activemq::wireformat::openwire::marshal::generated::Message- DispatchNotificationMarshaller	1899
activemq::wireformat::openwire::marshal::generated::Message- Marshaller	1917
activemq::wireformat::openwire::marshal::generated::ActiveM- QBlobMessageMarshaller	161
activemq::wireformat::openwire::marshal::generated::ActiveM- QBytesMessageMarshaller	183
activemq::wireformat::openwire::marshal::generated::ActiveM- QMapMessageMarshaller	284
activemq::wireformat::openwire::marshal::generated::ActiveM- QMessageMarshaller	291
activemq::wireformat::openwire::marshal::generated::ActiveM- QObjectMessageMarshaller	304
activemq::wireformat::openwire::marshal::generated::ActiveM- QStreamMessageMarshaller	375
activemq::wireformat::openwire::marshal::generated::ActiveM- QTextMessageMarshaller	410
activemq::wireformat::openwire::marshal::generated::Message- PullMarshaller	1944
activemq::wireformat::openwire::marshal::generated::Producer- AckMarshaller	2175
activemq::wireformat::openwire::marshal::generated::Producer- InfoMarshaller	2195
activemq::wireformat::openwire::marshal::generated::RemoveInfo- Marshaller	2271
activemq::wireformat::openwire::marshal::generated::Remove- SubscriptionInfoMarshaller	2280

activemq::wireformat::openwire::marshal::generated::Replay- CommandMarshaller	2287
activemq::wireformat::openwire::marshal::generated::Response- Marshaller	2307
activemq::wireformat::openwire::marshal::generated::Data- ArrayResponseMarshaller	1072
activemq::wireformat::openwire::marshal::generated::Data- ResponseMarshaller	1115
activemq::wireformat::openwire::marshal::generated::Exception- ResponseMarshaller	1290
activemq::wireformat::openwire::marshal::generated::Integer- ResponseMarshaller	1519
activemq::wireformat::openwire::marshal::generated::SessionInfo- Marshaller	2390
activemq::wireformat::openwire::marshal::generated::Shutdown- InfoMarshaller	2434
activemq::wireformat::openwire::marshal::generated::Transaction- InfoMarshaller	2778
activemq::wireformat::openwire::marshal::generated::BrokerIdMarshaller	567
activemq::wireformat::openwire::marshal::generated::ConnectionId- Marshaller	963
activemq::wireformat::openwire::marshal::generated::ConsumerId- Marshaller	1005
activemq::wireformat::openwire::marshal::generated::DiscoveryEvent- Marshaller	1230
activemq::wireformat::openwire::marshal::generated::JournalQueue- AckMarshaller	1564
activemq::wireformat::openwire::marshal::generated::JournalTopic- AckMarshaller	1572
activemq::wireformat::openwire::marshal::generated::JournalTrace- Marshaller	1579
activemq::wireformat::openwire::marshal::generated::JournalTransaction- Marshaller	1587
activemq::wireformat::openwire::marshal::generated::MessageId- Marshaller	1912
activemq::wireformat::openwire::marshal::generated::NetworkBridge- FilterMarshaller	1974
activemq::wireformat::openwire::marshal::generated::PartialCommand- Marshaller	2079
activemq::wireformat::openwire::marshal::generated::LastPartial- CommandMarshaller	1608
activemq::wireformat::openwire::marshal::generated::ProducerId- Marshaller	2186
activemq::wireformat::openwire::marshal::generated::SessionId- Marshaller	2382
activemq::wireformat::openwire::marshal::generated::SubscriptionInfo- Marshaller	2635
activemq::wireformat::openwire::marshal::generated::TransactionId- Marshaller	2770

activemq::wireformat::openwire::marshal::generated::Local-TransactionIdMarshaller	1678
activemq::wireformat::openwire::marshal::generated::XATransaction-IdMarshaller	2942
activemq::wireformat::openwire::marshal::generated::WireFormatInfo-Marshaller	2900
decaf::internal::net::ssl::DefaultSSLContext	1164
decaf::util::zip::Deflater	1180
cms::DeliveryMode	1195
cms::Destination	1210
cms::Queue	2221
activemq::commands::ActiveMQQueue	321
cms::TemporaryQueue	2698
activemq::commands::ActiveMQTempQueue	386
cms::TemporaryTopic	2700
activemq::commands::ActiveMQTempTopic	396
cms::Topic	2764
activemq::commands::ActiveMQTopic	414
activemq::commands::ActiveMQDestination::DestinationFilter	1213
activemq::cmsutil::DestinationResolver	1222
activemq::cmsutil::DynamicDestinationResolver	1272
activemq::core::DispatchData	1234
activemq::core::Dispatcher	1234
activemq::core::ActiveMQConsumer	234
activemq::core::ActiveMQSession	333
decaf::lang::DYNAMIC_CAST_TOKEN	1271
decaf::util::Map< K, V, COMPARATOR >::Entry	1274
decaf::util::logging::ErrorManager	1277
cms::ExceptionListener	1286
decaf::util::concurrent::Executor	1297
decaf::util::concurrent::ExecutorService	1302
decaf::util::concurrent::AbstractExecutorService	122
decaf::util::concurrent::ThreadPoolExecutor	2715
decaf::util::concurrent::Executors	1299
decaf::io::FileDescriptor	1330
decaf::internal::net::SocketFileDescriptor	2478
decaf::util::logging::Filter	1333
decaf::io::Flushable	1380
decaf::io::OutputStream	2066
decaf::io::Writer	2908
decaf::util::logging::Formatter	1387
decaf::util::logging::SimpleFormatter	2441
decaf::util::logging::XMLFormatter	2950
decaf::util::concurrent::Future< V >	1390
activemq::transport::correlator::FutureResponse	1393
gz_header_s	1398
gz_state	1400

decaf::internal::util::HexStringParser	1406
activemq::wireformat::openwire::utils::HexTable	1407
activemq::util::IdGenerator	1411
decaf::net::InetAddress	1437
decaf::net::Inet4Address	1431
decaf::net::Inet6Address	1435
inflate_state	1445
decaf::util::zip::Inflater	1448
internal_state	1523
decaf::lang::Iterable< E >	1556
decaf::util::Collection< E >	851
decaf::util::AbstractCollection< E >	106
decaf::util::AbstractList< E >	123
decaf::util::AbstractSequentialList< E >	143
decaf::util::LinkedList< E >	1633
decaf::util::ArrayList< E >	445
decaf::util::StlList< E >	2536
decaf::util::AbstractQueue< E >	136
decaf::util::concurrent::BlockingQueue< E >	538
decaf::util::concurrent::LinkedBlockingQueue< E >	1621
decaf::util::concurrent::SynchronousQueue< E >	2655
decaf::util::PriorityQueue< E >	2160
decaf::util::AbstractSet< E >	152
decaf::util::concurrent::CopyOnWriteArraySet< E >	1050
decaf::util::StlSet< E >	2574
decaf::util::List< E >	1658
decaf::util::AbstractList< E >	123
decaf::util::concurrent::CopyOnWriteArrayList< E >	1030
decaf::util::Queue< E >	2222
decaf::util::AbstractQueue< E >	136
decaf::util::Deque< E >	1196
decaf::util::LinkedList< E >	1633
decaf::util::Set< E >	2397
decaf::util::AbstractSet< E >	152
decaf::lang::Iterable< PrimitiveValueNode >	1556
decaf::util::Collection< PrimitiveValueNode >	851
decaf::util::AbstractCollection< PrimitiveValueNode >	106
decaf::util::AbstractList< PrimitiveValueNode >	123
decaf::util::AbstractSequentialList< PrimitiveValueNode >	143
decaf::util::LinkedList< PrimitiveValueNode >	1633
activemq::util::PrimitiveList	2114
decaf::util::List< PrimitiveValueNode >	1658
decaf::util::AbstractList< PrimitiveValueNode >	123
decaf::util::Queue< PrimitiveValueNode >	2222
decaf::util::Deque< PrimitiveValueNode >	1196
decaf::util::LinkedList< PrimitiveValueNode >	1633
decaf::util::Iterator< E >	1559

decaf::util::ListIterator< E >	1671
decaf::util::concurrent::CopyOnWriteArrayList< E >::ArrayListIterator	458
decaf::util::Iterator< T >	1559
decaf::security::Key	1598
decaf::security::PublicKey	2213
std::less< decaf::lang::ArrayPointer< T > >	1614
std::less< decaf::lang::Pointer< T > >	1615
decaf::util::concurrent::Lock	1682
decaf::util::concurrent::locks::Lock	1684
decaf::util::concurrent::locks::ReentrantLock	2256
decaf::util::concurrent::locks::LockSupport	1690
decaf::util::logging::Logger	1693
decaf::util::logging::LoggerHierarchy	1706
decaf::util::logging::LogManager	1712
decaf::util::logging::LogRecord	1719
decaf::util::logging::LogWriter	1724
activemq::util::LongSequenceGenerator	1765
decaf::util::logging::MarkBlockLogger	1791
activemq::wireformat::MarshalAware	1792
activemq::commands::DataStructure	1133
activemq::commands::BaseDataStructure	528
activemq::commands::ActiveMQDestination	246
activemq::commands::ActiveMQQueue	321
activemq::commands::ActiveMQTempDestination	379
activemq::commands::ActiveMQTopic	414
activemq::commands::BooleanExpression	551
activemq::commands::BrokerId	564
activemq::commands::Command	866
activemq::commands::BaseCommand	491
activemq::commands::BrokerError	558
activemq::commands::BrokerInfo	572
activemq::commands::ConnectionControl	938
activemq::commands::ConnectionError	948
activemq::commands::ConnectionInfo	968
activemq::commands::ConsumerControl	992
activemq::commands::ConsumerInfo	1009
activemq::commands::ControlCommand	1022
activemq::commands::DestinationInfo	1213
activemq::commands::FlushCommand	1381
activemq::commands::KeepAliveInfo	1591
activemq::commands::Message	1821
activemq::commands::ActiveMQMessageTemplate< T >	295
activemq::commands::ActiveMQMessageTemplate< cms::-	
BytesMessage >	295
activemq::commands::ActiveMQBytesMessage	166
activemq::commands::ActiveMQMessageTemplate< cms::-	
MapMessage >	295
activemq::commands::ActiveMQMapMessage	264

activemq::commands::ActiveMQMessageTemplate< cms::-	
Message >	295
activemq::commands::ActiveMQBlobMessage	157
activemq::commands::ActiveMQMessage	288
activemq::commands::ActiveMQMessageTemplate< cms::-	
ObjectMessage >	295
activemq::commands::ActiveMQObjectMessage	301
activemq::commands::ActiveMQMessageTemplate< cms::-	
StreamMessage >	295
activemq::commands::ActiveMQStreamMessage	359
activemq::commands::ActiveMQMessageTemplate< cms::-	
TextMessage >	295
activemq::commands::ActiveMQTextMessage	405
activemq::commands::MessageAck	1867
activemq::commands::MessageDispatch	1881
activemq::commands::MessageDispatchNotification	1894
activemq::commands::MessagePull	1939
activemq::commands::ProducerAck	2171
activemq::commands::ProducerInfo	2190
activemq::commands::RemoveInfo	2267
activemq::commands::RemoveSubscriptionInfo	2275
activemq::commands::ReplayCommand	2284
activemq::commands::Response	2298
activemq::commands::DataArrayResponse	1069
activemq::commands::DataResponse	1112
activemq::commands::ExceptionResponse	1287
activemq::commands::IntegerResponse	1516
activemq::state::Tracked	2765
activemq::commands::SessionInfo	2386
activemq::commands::ShutdownInfo	2431
activemq::commands::TransactionInfo	2774
activemq::commands::WireFormatInfo	2889
activemq::commands::ConnectionId	960
activemq::commands::ConsumerId	1000
activemq::commands::DiscoveryEvent	1226
activemq::commands::JournalQueueAck	1561
activemq::commands::JournalTopicAck	1568
activemq::commands::JournalTrace	1577
activemq::commands::JournalTransaction	1583
activemq::commands::MessageId	1908
activemq::commands::NetworkBridgeFilter	1971
activemq::commands::PartialCommand	2076
activemq::commands::LastPartialCommand	1606
activemq::commands::ProducerId	2182
activemq::commands::SessionId	2378
activemq::commands::SubscriptionInfo	2631
activemq::commands::TransactionId	2766
activemq::wireformat::openwire::marshal::generated::MarshallerFactory	1795
activemq::util::MarshallingSupport	1796

decaf::lang::Math	1800
cms::Message	1839
activemq::commands::ActiveMQMessageTemplate< cms::Message >	295
cms::BytesMessage	718
activemq::commands::ActiveMQMessageTemplate< cms::Bytes- Message >	295
cms::MapMessage	1779
activemq::commands::ActiveMQMessageTemplate< cms::Map- Message >	295
cms::ObjectMessage	1997
activemq::commands::ActiveMQMessageTemplate< cms::Object- Message >	295
cms::StreamMessage	2606
activemq::commands::ActiveMQMessageTemplate< cms::Stream- Message >	295
cms::TextMessage	2701
activemq::commands::ActiveMQMessageTemplate< cms::Text- Message >	295
activemq::cmsutil::MessageCreator	1880
cms::MessageEnumeration	1903
activemq::core::ActiveMQQueueBrowser	326
cms::MessageListener	1916
activemq::wireformat::openwire::utils::MessagePropertyInterceptor	1933
decaf::util::concurrent::MutexHandle	1965
decaf::internal::util::concurrent::MutexImpl	1966
decaf::internal::net::Network	1968
decaf::lang::Number	1992
decaf::lang::Byte	614
decaf::lang::Character	765
decaf::lang::Double	1235
decaf::lang::Float	1344
decaf::lang::Integer	1500
decaf::lang::Long	1726
decaf::lang::Short	2398
decaf::util::concurrent::atomic::AtomicInteger	475
decaf::internal::net::ssl::openssl::OpenSSLParameters	2001
decaf::lang::Pointer< T, REFCOUNTER >	2083
activemq::core::PrefetchPolicy	2110
activemq::core::policies::DefaultPrefetchPolicy	1146
activemq::wireformat::openwire::marshal::PrimitiveTypesMarshaller	2135
activemq::util::PrimitiveValueNode::PrimitiveValue	2142
activemq::util::PrimitiveValueConverter	2144
activemq::util::PrimitiveValueNode	2145
decaf::security::Principal	2159
decaf::security::auth::x500::X500Principal	2914
activemq::cmsutil::ProducerCallback	2179
activemq::cmsutil::CmsTemplate::SendExecutor	2341

activemq::state::ProducerState	2199
decaf::util::Properties	2200
decaf::util::logging::PropertiesChangeListener	2210
decaf::util::Random	2229
decaf::security::SecureRandom	2320
decaf::lang::Readable	2235
decaf::io::Reader	2237
decaf::util::concurrent::locks::ReadWriteLock	2246
activemq::core::RedeliveryPolicy	2250
activemq::core::policies::DefaultRedeliveryPolicy	1150
decaf::util::concurrent::RejectedExecutionHandler	2266
decaf::util::concurrent::ThreadPoolExecutor::AbortPolicy	105
decaf::util::concurrent::ThreadPoolExecutor::CallerRunsPolicy	747
decaf::util::concurrent::ThreadPoolExecutor::CallerRunsPolicy::Discard- OldestPolicy	1224
decaf::util::concurrent::ThreadPoolExecutor::CallerRunsPolicy::Discard- Policy	1225
decaf::internal::util::Resource	2293
decaf::internal::util::GenericResource< T >	1397
activemq::cmsutil::ResourceLifecycleManager	2293
decaf::internal::util::ResourceLifecycleManager	2297
activemq::transport::mock::ResponseBuilder	2301
activemq::wireformat::openwire::OpenWireResponseBuilder	2064
decaf::lang::Runnable	2312
activemq::threads::CompositeTaskRunner	893
activemq::threads::DedicatedTaskRunner	1144
activemq::transport::IOTransport	1548
decaf::lang::Thread	2703
activemq::transport::mock::InternalCommandListener	1527
decaf::util::TimerTask	2751
activemq::threads::SchedulerTimerTask	2319
activemq::transport::inactivity::ReadChecker	2236
activemq::transport::inactivity::WriteChecker	2907
decaf::lang::Runtime	2313
decaf::internal::DecafRuntime	1143
decaf::security::SecureRandomSpi	2329
decaf::internal::security::SecureRandomImpl	2325
decaf::internal::security::SecureRandomImpl	2325
decaf::util::concurrent::Semaphore	2331
decaf::net::ServerSocket	2342
decaf::net::ssl::SSLServerSocket	2504
decaf::internal::net::ssl::openssl::OpenSSLServerSocket	2003
decaf::net::ServerSocketFactory	2352
decaf::internal::net::DefaultServerSocketFactory	1155
decaf::net::ssl::SSLServerSocketFactory	2510
decaf::internal::net::ssl::DefaultSSLServerSocketFactory	1165

decaf::internal::net::ssl::openssl::OpenSSLServerSocketFactory	2009
activemq::util::Service	2355
activemq::util::ServiceSupport	2358
activemq::threads::Scheduler	2317
activemq::util::ServiceListener	2356
activemq::util::ServiceStopper	2358
activemq::cmsutil::SessionCallback	2377
activemq::cmsutil::CmsTemplate::ProducerExecutor	2180
activemq::cmsutil::CmsTemplate::ResolveProducerExecutor	2291
activemq::cmsutil::CmsTemplate::ReceiveExecutor	2248
activemq::cmsutil::CmsTemplate::ResolveReceiveExecutor	2292
activemq::cmsutil::SessionPool	2394
activemq::state::SessionState	2395
decaf::util::logging::SimpleLogger	2442
decaf::net::SocketAddress	2470
decaf::net::InetSocketAddress	1445
decaf::net::SocketError	2470
decaf::net::SocketFactory	2473
decaf::internal::net::DefaultSocketFactory	1159
decaf::net::ssl::SSLSocketFactory	2522
decaf::internal::net::ssl::DefaultSSLSocketFactory	1170
decaf::internal::net::ssl::openssl::OpenSSLSocketFactory	2032
decaf::net::SocketImplFactory	2487
decaf::net::SocketOptions	2488
decaf::net::SocketImpl	2479
decaf::internal::net::tcp::TcpSocket	2678
decaf::net::ssl::SSLContext	2495
decaf::net::ssl::SSLContextSpi	2498
decaf::internal::net::ssl::openssl::OpenSSLContextSpi	1998
decaf::net::ssl::SSLParameters	2501
activemq::commands::BrokerError::StackTraceElement	2527
cms::Startable	2534
cms::Connection	933
cms::MessageConsumer	1877
cms::Session	2361
decaf::lang::STATIC_CAST_TOKEN	2535
activemq::core::ActiveMQConstants::StaticInitializer	2535
activemq::wireformat::stomp::StompCommandConstants	2581
activemq::wireformat::stomp::StompFrame	2587
activemq::wireformat::stomp::StompHelper	2592
cms::Stoppable	2602
cms::Connection	933
cms::MessageConsumer	1877
cms::Session	2361
decaf::util::StringTokenizer	2627
decaf::util::concurrent::Synchronizable	2639
activemq::core::MessageDispatchChannel	1886

activemq::core::FifoMessageDispatchChannel	1322
activemq::core::SimplePriorityMessageDispatchChannel	2444
decaf::util::Collection< PrimitiveValueNode >	851
decaf::internal::util::concurrent::SynchronizableImpl	2650
decaf::io::InputStream	1464
decaf::io::OutputStream	2066
decaf::util::Collection< E >	851
decaf::util::concurrent::Mutex	1960
decaf::util::Map< K, V, COMPARATOR >	1768
decaf::util::AbstractMap< K, V, COMPARATOR >	134
decaf::util::concurrent::ConcurrentMap< K, V, COMPARATOR >	898
decaf::util::concurrent::ConcurrentStlMap< K, V, COMPARATOR >	905
decaf::util::StlMap< K, V, COMPARATOR >	2552
decaf::util::StlQueue< T >	2565
decaf::util::Map< std::string, PrimitiveValueNode, std::less< std::string > >	1768
decaf::util::StlMap< std::string, PrimitiveValueNode >	2552
activemq::util::PrimitiveMap	2125
activemq::core::Synchronization	2654
decaf::lang::System	2665
activemq::threads::Task	2676
activemq::core::ActiveMQSessionExecutor	355
activemq::threads::CompositeTask	892
activemq::transport::failover::BackupTransportPool	489
activemq::transport::failover::CloseTransportsTask	818
activemq::transport::failover::FailoverTransport	1305
activemq::threads::CompositeTaskRunner	893
activemq::threads::TaskRunner	2677
activemq::threads::CompositeTaskRunner	893
activemq::threads::DedicatedTaskRunner	1144
decaf::util::concurrent::ThreadFactory	2714
decaf::lang::ThreadGroup	2715
decaf::lang::Throwable	2732
decaf::lang::Exception	1279
activemq::exceptions::ActiveMQException	262
activemq::exceptions::BrokerException	563
activemq::exceptions::ConnectionFailedException	958
decaf::io::IOException	1545
decaf::io::EOFException	1275
decaf::io::InterruptedIOException	1531
decaf::net::SocketTimeoutException	2493
decaf::io::UnsupportedEncodingException	2821
decaf::io::UTFDataFormatException	2874
decaf::net::HttpRetryException	1408
decaf::net::MalformedURLException	1766
decaf::net::ProtocolException	2211
decaf::net::SocketException	2471
decaf::internal::net::ssl::openssl::OpenSSLSocketException	2029
decaf::net::BindException	532

decaf::net::ConnectException	931
decaf::net::NoRouteToHostException	1978
decaf::net::PortUnreachableException	2107
decaf::net::UnknownHostException	2816
decaf::net::UnknownServiceException	2819
decaf::util::zip::ZipException	2953
decaf::lang::exceptions::ClassCastException	813
decaf::lang::exceptions::IllegalArgumentException	1413
decaf::lang::exceptions::IllegalMonitorStateException	1416
decaf::lang::exceptions::IllegalStateException	1420
decaf::nio::InvalidMarkException	1539
decaf::lang::exceptions::IllegalThreadStateException	1423
decaf::lang::exceptions::IndexOutOfBoundsException	1429
decaf::lang::exceptions::InterruptedException	1529
decaf::lang::exceptions::InvalidStateException	1543
decaf::lang::exceptions::NullPointerException	1989
decaf::lang::exceptions::NumberFormatException	1994
decaf::lang::exceptions::RuntimeException	2315
decaf::util::ConcurrentModificationException	902
decaf::util::NoSuchElementException	1984
decaf::lang::exceptions::UnsupportedOperationException	2824
decaf::nio::ReadOnlyBufferException	2244
decaf::net::URISyntaxException	2856
decaf::nio::BufferOverflowException	609
decaf::nio::BufferUnderflowException	611
decaf::security::GeneralSecurityException	1395
decaf::security::cert::CertificateException	757
decaf::security::cert::CertificateEncodingException	755
decaf::security::cert::CertificateExpiredException	759
decaf::security::cert::CertificateNotYetValidException	761
decaf::security::cert::CertificateParsingException	763
decaf::security::KeyException	1600
decaf::security::InvalidKeyException	1536
decaf::security::KeyManagementException	1603
decaf::security::NoSuchAlgorithmException	1981
decaf::security::NoSuchProviderException	1986
decaf::security::SignatureException	2438
decaf::util::concurrent::BrokenBarrierException	556
decaf::util::concurrent::CancellationException	748
decaf::util::concurrent::ExecutionException	1294
decaf::util::concurrent::RejectedExecutionException	2263
decaf::util::concurrent::TimeoutException	2737
decaf::util::zip::DataFormatException	1076
decaf::util::Timer	2739
decaf::internal::util::TimerTaskHeap	2753
activemq::state::TransactionState	2785
decaf::internal::util::concurrent::Transferer< E >	2786
decaf::internal::util::concurrent::TransferQueue< E >	2787
decaf::internal::util::concurrent::TransferStack< E >	2789

activemq::transport::TransportFactory	2798
activemq::transport::AbstractTransportFactory	154
activemq::transport::failover::FailoverTransportFactory	1318
activemq::transport::mock::MockTransportFactory	1958
activemq::transport::tcp::TcpTransportFactory	2696
activemq::transport::tcp::SslTransportFactory	2527
activemq::transport::TransportListener	2810
activemq::core::ActiveMQConnection	187
activemq::transport::DefaultTransportListener	1178
activemq::transport::failover::BackupTransport	486
activemq::transport::mock::InternalCommandListener	1527
activemq::transport::failover::FailoverTransportListener	1320
activemq::transport::TransportFilter	2800
activemq::transport::TransportRegistry	2812
tree_desc_s	2815
decaf::lang::Thread::UncaughtExceptionHandler	2815
decaf::internal::net::URLEncoderDecoder	2841
decaf::internal::net::URIHelper	2844
activemq::transport::failover::URIPool	2851
activemq::util::URISupport	2854
decaf::internal::net::URIType	2860
decaf::net::URL	2868
decaf::net::URLDecoder	2870
decaf::net::URLEncoder	2871
activemq::util::Usage	2872
activemq::util::MemoryUsage	1818
activemq::wireformat::WireFormat	2884
activemq::wireformat::openwire::OpenWireFormat	2045
activemq::wireformat::stomp::StompWireFormat	2597
activemq::wireformat::WireFormatFactory	2888
activemq::wireformat::openwire::OpenWireFormatFactory	2059
activemq::wireformat::stomp::StompWireFormatFactory	2601
activemq::wireformat::WireFormatRegistry	2905
cms::XAConnectionFactory	2918
activemq::core::ActiveMQXAConnectionFactory	433
cms::XAResource	2926
activemq::core::ActiveMQTransactionContext	423
cms::Xid	2946
activemq::commands::XATransactionId	2936
z_stream_s	2952

Chapter 3

Data Structure Index

3.1 Data Structures

Here are the data structures with brief descriptions:

decaf::util::concurrent::ThreadPoolExecutor::AbortPolicy	
Handler policy for tasks that are rejected upon a call to ThreadPool- Executor::execute (p. 2724) this class always throws a Rejected- ExecutionException (p. 2263)	105
decaf::util::AbstractCollection< E >	
This class provides a skeletal implementation of the Collection (p. 851) interface, to minimize the effort required to implement this interface	106
decaf::util::concurrent::AbstractExecutorService	
Provides a default implementation for the methods of the Executor- Service (p. 1302) interface	122
decaf::util::AbstractList< E >	
This class provides a skeletal implementation of the List (p. 1658) interface to minimize the effort required to implement this interface backed by a "random access" data store (such as an array)	123
decaf::util::AbstractMap< K, V, COMPARATOR >	
This class provides a skeletal implementation of the Map (p. 1768) interface, to minimize the effort required to implement this interface	134
decaf::util::concurrent::locks::AbstractOwnableSynchronizer	
Base class for locks that provide the notion of Ownership, the types of locks that are implemented using this base class would be owned by one specific Thread at any given time	135
decaf::util::AbstractQueue< E >	
This class provides skeletal implementations of some Queue (p. 2222) operations	136
decaf::util::AbstractSequentialList< E >	
This class provides a skeletal implementation of the List (p. 1658) interface to minimize the effort required to implement this interface backed by a "sequential access" data store (such as a linked list)	143

decaf::util::AbstractSet< E >	
This class provides a skeletal implementation of the Set (p. 2397) interface to minimize the effort required to implement this interface	152
activemq::transport::AbstractTransportFactory	
Abstract implementation of the TransportFactory (p. 2798) interface, providing the base functionality that's common to most of the TransportFactory (p. 2798) instances	154
activemq::core::ActiveMQAckHandler	
Interface class that is used to give CMS Messages an interface to Ack themselves with	156
activemq::commands::ActiveMQBlobMessage	157
activemq::wireformat::openwire::marshal::generated::ActiveMQBlobMessageMarshaller	
Marshaling code for Open Wire Format for ActiveMQBlobMessageMarshaller (p. 161)	161
activemq::commands::ActiveMQBytesMessage	166
activemq::wireformat::openwire::marshal::generated::ActiveMQBytesMessageMarshaller	
Marshaling code for Open Wire Format for ActiveMQBytesMessageMarshaller (p. 183)	183
activemq::core::ActiveMQConnection	
Concrete connection used for all connectors to the ActiveMQ broker	187
activemq::core::ActiveMQConnectionFactory	213
activemq::core::ActiveMQConnectionMetaData	
This class houses all the various settings and information that is used by an instance of an ActiveMQConnection (p. 187) class	227
activemq::core::ActiveMQConstants	
Class holding constant values for various ActiveMQ specific things Each constant is defined as an enumeration and has functions that convert back an forth between string and enum values	231
activemq::core::ActiveMQConsumer	234
activemq::library::ActiveMQCPP	244
activemq::commands::ActiveMQDestination	246
activemq::wireformat::openwire::marshal::generated::ActiveMQDestinationMarshaller	
Marshaling code for Open Wire Format for ActiveMQDestinationMarshaller (p. 258)	258
activemq::exceptions::ActiveMQException	262
activemq::commands::ActiveMQMapMessage	264
activemq::wireformat::openwire::marshal::generated::ActiveMQMapMessageMarshaller	
Marshaling code for Open Wire Format for ActiveMQMapMessageMarshaller (p. 284)	284
activemq::commands::ActiveMQMessage	288
activemq::wireformat::openwire::marshal::generated::ActiveMQMessageMarshaller	
Marshaling code for Open Wire Format for ActiveMQMessageMarshaller (p. 291)	291
activemq::commands::ActiveMQMessageTemplate< T >	295
activemq::commands::ActiveMQObjectMessage	301

activemq::wireformat::openwire::marshal::generated::ActiveMQObject- MessageMarshaller	
Marshaling code for Open Wire Format for ActiveMQObject- MessageMarshaller (p.304)	304
activemq::core::ActiveMQProducer	308
activemq::util::ActiveMQProperties	
Implementation of the CMSProperties interface that delegates to a decaf::util::Properties (p.2200) object	316
activemq::commands::ActiveMQQueue	321
activemq::core::ActiveMQQueueBrowser	326
activemq::wireformat::openwire::marshal::generated::ActiveMQQueue- Marshaller	
Marshaling code for Open Wire Format for ActiveMQQueue- Marshaller (p.329)	329
activemq::core::ActiveMQSession	333
activemq::core::ActiveMQSessionExecutor	
Delegate dispatcher for a single session	355
activemq::commands::ActiveMQStreamMessage	359
activemq::wireformat::openwire::marshal::generated::ActiveMQStream- MessageMarshaller	
Marshaling code for Open Wire Format for ActiveMQStream- MessageMarshaller (p.375)	375
activemq::commands::ActiveMQTempDestination	379
activemq::wireformat::openwire::marshal::generated::ActiveMQTemp- DestinationMarshaller	
Marshaling code for Open Wire Format for ActiveMQTemp- DestinationMarshaller (p.382)	382
activemq::commands::ActiveMQTempQueue	386
activemq::wireformat::openwire::marshal::generated::ActiveMQTemp- QueueMarshaller	
Marshaling code for Open Wire Format for ActiveMQTempQueue- Marshaller (p.391)	391
activemq::commands::ActiveMQTempTopic	396
activemq::wireformat::openwire::marshal::generated::ActiveMQTemp- TopicMarshaller	
Marshaling code for Open Wire Format for ActiveMQTempTopic- Marshaller (p.401)	401
activemq::commands::ActiveMQTextMessage	405
activemq::wireformat::openwire::marshal::generated::ActiveMQText- MessageMarshaller	
Marshaling code for Open Wire Format for ActiveMQTextMessage- Marshaller (p.410)	410
activemq::commands::ActiveMQTopic	414
activemq::wireformat::openwire::marshal::generated::ActiveMQTopic- Marshaller	
Marshaling code for Open Wire Format for ActiveMQTopic- Marshaller (p.419)	419
activemq::core::ActiveMQTransactionContext	
Transaction Management class, hold messages that are to be rede- livered upon a request to roll-back	423

activemq::core::ActiveMQXAConnection	432
activemq::core::ActiveMQXAConnectionFactory	433
activemq::core::ActiveMQXASession	436
decaf::util::zip::Adler32	
Clas that can be used to compute an Adler-32 Checksum (p. 810) for a data stream	439
decaf::lang::Appendable	
An object to which char sequences and values can be appended	442
decaf::internal::AprPool	
Wraps an APR pool object so that classes in decaf can create a static member for use in static methods where apr function calls that need a pool are made	444
decaf::util::ArrayList< E >	445
decaf::util::concurrent::CopyOnWriteArrayList< E >::ArrayListIterator	458
decaf::lang::ArrayPointer< T, REFCOUNTER >	
Decaf's implementation of a Smart Pointer (p. 2083) that is a tem- plate on a Type and is Thread (p. 2703) Safe if the default Reference Counter is used	462
decaf::lang::ArrayPointerComparator< T, R >	
This implementation of Comparator is designed to allows objects in a Collection to be sorted or tested for equality based on the value of the value of the actual pointer to the array being contained in this ArrayPointer (p. 462)	470
decaf::util::Arrays	471
decaf::util::concurrent::atomic::AtomicBoolean	
A boolean value that may be updated atomically	473
decaf::util::concurrent::atomic::AtomicInteger	
An int value that may be updated atomically	475
decaf::util::concurrent::atomic::AtomicRefCounter	481
decaf::util::concurrent::atomic::AtomicReference< T >	
An Pointer reference that may be updated atomically	484
activemq::transport::failover::BackupTransport	486
activemq::transport::failover::BackupTransportPool	489
activemq::commands::BaseCommand	491
activemq::wireformat::openwire::marshal::generated::BaseCommand- Marshaller	
Marshaling code for Open Wire Format for BaseCommand- Marshaller (p. 499)	499
activemq::wireformat::openwire::marshal::BaseDataStreamMarshaller	
Base class for all Marshallers that marshal DataStructures to and from the wire using the OpenWire protocol	507
activemq::commands::BaseDataStructure	528
decaf::net::BindException	532
decaf::io::BlockingByteArrayInputStream	
This is a blocking version of a byte buffer stream	534
decaf::util::concurrent::BlockingQueue< E >	
A decaf::util::Queue (p. 2222) that additionally supports operations that wait for the queue to become non-empty when retrieving an el- ement, and wait for space to become available in the queue when storing an element	538

decaf::lang::Boolean	545
activemq::commands::BooleanExpression	551
activemq::wireformat::openwire::utils::BooleanStream	
Manages the writing and reading of boolean data streams to and from a data source such as a <code>DataInputStream</code> or <code>DataOutputStream</code>	552
decaf::util::concurrent::BrokenBarrierException	556
activemq::commands::BrokerError	
This class represents an Exception sent from the Broker	558
activemq::exceptions::BrokerException	563
activemq::commands::BrokerId	564
activemq::wireformat::openwire::marshal::generated::BrokerIdMarshaller	
Marshaling code for Open Wire Format for BrokerIdMarshaller (p. 567)	567
activemq::commands::BrokerInfo	572
activemq::wireformat::openwire::marshal::generated::BrokerInfoMarshaller	
Marshaling code for Open Wire Format for BrokerInfoMarshaller (p. 578)	578
decaf::nio::Buffer	
A container for data of a specific primitive type	582
decaf::io::BufferedInputStream	
A wrapper around another input stream that performs a buffered read, where it reads more data than it needs in order to reduce the number of io operations on the input stream	589
decaf::io::BufferedOutputStream	
Wrapper around another output stream that buffers output before writing to the target output stream	595
decaf::internal::nio::BufferFactory	
Factory class used by static methods in the decaf::nio (p. 94) package to create the various default version of the NIO interfaces	597
decaf::nio::BufferOverflowException	609
decaf::nio::BufferUnderflowException	611
decaf::lang::Byte	614
decaf::internal::util::ByteArrayAdapter	
This class adapts primitive type arrays to a base byte array so that the classes can inter-operate on the same base byte array without copying data	624
decaf::internal::nio::ByteArrayBuffer	
This class defines six categories of operations upon byte buffers:	646
decaf::io::ByteArrayInputStream	
A ByteArrayInputStream (p. 679) contains an internal buffer that contains bytes that may be read from the stream	679
decaf::io::ByteArrayOutputStream	687
decaf::nio::ByteBuffer	
This class defines six categories of operations upon byte buffers:	690
cms::BytesMessage	
A BytesMessage (p. 718) object is used to send a message containing a stream of unsigned bytes	718
activemq::cmsutil::CachedConsumer	
A cached message consumer contained within a pooled session	734

activemq::cmsutil::CachedProducer	
A cached message producer contained within a pooled session . . .	738
decaf::util::concurrent::Callable< V >	
A task that returns a result and may throw an exception	746
decaf::util::concurrent::ThreadPoolExecutor::CallerRunsPolicy	
Handler policy for tasks that are rejected upon a call to ThreadPoolExecutor::execute (p. 2724) this class will attempt to run the task in the Thread that called the execute method unless the executor is shutdown in which case the task is not run and is destroyed	747
decaf::util::concurrent::CancellationException	748
decaf::security::cert::Certificate	
Base interface for all identity certificates	751
decaf::security::cert::CertificateEncodingException	755
decaf::security::cert::CertificateException	757
decaf::security::cert::CertificateExpiredException	759
decaf::security::cert::CertificateNotYetValidException	761
decaf::security::cert::CertificateParsingException	763
decaf::lang::Character	765
decaf::internal::nio::CharArrayBuffer	773
decaf::nio::CharBuffer	
This class defines four categories of operations upon character buffers:	785
decaf::lang::CharSequence	
A CharSequence (p. 803) is a readable sequence of char values . .	803
decaf::util::zip::CheckedInputStream	
An implementation of a FilterInputStream that will maintain a - Checksum (p. 810) of the bytes read, the Checksum (p. 810) can then be used to verify the integrity of the input stream	805
decaf::util::zip::CheckedOutputStream	
An implementation of a FilterOutputStream that will maintain a - Checksum (p. 810) of the bytes written, the Checksum (p. 810) can then be used to verify the integrity of the output stream	808
decaf::util::zip::Checksum	
An interface used to represent Checksum (p. 810) values in the Zip package	810
decaf::lang::exceptions::ClassCastException	813
cms::Closeable	
Interface for a class that implements the close method	815
decaf::io::Closeable	
Interface for a class that implements the close method	816
activemq::transport::failover::CloseTransportsTask	818
activemq::cmsutil::CmsAccessor	
Base class for activemq::cmsutil::CmsTemplate (p. 836) and other CMS-accessing gateway helpers, defining common properties such as the CMS cms.ConnectionFactory (p. 955) to operate on	819
activemq::cmsutil::CmsDestinationAccessor	
Extends the CmsAccessor (p. 819) to add support for resolving destination names	823

cms::CMSException	
CMS API Exception that is the base for all exceptions thrown from CMS classes	826
activemq::util::CMSExceptionSupport	829
cms::CMSProperties	
Interface for a Java-like properties object	830
cms::CMSSecurityException	
This exception must be thrown when a provider rejects a user name/-password submitted by a client	835
activemq::cmsutil::CmsTemplate	
CmsTemplate (p. 836) simplifies performing synchronous CMS operations	836
code	851
decaf::util::Collection< E >	
The root interface in the collection hierarchy	851
activemq::commands::Command	866
activemq::state::CommandVisitor	
Interface for an Object that can visit the various Command Objects that are sent from and to this client	872
activemq::state::CommandVisitorAdapter	
Default Implementation of a CommandVisitor (p. 872) that returns NULL for all calls	878
decaf::lang::Comparable< T >	
This interface imposes a total ordering on the objects of each class that implements it	885
decaf::util::Comparator< T >	
A comparison function, which imposes a total ordering on some collection of objects	888
activemq::util::CompositeData	
Represents a Composite URI	890
activemq::threads::CompositeTask	
Represents a single task that can be part of a set of Tasks that are contained in a CompositeTaskRunner (p. 893)	892
activemq::threads::CompositeTaskRunner	
A Task (p. 2676) Runner that can contain one or more Composite-Tasks that are each checked for pending work and run if any is present in the order that the tasks were added	893
activemq::transport::CompositeTransport	
A Composite Transport (p. 2790) is a Transport (p. 2790) implementation that is composed of several Transports	896
decaf::util::concurrent::ConcurrentMap< K, V, COMPARATOR >	
Interface for a Map (p. 1768) type that provides additional atomic put-IfAbsent, remove, and replace methods alongside the already available Map (p. 1768) interface	898
decaf::util::ConcurrentModificationException	902
decaf::util::concurrent::ConcurrentStlMap< K, V, COMPARATOR >	
Map (p. 1768) template that wraps around a std::map to provide a more user-friendly interface and to provide common functions that do not exist in std::map	905

decaf::util::concurrent::locks::Condition	
Condition (p. 921) factors out the Mutex (p. 1960) monitor methods (wait, notify and notifyAll) into distinct objects to give the effect of having multiple wait-sets per object, by combining them with the use of arbitrary Lock (p. 1684) implementations	921
decaf::util::concurrent::ConditionHandle	928
decaf::internal::util::concurrent::ConditionImpl	929
decaf::net::ConnectException	931
cms::Connection	
The client's connection to its provider	933
activemq::commands::ConnectionControl	938
activemq::wireformat::openwire::marshal::generated::Connection- ControlMarshaller	
Marshaling code for Open Wire Format for ConnectionControl- Marshaller (p. 943)	943
activemq::commands::ConnectionError	948
activemq::wireformat::openwire::marshal::generated::ConnectionError- Marshaller	
Marshaling code for Open Wire Format for ConnectionError- Marshaller (p. 951)	951
cms::ConnectionFactory	
Defines the interface for a factory that creates connection objects, the Connection (p. 933) objects returned implement the CMS - Connection (p. 933) interface and hide the CMS Provider specific implementation details behind that interface	955
activemq::exceptions::ConnectionFailedException	958
activemq::commands::ConnectionId	960
activemq::wireformat::openwire::marshal::generated::ConnectionId- Marshaller	
Marshaling code for Open Wire Format for ConnectionIdMarshaller (p. 963)	963
activemq::commands::ConnectionInfo	968
activemq::wireformat::openwire::marshal::generated::ConnectionInfo- Marshaller	
Marshaling code for Open Wire Format for ConnectionInfo- Marshaller (p. 974)	974
cms::ConnectionMetaData	
A ConnectionMetaData (p. 978) object provides information describing the Connection (p. 933) object	978
activemq::state::ConnectionState	982
activemq::state::ConnectionStateTracker	984
decaf::util::logging::ConsoleHandler	
This Handler (p. 1401) publishes log records to System.err	990
activemq::commands::ConsumerControl	992
activemq::wireformat::openwire::marshal::generated::Consumer- ControlMarshaller	
Marshaling code for Open Wire Format for ConsumerControl- Marshaller (p. 996)	996
activemq::commands::ConsumerId	1000

activemq::wireformat::openwire::marshal::generated::ConsumerId- Marshaller	
Marshaling code for Open Wire Format for ConsumerIdMarshaller (p. 1005)	1005
activemq::commands::ConsumerInfo	1009
activemq::wireformat::openwire::marshal::generated::ConsumerInfo- Marshaller	
Marshaling code for Open Wire Format for ConsumerInfo- Marshaller (p. 1017)	1017
activemq::state::ConsumerState	1022
activemq::commands::ControlCommand	1022
activemq::wireformat::openwire::marshal::generated::ControlCommand- Marshaller	
Marshaling code for Open Wire Format for ControlCommand- Marshaller (p. 1025)	1025
decaf::util::concurrent::CopyOnWriteArrayList< E >	1030
decaf::util::concurrent::CopyOnWriteArraySet< E >	
Since the CopyOnWriteArraySet (p. 1050) and the CopyOnWrite- ArrayList (p. 1030) share much of the same operational semantics this class uses the CopyOnWriteArrayList (p. 1030) for all its un- derlying operations	1050
decaf::util::concurrent::CountDownLatch	1062
decaf::util::zip::CRC32	
Class that can be used to compute a CRC-32 checksum for a data stream	1065
ct_data_s	1068
activemq::commands::DataArrayResponse	1069
activemq::wireformat::openwire::marshal::generated::DataArrayResponse- Marshaller	
Marshaling code for Open Wire Format for DataArrayResponse- Marshaller (p. 1072)	1072
decaf::util::zip::DataFormatException	1076
decaf::net::DatagramPacket	
Class that represents a single datagram packet	1078
decaf::io::DataInput	
The DataInput (p. 1086) interface provides for reading bytes from a binary stream and reconstructing from them data in any of the C++ primitive types	1086
decaf::io::DataInputStream	
A data input stream lets an application read primitive Java data types from an underlying input stream in a machine-independent way	1094
decaf::io::DataOutput	
The DataOutput (p. 1103) interface provides for converting data from any of the C++ primitive types to a series of bytes and writing these bytes to a binary stream	1103
decaf::io::DataOutputStream	
A data output stream lets an application write primitive Java data types to an output stream in a portable way	1109
activemq::commands::DataResponse	1112

activemq::wireformat::openwire::marshal::generated::DataResponse-Marshaller	
Marshaling code for Open Wire Format for DataResponse-Marshaller (p. 1115)	1115
activemq::wireformat::openwire::marshal::DataStreamMarshaller	
Base class for all classes that marshal commands for Openwire	1119
activemq::commands::DataStructure	1133
decaf::util::Date	
Wrapper class around a time value in milliseconds	1139
decaf::internal::DecafRuntime	
Handles APR initialization and termination	1143
activemq::threads::DedicatedTaskRunner	1144
activemq::core::policies::DefaultPrefetchPolicy	1146
activemq::core::policies::DefaultRedeliveryPolicy	1150
decaf::internal::net::DefaultServerSocketFactory	
Default implementation of the Decaf ServerSocketFactory, creates ServerSocket objects with supplied options	1155
decaf::internal::net::DefaultSocketFactory	
SocketFactory implementation that is used to create Sockets	1159
decaf::internal::net::ssl::DefaultSSLContext	
Default SSLContext manager for the Decaf library	1164
decaf::internal::net::ssl::DefaultSSLServerSocketFactory	
Default implementation of the SSLServerSocketFactory, this factory throws an Exception from all its create methods to indicate that SSL is not supported, this factory is used when OpenSSL is not enabled in the builds	1165
decaf::internal::net::ssl::DefaultSSLSocketFactory	
Default implementation of the SSLSocketFactory, this factory throws an Exception from all its create methods to indicate that SSL is not supported, this factory is used when OpenSSL is not enabled in the builds	1170
activemq::transport::DefaultTransportListener	
A Utility class that create empty implementations for the Transport-Listener (p. 2810) interface so that a subclass only needs to override the one's its interested	1178
decaf::util::zip::Deflater	
This class compresses data using the <i>DEFLATE</i> algorithm (see <i>specification</i>)	1180
decaf::util::zip::DeflaterOutputStream	
Provides a FilterOutputStream instance that compresses the data before writing it to the wrapped OutputStream	1189
decaf::util::concurrent::Delayed	
A mix-in style interface for marking objects that should be acted upon after a given delay	1194
cms::DeliveryMode	
This is an Abstract class whose purpose is to provide a container for the delivery mode enumeration for CMS messages	1195
decaf::util::Deque< E >	
Defines a 'Double ended Queue (p. 2222)' interface that allows for insertion and removal of elements from both ends	1196

cms::Destination	
A Destination (p. 1210) object encapsulates a provider-specific address	1210
activemq::commands::ActiveMQDestination::DestinationFilter	1213
activemq::commands::DestinationInfo	1213
activemq::wireformat::openwire::marshal::generated::DestinationInfo-Marshaller	
Marshaling code for Open Wire Format for DestinationInfo-Marshaller (p. 1218)	1218
activemq::cmsutil::DestinationResolver	
Resolves a CMS destination name to a Destination	1222
decaf::util::concurrent::ThreadPoolExecutor::CallerRunsPolicy::DiscardOldestPolicy	
Handler policy for tasks that are rejected upon a call to ThreadPoolExecutor::execute (p. 2724) this class always destroys the oldest unexecuted task in the Queue (p. 2222) and then attempts to execute the rejected task using the passed in executor	1224
decaf::util::concurrent::ThreadPoolExecutor::CallerRunsPolicy::DiscardPolicy	
Handler policy for tasks that are rejected upon a call to ThreadPoolExecutor::execute (p. 2724) this class always destroys the rejected task and returns quietly	1225
activemq::commands::DiscoveryEvent	1226
activemq::wireformat::openwire::marshal::generated::DiscoveryEvent-Marshaller	
Marshaling code for Open Wire Format for DiscoveryEvent-Marshaller (p. 1230)	1230
activemq::core::DispatchData	
Simple POCO that contains the information necessary to route a message to a specified consumer	1234
activemq::core::Dispatcher	
Interface for an object responsible for dispatching messages to consumers	1234
decaf::lang::Double	1235
decaf::internal::nio::DoubleArrayBuffer	1248
decaf::nio::DoubleBuffer	
This class defines four categories of operations upon double buffers:	1259
decaf::lang::DYNAMIC_CAST_TOKEN	1271
activemq::cmsutil::DynamicDestinationResolver	
Resolves a CMS destination name to a Destination	1272
decaf::util::Map< K, V, COMPARATOR >::Entry	1274
decaf::io::EOFException	1275
decaf::util::logging::ErrorManager	
ErrorManager (p. 1277) objects can be attached to Handlers to process any error that occur on a Handler (p. 1401) during Logging	1277
decaf::lang::Exception	1279
cms::ExceptionListener	
If a CMS provider detects a serious problem, it notifies the client application through an ExceptionListener (p. 1286) that is registered with the Connection (p. 933)	1286

activemq::commands::ExceptionResponse	1287
activemq::wireformat::openwire::marshal::generated::ExceptionResponse-Marshaller	
Marshaling code for Open Wire Format for ExceptionResponse-Marshaller (p. 1290)	1290
decaf::util::concurrent::ExecutionException	1294
decaf::util::concurrent::Executor	
An object that executes submitted decaf.lang Runnable (p. 2312) tasks	1297
decaf::util::concurrent::Executors	
Implements a set of utilities for use with Executors (p. 1299), ExecutorService (p. 1302), ThreadFactory (p. 2714), and Callable (p. 746) types, as well as providing factory methods for instance of these types configured for the most common use cases	1299
decaf::util::concurrent::ExecutorService	
An Executor (p. 1297) that provides methods to manage termination and methods that can produce a Future (p. 1390) for tracking progress of one or more asynchronous tasks	1302
activemq::transport::failover::FailoverTransport	1305
activemq::transport::failover::FailoverTransportFactory	
Creates an instance of a FailoverTransport (p. 1305)	1318
activemq::transport::failover::FailoverTransportListener	
Utility class used by the Transport (p. 2790) to perform the work of responding to events from the active Transport (p. 2790)	1320
activemq::core::FifoMessageDispatchChannel	1322
decaf::io::FileDescriptor	
This class servers as an opaque wrapper around an underlying OS level resource that can be used as a source / sink for byte level data such as sockets and files	1330
decaf::util::logging::Filter	
A Filter (p. 1333) can be used to provide fine grain control over what is logged, beyond the control provided by log levels	1333
decaf::io::FilterInputStream	
A FilterInputStream (p. 1334) contains some other input stream, which it uses as its basic source of data, possibly transforming the data along the way or providing additional functionality	1334
decaf::io::FilterOutputStream	
This class is the superclass of all classes that filter output streams	1341
decaf::lang::Float	1344
decaf::internal::nio::FloatArrayBuffer	1357
decaf::nio::FloatBuffer	
This class defines four categories of operations upon float buffers:	1368
decaf::io::Flushable	
A Flushable (p. 1380) is a destination of data that can be flushed	1380
activemq::commands::FlushCommand	1381
activemq::wireformat::openwire::marshal::generated::FlushCommand-Marshaller	
Marshaling code for Open Wire Format for FlushCommand-Marshaller (p. 1383)	1383

decaf::util::logging::Formatter	
A Formatter (p. 1387) provides support for formatting LogRecords	1387
decaf::util::concurrent::Future< V >	
A Future (p. 1390) represents the result of an asynchronous computation	1390
activemq::transport::correlator::FutureResponse	
A container that holds a response object	1393
decaf::security::GeneralSecurityException	1395
decaf::internal::util::GenericResource< T >	
A Generic Resource (p. 2293) wraps some type and will delete it when the Resource (p. 2293) itself is deleted	1397
gz_header_s	1398
gz_state	1400
decaf::util::logging::Handler	
A Handler (p. 1401) object takes log messages from a Logger (p. 1693) and exports them	1401
decaf::internal::util::HexStringParser	1406
activemq::wireformat::openwire::utils::HexTable	
Maps hexadecimal strings to the value of an index into the table, i.e	1407
decaf::net::HttpRetryException	1408
activemq::util::IdGenerator	1411
decaf::lang::exceptions::IllegalArgumentException	1413
decaf::lang::exceptions::IllegalMonitorStateException	1416
cms::IllegalStateException	
This exception is thrown when a method is invoked at an illegal or inappropriate time or if the provider is not in an appropriate state for the requested operation	1419
decaf::lang::exceptions::IllegalStateException	1420
decaf::lang::exceptions::IllegalThreadStateException	1423
activemq::transport::inactivity::InactivityMonitor	1425
decaf::lang::exceptions::IndexOutOfBoundsException	1429
decaf::net::Inet4Address	1431
decaf::net::Inet6Address	1435
decaf::net::InetAddress	
Represents an IP address	1437
decaf::net::InetSocketAddress	1445
inflate_state	1445
decaf::util::zip::Inflater	
This class uncompresses data that was compressed using the <i>DEFLATE</i> algorithm (see <i>specification</i>)	1448
decaf::util::zip::InflaterInputStream	
A FilterInputStream that decompresses data read from the wrapped InputStream instance	1456
decaf::io::InputStream	
A base class that must be implemented by all classes wishing to provide a class that reads in a stream of bytes	1464
decaf::io::InputStreamReader	
An InputStreamReader (p. 1475) is a bridge from byte streams to character streams	1475
decaf::internal::nio::IntArrayBuffer	1477

decaf::nio::IntBuffer	
This class defines four categories of operations upon int buffers:	1488
decaf::lang::Integer	1500
activemq::commands::IntegerResponse	1516
activemq::wireformat::openwire::marshal::generated::IntegerResponse- Marshaller	
Marshaling code for Open Wire Format for IntegerResponse- Marshaller (p. 1519)	1519
internal_state	1523
activemq::transport::mock::InternalCommandListener	
Listens for Commands sent from the MockTransport (p. 1948)	1527
decaf::lang::exceptions::InterruptedException	1529
decaf::io::InterruptedException	1531
cms::InvalidClientIdException	
This exception must be thrown when a client attempts to set a con- nection's client ID to a value that is rejected by a provider	1534
cms::InvalidDestinationException	
This exception must be thrown when a destination either is not un- derstood by a provider or is no longer valid	1535
decaf::security::InvalidKeyException	1536
decaf::nio::InvalidMarkException	1539
cms::InvalidSelectorException	
This exception must be thrown when a CMS client attempts to give a provider a message selector with invalid syntax	1541
decaf::lang::exceptions::InvalidStateException	1543
decaf::io::IOException	1545
activemq::transport::IOTransport	
Implementation of the Transport (p. 2790) interface that performs marshaling of commands to IO streams	1548
decaf::lang::Iterable< E >	
Implementing this interface allows an object to be cast to an Iterable (p. 1556) type for generic collections API calls	1556
decaf::util::Iterator< E >	
Defines an object that can be used to iterate over the elements of a collection	1559
activemq::commands::JournalQueueAck	1561
activemq::wireformat::openwire::marshal::generated::JournalQueue- AckMarshaller	
Marshaling code for Open Wire Format for JournalQueueAck- Marshaller (p. 1564)	1564
activemq::commands::JournalTopicAck	1568
activemq::wireformat::openwire::marshal::generated::JournalTopicAck- Marshaller	
Marshaling code for Open Wire Format for JournalTopicAck- Marshaller (p. 1572)	1572
activemq::commands::JournalTrace	1577
activemq::wireformat::openwire::marshal::generated::JournalTrace- Marshaller	
Marshaling code for Open Wire Format for JournalTraceMarshaller (p. 1579)	1579

activemq::commands::JournalTransaction	1583
activemq::wireformat::openwire::marshal::generated::JournalTransaction- Marshaller	
Marshaling code for Open Wire Format for JournalTransaction- Marshaller (p. 1587)	1587
activemq::commands::KeepAliveInfo	1591
activemq::wireformat::openwire::marshal::generated::KeepAliveInfo- Marshaller	
Marshaling code for Open Wire Format for KeepAliveInfo- Marshaller (p. 1594)	1594
decaf::security::Key	
The Key (p. 1598) interface is the top-level interface for all keys . . .	1598
decaf::security::KeyException	1600
decaf::security::KeyManagementException	1603
activemq::commands::LastPartialCommand	1606
activemq::wireformat::openwire::marshal::generated::LastPartial- CommandMarshaller	
Marshaling code for Open Wire Format for LastPartialCommand- Marshaller (p. 1608)	1608
decaf::util::comparators::Less< E >	
Simple Less (p. 1612) Comparator (p. 888) that compares to ele- ments to determine if the first is less than the second	1612
std::less< decaf::lang::ArrayPointer< T > >	
An override of the less function object so that the Pointer objects can be stored in STL Maps, etc	1614
std::less< decaf::lang::Pointer< T > >	
An override of the less function object so that the Pointer objects can be stored in STL Maps, etc	1615
decaf::util::logging::Level	
Defines a set of standard logging levels that can be used to control logging output	1616
decaf::util::concurrent::LinkedBlockingQueue< E >	
A BlockingQueue (p. 538) derivative that allows for a bound to be placed on the number of elements that can be enqueued at any one time	1621
decaf::util::LinkedList< E >	
A complete implementation of the List (p. 1658) interface using a doubly linked list data structure	1633
decaf::util::List< E >	
An ordered collection (also known as a sequence)	1658
decaf::util::ListIterator< E >	
An iterator for lists that allows the programmer to traverse the list in either direction, modify the list during iteration, and obtain the itera- tor's current position in the list	1671
activemq::commands::LocalTransactionId	1674
activemq::wireformat::openwire::marshal::generated::LocalTransaction- IdMarshaller	
Marshaling code for Open Wire Format for LocalTransactionId- Marshaller (p. 1678)	1678

decaf::util::concurrent::Lock	
A wrapper class around a given synchronization mechanism that provides automatic release upon destruction	1682
decaf::util::concurrent::locks::Lock	
Lock (p. 1684) implementations provide more extensive locking operations than can be obtained using synchronized statements	1684
decaf::util::concurrent::locks::LockSupport	
Basic thread blocking primitives for creating locks and other synchronization classes	1690
decaf::util::logging::Logger	
A Logger (p. 1693) object is used to log messages for a specific system or application component	1693
decaf::util::logging::LoggerHierarchy	1706
activemq::io::LoggingInputStream	1706
activemq::io::LoggingOutputStream	
OutputStream filter that just logs the data being written	1707
activemq::transport::logging::LoggingTransport	
A transport filter that logs commands as they are sent/received . . .	1709
decaf::util::logging::LogManager	
There is a single global LogManager (p. 1712) object that is used to maintain a set of shared state about Loggers and log services . . .	1712
decaf::util::logging::LogRecord	
LogRecord (p. 1719) objects are used to pass logging requests between the logging framework and individual log Handlers	1719
decaf::util::logging::LogWriter	1724
decaf::lang::Long	1726
decaf::internal::nio::LongArrayBuffer	1742
decaf::nio::LongBuffer	
This class defines four categories of operations upon long long buffers:	1752
activemq::util::LongSequenceGenerator	
This class is used to generate a sequence of long long values that are incremented each time a new value is requested	1765
decaf::net::MalformedURLException	1766
decaf::util::Map< K, V, COMPARATOR >	
Map (p. 1768) template that wraps around a std::map to provide a more user-friendly interface and to provide common functions that do not exist in std::map	1768
cms::MapMessage	
A MapMessage (p. 1779) object is used to send a set of name-value pairs	1779
decaf::util::logging::MarkBlockLogger	
Defines a class that can be used to mark the entry and exit from scoped blocks	1791
activemq::wireformat::MarshalAware	1792
activemq::wireformat::openwire::marshal::generated::MarshallerFactory	
Used to create marshallers for a specific version of the wire protocol	1795
activemq::util::MarshallingSupport	1796

decaf::lang::Math	
The class Math (p. 1800) contains methods for performing basic numeric operations such as the elementary exponential, logarithm, square root, and trigonometric functions	1800
activemq::util::MemoryUsage	1818
activemq::commands::Message	1821
cms::Message	
Root of all messages	1839
activemq::commands::MessageAck	1867
activemq::wireformat::openwire::marshal::generated::MessageAck-Marshaller	
Marshaling code for Open Wire Format for MessageAckMarshaller (p. 1873)	1873
cms::MessageConsumer	
A client uses a MessageConsumer (p. 1877) to received messages from a destination	1877
activemq::cmsutil::MessageCreator	
Creates the user-defined message to be sent by the Cms-Template (p. 836)	1880
activemq::commands::MessageDispatch	1881
activemq::core::MessageDispatchChannel	1886
activemq::wireformat::openwire::marshal::generated::MessageDispatch-Marshaller	
Marshaling code for Open Wire Format for MessageDispatch-Marshaller (p. 1890)	1890
activemq::commands::MessageDispatchNotification	1894
activemq::wireformat::openwire::marshal::generated::MessageDispatch-NotificationMarshaller	
Marshaling code for Open Wire Format for MessageDispatch-NotificationMarshaller (p. 1899)	1899
cms::MessageEnumeration	
Defines an object that enumerates a collection of Messages	1903
cms::MessageEOFException	
This exception must be thrown when an unexpected end of stream has been reached when a StreamMessage (p. 2606) or Bytes-Message (p. 718) is being read	1905
cms::MessageFormatException	
This exception must be thrown when a CMS client attempts to use a data type not supported by a message or attempts to read data in a message as the wrong type	1906
activemq::commands::Messageld	1908
activemq::wireformat::openwire::marshal::generated::Messageld-Marshaller	
Marshaling code for Open Wire Format for MessageldMarshaller (p. 1912)	1912
cms::MessageListener	
A MessageListener (p. 1916) object is used to receive asynchronously delivered messages	1916

activemq::wireformat::openwire::marshal::generated::MessageMarshaller	
Marshaling code for Open Wire Format for MessageMarshaller	
(p. 1917)	1917
cms::MessageNotReadableException	
This exception must be thrown when a CMS client attempts to read	
a write-only message	1922
cms::MessageNotWriteableException	
This exception must be thrown when a CMS client attempts to write	
to a read-only message	1923
cms::MessageProducer	
A client uses a MessageProducer (p. 1924) object to send	
messages to a Destination (p. 1210)	1924
activemq::wireformat::openwire::utils::MessagePropertyInterceptor	
Used the base ActiveMQMessage class to intercept calls to get and	
set properties in order to capture the calls that use the reserved JMS	
properties and get and set them in the OpenWire Message proper-	
ties	1933
activemq::commands::MessagePull	1939
activemq::wireformat::openwire::marshal::generated::MessagePull-	
Marshaller	
Marshaling code for Open Wire Format for MessagePullMarshaller	
(p. 1944)	1944
activemq::transport::mock::MockTransport	
The MockTransport (p. 1948) defines a base level Transport	
(p. 2790) class that is intended to be used in place of an a regular	
protocol Transport (p. 2790) such as TCP	1948
activemq::transport::mock::MockTransportFactory	
Manufactures MockTransports, which are objects that read from in-	
put streams and write to output streams	1958
decaf::util::concurrent::Mutex	
Mutex (p. 1960) object that offers recursive support on all platforms	
as well as providing the ability to use the standard wait / notify pattern	
used in languages like Java	1960
decaf::util::concurrent::MutexHandle	1965
decaf::internal::util::concurrent::MutexImpl	1966
decaf::internal::net::Network	
Internal class used to manage Networking related resources and	
hide platform dependent calls from the higher level API	1968
activemq::commands::NetworkBridgeFilter	1971
activemq::wireformat::openwire::marshal::generated::NetworkBridge-	
FilterMarshaller	
Marshaling code for Open Wire Format for NetworkBridgeFilter-	
Marshaller (p. 1974)	1974
decaf::net::NoRouteToHostException	1978
decaf::security::NoSuchAlgorithmException	1981
decaf::util::NoSuchElementException	1984
decaf::security::NoSuchProviderException	1986
decaf::lang::exceptions::NullPointerException	1989

decaf::lang::Number	
The abstract class Number (p. 1992) is the superclass of classes - Byte (p. 614), Double (p. 1235), Float (p. 1344), Integer (p. 1500), Long (p. 1726), and Short (p. 2398)	1992
decaf::lang::exceptions::NumberFormatException	1994
cms::ObjectMessage	
Place holder for interaction with JMS systems that support Java, the C++ client is not responsible for deserializing the contained Object	1997
decaf::internal::net::ssl::openssl::OpenSSLContextSpi	
Provides an SSLContext that wraps the OpenSSL API	1998
decaf::internal::net::ssl::openssl::OpenSSLParameters	
Container class for parameters that are Common to OpenSSL socket classes	2001
decaf::internal::net::ssl::openssl::OpenSSLServerSocket	
SSLServerSocket based on OpenSSL library code	2003
decaf::internal::net::ssl::openssl::OpenSSLServerSocketFactory	
SSLServerSocketFactory that creates Server Sockets that use - OpenSSL	2009
decaf::internal::net::ssl::openssl::OpenSSLSocket	
Wraps a a Normal Socket object and extends or overrides functions in that class to make use of the OpenSSL Socket API	2014
decaf::internal::net::ssl::openssl::OpenSSLSocketException	
Subclass of the standard SocketException that knows how to pro- duce an error message from the OpenSSL error stack	2029
decaf::internal::net::ssl::openssl::OpenSSLSocketFactory	
Client Socket Factory that creates SSL based client sockets using the OpenSSL library	2032
decaf::internal::net::ssl::openssl::OpenSSLSocketInputStream	
An output stream for reading data from an OpenSSL Socket instance	2040
decaf::internal::net::ssl::openssl::OpenSSLSocketOutputStream	
OutputStream implementation used to write data to an OpenSSL- Socket (p. 2014) instance	2043
activemq::wireformat::openwire::OpenWireFormat	2045
activemq::wireformat::openwire::OpenWireFormatFactory	2059
activemq::wireformat::openwire::OpenWireFormatNegotiator	2060
activemq::wireformat::openwire::OpenWireResponseBuilder	
Used to allow a MockTransport to generate response commands to OpenWire Commands	2064
decaf::io::OutputStream	
Base interface for any class that wants to represent an output stream of bytes	2066
decaf::io::OutputStreamWriter	
A class for turning a character stream into a byte stream	2074
activemq::commands::PartialCommand	2076
activemq::wireformat::openwire::marshal::generated::PartialCommand- Marshaller	
Marshaling code for Open Wire Format for PartialCommand- Marshaller (p. 2079)	2079

decaf::lang::Pointer< T, REFCOUNTER >	
Decaf's implementation of a Smart Pointer (p.2083) that is a template on a Type and is Thread (p.2703) Safe if the default Reference Counter is used	2083
decaf::lang::PointerComparator< T, R >	
This implementation of Comparator is designed to allows objects in a Collection to be sorted or tested for equality based on the value of the Object being Pointed to and not the value of the contained pointer in the Pointer (p.2083) instance	2091
activemq::cmsutil::PooledSession	
A pooled session object that wraps around a delegate session	2092
decaf::net::PortUnreachableException	2107
activemq::core::PrefetchPolicy	
Interface for a Policy object that controls message Prefetching on various destination types in ActiveMQ-CPP	2110
activemq::util::PrimitiveList	
List of primitives	2114
activemq::util::PrimitiveMap	
Map of named primitives	2125
activemq::wireformat::openwire::marshal::PrimitiveTypesMarshaller	
This class wraps the functionality needed to marshal a primitive map to the Openwire Format's expectation of what the map looks like on the wire	2135
activemq::util::PrimitiveValueNode::PrimitiveValue	
Define a union type comprised of the various types	2142
activemq::util::PrimitiveValueConverter	
Class controls the conversion of data contained in a PrimitiveValueNode (p.2145) from one type to another	2144
activemq::util::PrimitiveValueNode	
Class that wraps around a single value of one of the many types . . .	2145
decaf::security::Principal	
Base interface for a principal, which can represent an individual or organization	2159
decaf::util::PriorityQueue< E >	
An unbounded priority queue based on a binary heap algorithm . . .	2160
activemq::commands::ProducerAck	2171
activemq::wireformat::openwire::marshal::generated::ProducerAck-Marshaller	
Marshaling code for Open Wire Format for ProducerAckMarshaller (p.2175)	2175
activemq::cmsutil::ProducerCallback	
Callback for sending a message to a CMS destination	2179
activemq::cmsutil::CmsTemplate::ProducerExecutor	2180
activemq::commands::ProducerId	2182
activemq::wireformat::openwire::marshal::generated::ProducerId-Marshaller	
Marshaling code for Open Wire Format for ProducerIdMarshaller (p.2186)	2186
activemq::commands::ProducerInfo	2190

activemq::wireformat::openwire::marshal::generated::ProducerInfo- Marshaller	
Marshaling code for Open Wire Format for ProducerInfoMarshaller (p. 2195)	2195
activemq::state::ProducerState	2199
decaf::util::Properties	
Java-like properties class for mapping string names to string values	2200
decaf::util::logging::PropertiesChangeListener	
Defines the interface that classes can use to listen for change events on Properties (p. 2200)	2210
decaf::net::ProtocolException	2211
decaf::security::PublicKey	
A public key	2213
decaf::io::PushbackInputStream	
A PushbackInputStream (p. 2214) adds functionality to another in- put stream, namely the ability to "push back" or "unread" one byte	2214
cms::Queue	
An interface encapsulating a provider-specific queue name	2221
decaf::util::Queue< E >	
A kind of collection provides advanced operations than other basic collections, such as insertion, extraction, and inspection	2222
cms::QueueBrowser	
This class implements in interface for browsing the messages in a Queue (p. 2221) without removing them	2227
decaf::util::Random	
Random (p. 2229) Value Generator which is used to generate a stream of pseudorandom numbers	2229
decaf::lang::Readable	
A Readable (p. 2235) is a source of characters	2235
activemq::transport::inactivity::ReadChecker	
Runnable class that is used by the {	2236
decaf::io::Reader	2237
decaf::nio::ReadOnlyBufferException	2244
decaf::util::concurrent::locks::ReadWriteLock	
A ReadWriteLock (p. 2246) maintains a pair of associated locks, one for read-only operations and one for writing	2246
activemq::cmsutil::CmsTemplate::ReceiveExecutor	2248
activemq::core::RedeliveryPolicy	
Interface for a RedeliveryPolicy (p. 2250) object that controls how message Redelivery is handled in ActiveMQ-CPP when a transac- tion is rolled back	2250
decaf::util::concurrent::locks::ReentrantLock	
A reentrant mutual exclusion Lock (p. 1684) with extended capabili- ties	2256
decaf::util::concurrent::RejectedExecutionException	2263
decaf::util::concurrent::RejectedExecutionHandler	
A handler for tasks that cannot be executed by a ThreadPool- Executor (p. 2715)	2266
activemq::commands::RemoveInfo	2267

activemq::wireformat::openwire::marshal::generated::RemoveInfo- Marshaller	
Marshaling code for Open Wire Format for RemoveInfoMarshaller (p. 2271)	2271
activemq::commands::RemoveSubscriptionInfo	2275
activemq::wireformat::openwire::marshal::generated::RemoveSubscription- InfoMarshaller	
Marshaling code for Open Wire Format for RemoveSubscription- InfoMarshaller (p. 2280)	2280
activemq::commands::ReplayCommand	2284
activemq::wireformat::openwire::marshal::generated::ReplayCommand- Marshaller	
Marshaling code for Open Wire Format for ReplayCommand- Marshaller (p. 2287)	2287
activemq::cmsutil::CmsTemplate::ResolveProducerExecutor	2291
activemq::cmsutil::CmsTemplate::ResolveReceiveExecutor	2292
decaf::internal::util::Resource	
Interface for all Managed Resources in Decaf, these objects are added to the Runtime System and are destroyed at shutdown	2293
activemq::cmsutil::ResourceLifecycleManager	
Manages the lifecycle of a set of CMS resources	2293
decaf::internal::util::ResourceLifecycleManager	2297
activemq::commands::Response	2298
activemq::transport::mock::ResponseBuilder	
Interface for all Protocols to implement that defines the behavior of the Broker in response to messages of that protocol	2301
activemq::transport::correlator::ResponseCorrelator	
This type of transport filter is responsible for correlating asyn- chronous responses with requests	2303
activemq::wireformat::openwire::marshal::generated::ResponseMarshaller	
Marshaling code for Open Wire Format for ResponseMarshaller (p. 2307)	2307
decaf::lang::Runnable	
Interface for a runnable object - defines a task that can be run by a thread	2312
decaf::lang::Runtime	2313
decaf::lang::exceptions::RuntimeException	2315
activemq::threads::Scheduler	
Scheduler (p. 2317) class for use in executing Runnable Tasks either periodically or one time only with optional delay	2317
activemq::threads::SchedulerTimerTask	
Extension of the Decaf TimerTask that adds a Runnable instance which is the target of this task	2319
decaf::security::SecureRandom	2320
decaf::internal::security::SecureRandomImpl	
Secure Random Number Generator for Unix based platforms that at- tempts to obtain secure bytes with high entropy from known sources .	2325
decaf::security::SecureRandomSpi	
Interface class used by Security Service Providers to implement a source of secure random bytes	2329

decaf::util::concurrent::Semaphore	
A counting semaphore	2331
activemq::cmsutil::CmsTemplate::SendExecutor	2341
decaf::net::ServerSocket	
This class implements server sockets	2342
decaf::net::ServerSocketFactory	
Class used to create Server Sockets, subclasses can be created that create certain types of server sockets according to specific policies .	2352
activemq::util::Service	
Base interface for all classes that run as a Service (p. 2355) inside the application	2355
activemq::util::ServiceListener	
Listener interface for observers of Service (p. 2355) related events .	2356
activemq::util::ServiceStopper	2358
activemq::util::ServiceSupport	
Provides a base class for Service (p. 2355) implementations	2358
cms::Session	
A Session (p. 2361) object is a single-threaded context for producing and consuming messages	2361
activemq::cmsutil::SessionCallback	
Callback for executing any number of operations on a provided CMS Session	2377
activemq::commands::SessionId	2378
activemq::wireformat::openwire::marshal::generated::SessionIdMarshaller	
Marshaling code for Open Wire Format for SessionIdMarshaller (p. 2382)	2382
activemq::commands::SessionInfo	2386
activemq::wireformat::openwire::marshal::generated::SessionInfo- Marshaller	
Marshaling code for Open Wire Format for SessionInfoMarshaller (p. 2390)	2390
activemq::cmsutil::SessionPool	
A pool of CMS sessions from the same connection and with the same acknowledge mode	2394
activemq::state::SessionState	2395
decaf::util::Set< E >	
A collection that contains no duplicate elements	2397
decaf::lang::Short	2398
decaf::internal::nio::ShortArrayBuffer	2408
decaf::nio::ShortBuffer	
This class defines four categories of operations upon short buffers: .	2419
activemq::commands::ShutdownInfo	2431
activemq::wireformat::openwire::marshal::generated::ShutdownInfo- Marshaller	
Marshaling code for Open Wire Format for ShutdownInfo- Marshaller (p. 2434)	2434
decaf::security::SignatureException	2438
decaf::util::logging::SimpleFormatter	
Print a brief summary of the LogRecord (p. 1719) in a human read- able format	2441

decaf::util::logging::SimpleLogger	2442
activemq::core::SimplePriorityMessageDispatchChannel	2444
decaf::net::Socket	2452
decaf::net::SocketAddress	
Base class for protocol specific Socket (p. 2452) addresses	2470
decaf::net::SocketError	
Static utility class to simplify handling of error codes for socket operations	2470
decaf::net::SocketException	
Exception for errors when manipulating sockets	2471
decaf::net::SocketFactory	
The SocketFactory (p. 2473) is used to create Socket (p. 2452) objects and can be sub-classed to provide other types of Sockets or Sockets with varying configurations	2473
decaf::internal::net::SocketFileDescriptor	
File Descriptor type used internally by Decaf Socket objects	2478
decaf::net::SocketImpl	
Acts as a base class for all physical Socket (p. 2452) implementations	2479
decaf::net::SocketImplFactory	
Factory class interface for a Factory that creates SocketImpl objects	2487
decaf::net::SocketOptions	2488
decaf::net::SocketTimeoutException	2493
decaf::net::ssl::SSLContext	
Represents on implementation of the Secure Socket (p. 2452) Layer for streaming based sockets	2495
decaf::net::ssl::SSLContextSpi	
Defines the interface that should be provided by an SSLContext (p. 2495) provider	2498
decaf::net::ssl::SSLParameters	2501
decaf::net::ssl::SSLServerSocket	
Represents a server socket that is used to accept connections from clients using the Secure Sockets protocol or the Top Level Security protocol	2504
decaf::net::ssl::SSLServerSocketFactory	
Factory class interface that provides methods to create SSL Server Sockets	2510
decaf::net::ssl::SSLSocket	2513
decaf::net::ssl::SSLSocketFactory	
Factory class interface for a SocketFactory (p. 2473) that can create SSLSocket (p. 2513) objects	2522
activemq::transport::tcp::SslTransport	
Transport (p. 2790) for connecting to a Broker using an SSL Socket	2525
activemq::transport::tcp::SslTransportFactory	2527
activemq::commands::BrokerError::StackTraceElement	2527
decaf::internal::io::StandardErrorOutputStream	
Wrapper Around the Standard error Output facility on the current platform	2528
decaf::internal::io::StandardInputStream	2530
decaf::internal::io::StandardOutputStream	2532

cms::Startable	
Interface for a class that implements the start method	2534
decaf::lang::STATIC_CAST_TOKEN	2535
activemq::core::ActiveMQConstants::StaticInitializer	2535
decaf::util::StlList< E >	
List (p. 1658) class that wraps the STL list object to provide a simpler interface and additional methods not provided by the STL type	2536
decaf::util::StlMap< K, V, COMPARATOR >	
Map (p. 1768) template that wraps around a std::map to provide a more user-friendly interface and to provide common functions that do not exist in std::map	2552
decaf::util::StlQueue< T >	
The Queue (p. 2222) class accepts messages with an psuh(m) command where m is the message to be queued	2565
decaf::util::StlSet< E >	
Set (p. 2397) template that wraps around a std::set to provide a more user-friendly interface and to provide common functions that do not exist in std::set	2574
activemq::wireformat::stomp::StompCommandConstants	2581
activemq::wireformat::stomp::StompFrame	
A Stomp-level message frame that encloses all messages to and from the broker	2587
activemq::wireformat::stomp::StompHelper	
Utility Methods used when marshaling to and from StompFrame (p. 2587)'s	2592
activemq::wireformat::stomp::StompWireFormat	2597
activemq::wireformat::stomp::StompWireFormatFactory	
Factory used to create the Stomp Wire Format instance	2601
cms::Stoppable	
Interface for a class that implements the stop method	2602
decaf::util::logging::StreamHandler	
Stream based logging Handler (p. 1401)	2603
cms::StreamMessage	
Interface for a StreamMessage (p. 2606)	2606
decaf::lang::String	
Immutable sequence of chars	2620
decaf::util::StringTokenizer	
Class that allows for parsing of string based on Tokens	2627
activemq::commands::SubscriptionInfo	2631
activemq::wireformat::openwire::marshal::generated::SubscriptionInfo-Marshaller	
Marshaling code for Open Wire Format for SubscriptionInfo-Marshaller (p. 2635)	2635
decaf::util::concurrent::Synchronizable	
The interface for all synchronizable objects (that is, objects that can be locked and unlocked)	2639
decaf::internal::util::concurrent::SynchronizableImpl	
A convenience class used by some Decaf classes to implement the Synchronizable interface when there is no issues related to multiple inheritance	2650

activemq::core::Synchronization	
Transacted Object Synchronization (p.2654), used to sync the events of a Transaction with the items in the Transaction	2654
decaf::util::concurrent::SynchronousQueue< E >	
A blocking queue (p.538) in which each insert operation must wait for a corresponding remove operation by another thread, and vice versa	2655
decaf::lang::System	
Static methods for accessing system level resources and performing some system dependent tasks such as looking up environment values and copying memory and arrays	2665
activemq::threads::Task	
Represents a unit of work that requires one or more iterations to complete	2676
activemq::threads::TaskRunner	2677
decaf::internal::net::tcp::TcpSocket	
Platform-independent implementation of the socket interface	2678
decaf::internal::net::tcp::TcpSocketInputStream	
Input stream for performing reads on a socket	2688
decaf::internal::net::tcp::TcpSocketOutputStream	
Output stream for performing write operations on a socket	2691
activemq::transport::tcp::TcpTransport	
Implements a TCP/IP based transport filter, this transport is meant to wrap an instance of an IOTransport (p.1548)	2693
activemq::transport::tcp::TcpTransportFactory	
Factory Responsible for creating the TcpTransport (p.2693)	2696
cms::TemporaryQueue	
Defines a Temporary Queue (p.2221) based Destination (p.1210)	2698
cms::TemporaryTopic	
Defines a Temporary Topic (p.2764) based Destination (p.1210)	2700
cms::TextMessage	
Interface for a text message	2701
decaf::lang::Thread	
A Thread (p.2703) is a concurrent unit of execution	2703
decaf::util::concurrent::ThreadFactory	
Public interface ThreadFactory (p.2714)	2714
decaf::lang::ThreadGroup	2715
decaf::util::concurrent::ThreadPoolExecutor	
Defines a Thread Pool object that implements the functionality of pooling threads to perform user tasks	2715
decaf::lang::Throwable	
This class represents an error that has occurred	2732
decaf::util::concurrent::TimeoutException	2737
decaf::util::Timer	
A facility for threads to schedule tasks for future execution in a background thread	2739
decaf::util::TimerTask	
A Base class for a task object that can be scheduled for one-time or repeated execution by a Timer (p.2739)	2751

decaf::internal::util::TimerTaskHeap	
A Binary Heap implemented specifically for the Timer class in Decaf	
Util	2753
decaf::util::concurrent::TimeUnit	
A TimeUnit (p. 2756) represents time durations at a given unit of granularity and provides utility methods to convert across units, and to perform timing and delay operations in these units	2756
cms::Topic	
An interface encapsulating a provider-specific topic name	2764
activemq::state::Tracked	2765
activemq::commands::TransactionId	2766
activemq::wireformat::openwire::marshal::generated::TransactionId-Marshaller	
Marshaling code for Open Wire Format for TransactionIdMarshaller (p. 2770)	2770
activemq::commands::TransactionInfo	2774
activemq::wireformat::openwire::marshal::generated::TransactionInfo-Marshaller	
Marshaling code for Open Wire Format for TransactionInfo-Marshaller (p. 2778)	2778
cms::TransactionInProgressException	
This exception is thrown when an operation is invalid because a transaction is in progress	2782
cms::TransactionRolledBackException	
This exception must be thrown when a call to Session.commit (p. 2366) results in a rollback of the current transaction	2783
activemq::state::TransactionState	2785
decaf::internal::util::concurrent::Transferer< E >	
Shared internal API for dual stacks and queues	2786
decaf::internal::util::concurrent::TransferQueue< E >	
This extends Scherer-Scott dual queue algorithm, differing, among other ways, by using modes within nodes rather than marked pointers	2787
decaf::internal::util::concurrent::TransferStack< E >	2789
activemq::transport::Transport	
Interface for a transport layer for command objects	2790
activemq::transport::TransportFactory	
Defines the interface for Factories that create Transports or - TransportFilters	2798
activemq::transport::TransportFilter	
A filter on the transport layer	2800
activemq::transport::TransportListener	
A listener of asynchronous exceptions from a command transport object	2810
activemq::transport::TransportRegistry	
Registry of all Transport (p. 2790) Factories that are available to the client at runtime	2812
tree_desc_s	2815

decaf::lang::Thread::UncaughtExceptionHandler	
Interface for handlers invoked when a Thread (p. 2703) abruptly terminates due to an uncaught exception	2815
decaf::net::UnknownHostException	2816
decaf::net::UnknownServiceException	2819
decaf::io::UnsupportedEncodingException	
Thrown when the the Character Encoding is not supported	2821
decaf::lang::exceptions::UnsupportedOperationException	2824
cms::UnsupportedOperationException	
This exception must be thrown when a CMS client attempts use a CMS method that is not implemented or not supported by the CMS Provider in use	2827
decaf::net::URI	
This class represents an instance of a URI (p. 2828) as defined by RFC 2396	2828
decaf::internal::net::URLEncoderDecoder	2841
decaf::internal::net::URIHelper	
Helper class used by the URI classes in encoding and decoding of URI's	2844
activemq::transport::failover::URIPool	2851
activemq::util::URISupport	2854
decaf::net::URISyntaxException	2856
decaf::internal::net::URIType	
Basic type object that holds data that composes a given URI	2860
decaf::net::URL	
Class URL (p. 2868) represents a Uniform Resource Locator, a pointer to a "resource" on the World Wide Web	2868
decaf::net::URLDecoder	2870
decaf::net::URLEncoder	2871
activemq::util::Usage	2872
decaf::io::UTFDataFormatException	
Thrown from classes that attempt to read or write a UTF-8 encoded string and an encoding error is encountered	2874
decaf::util::UUID	
A class that represents an immutable universally unique identifier (- UUID (p. 2877))	2877
activemq::wireformat::WireFormat	
Provides a mechanism to marshal commands into and out of packets or into and out of streams, Channels and Datagrams	2884
activemq::wireformat::WireFormatFactory	
The WireFormatFactory (p. 2888) is the interface that all WireFormatFactory (p. 2888) classes must extend	2888
activemq::commands::WireFormatInfo	2889
activemq::wireformat::openwire::marshal::generated::WireFormatInfo-Marshaller	
Marshaling code for Open Wire Format for WireFormatInfo-Marshaller (p. 2900)	2900
activemq::wireformat::WireFormatNegotiator	
Defines a WireFormatNegotiator (p. 2904) which allows a WireFormat (p. 2884) to	2904

activemq::wireformat::WireFormatRegistry	
Registry of all WireFormat (p. 2884) Factories that are available to the client at runtime	2905
activemq::transport::inactivity::WriteChecker	
Runnable class used by the {	2907
decaf::io::Writer	2908
decaf::security::auth::x500::X500Principal	2914
decaf::security::cert::X509Certificate	
Base interface for all identity certificates	2915
cms::XAConnection	
The XAConnection (p. 2917) interface defines an extended - Connection (p. 933) type that is used to create XASession (p. 2935) objects	2917
cms::XAConnectionFactory	
The XAConnectionFactory (p. 2918) interface is specialized interface that defines an ConnectionFactory (p. 955) that creates - Connection (p. 933) instance that will participate in XA Transactions	2918
cms::XAException	
The XAException (p. 2921) is thrown by the Resource Manager (R-M) to inform the Transaction Manager of an error encountered by the involved transaction	2921
cms::XAResource	
The XAResource (p. 2926) interface is a CMS mapping of the industry standard XA interface based on the X/Open CAE Specification (Distributed Transaction Processing: The XA Specification)	2926
cms::XASession	
The XASession (p. 2935) interface extends the capability of - Session (p. 2361) by adding access to a CMS provider's support for the operating inside an XA Transaction (optional)	2935
activemq::commands::XATransactionId	2936
activemq::wireformat::openwire::marshal::generated::XATransactionId-Marshaller	
Marshaling code for Open Wire Format for XATransactionId-Marshaller (p. 2942)	2942
cms::Xid	
An interface which provides a mapping for the X/Open XID transaction identifier structure	2946
decaf::util::logging::XMLFormatter	
Format a LogRecord (p. 1719) into a standard XML format	2950
z_stream_s	2952
decaf::util::zip::ZipException	2953

Chapter 4

File Index

4.1 File List

Here is a list of all files with brief descriptions:

src/main/activemq/cmsutil/ CachedConsumer.h	2957
src/main/activemq/cmsutil/ CachedProducer.h	2957
src/main/activemq/cmsutil/ CmsAccessor.h	2958
src/main/activemq/cmsutil/ CmsDestinationAccessor.h	2958
src/main/activemq/cmsutil/ CmsTemplate.h	2959
src/main/activemq/cmsutil/ DestinationResolver.h	2959
src/main/activemq/cmsutil/ DynamicDestinationResolver.h	2960
src/main/activemq/cmsutil/ MessageCreator.h	2960
src/main/activemq/cmsutil/ PooledSession.h	2961
src/main/activemq/cmsutil/ ProducerCallback.h	2961
src/main/activemq/cmsutil/ ResourceLifecycleManager.h	2962
src/main/activemq/cmsutil/ SessionCallback.h	2962
src/main/activemq/cmsutil/ SessionPool.h	2963
src/main/activemq/commands/ ActiveMQBlobMessage.h	2963
src/main/activemq/commands/ ActiveMQBytesMessage.h	2964
src/main/activemq/commands/ ActiveMQDestination.h	2964
src/main/activemq/commands/ ActiveMQMapMessage.h	2965
src/main/activemq/commands/ ActiveMQMessage.h	2965
src/main/activemq/commands/ ActiveMQMessageTemplate.h	2966
src/main/activemq/commands/ ActiveMQObjectMessage.h	2966
src/main/activemq/commands/ ActiveMQQueue.h	2967
src/main/activemq/commands/ ActiveMQStreamMessage.h	2967
src/main/activemq/commands/ ActiveMQTempDestination.h	2968
src/main/activemq/commands/ ActiveMQTempQueue.h	2968
src/main/activemq/commands/ ActiveMQTempTopic.h	2969
src/main/activemq/commands/ ActiveMQTextMessage.h	2969
src/main/activemq/commands/ ActiveMQTopic.h	2970
src/main/activemq/commands/ BaseCommand.h	2970
src/main/activemq/commands/ BaseDataStructure.h	2971

src/main/activemq/commands/ BooleanExpression.h	2971
src/main/activemq/commands/ BrokerError.h	2971
src/main/activemq/commands/ BrokerId.h	2972
src/main/activemq/commands/ BrokerInfo.h	2972
src/main/activemq/commands/ Command.h	2973
src/main/activemq/commands/ ConnectionControl.h	2973
src/main/activemq/commands/ ConnectionError.h	2974
src/main/activemq/commands/ ConnectionId.h	2974
src/main/activemq/commands/ ConnectionInfo.h	2974
src/main/activemq/commands/ ConsumerControl.h	2975
src/main/activemq/commands/ ConsumerId.h	2975
src/main/activemq/commands/ ConsumerInfo.h	2976
src/main/activemq/commands/ ControlCommand.h	2976
src/main/activemq/commands/ DataArrayResponse.h	2977
src/main/activemq/commands/ DataResponse.h	2977
src/main/activemq/commands/ DataStructure.h	2978
src/main/activemq/commands/ DestinationInfo.h	2978
src/main/activemq/commands/ DiscoveryEvent.h	2978
src/main/activemq/commands/ ExceptionResponse.h	2979
src/main/activemq/commands/ FlushCommand.h	2979
src/main/activemq/commands/ IntegerResponse.h	2980
src/main/activemq/commands/ JournalQueueAck.h	2980
src/main/activemq/commands/ JournalTopicAck.h	2981
src/main/activemq/commands/ JournalTrace.h	2981
src/main/activemq/commands/ JournalTransaction.h	2982
src/main/activemq/commands/ KeepAliveInfo.h	2982
src/main/activemq/commands/ LastPartialCommand.h	2982
src/main/activemq/commands/ LocalTransactionId.h	2983
src/main/activemq/commands/ Message.h	2983
src/main/activemq/commands/ MessageAck.h	2984
src/main/activemq/commands/ MessageDispatch.h	2985
src/main/activemq/commands/ MessageDispatchNotification.h	2985
src/main/activemq/commands/ MessageId.h	2986
src/main/activemq/commands/ MessagePull.h	2986
src/main/activemq/commands/ NetworkBridgeFilter.h	2987
src/main/activemq/commands/ PartialCommand.h	2987
src/main/activemq/commands/ ProducerAck.h	2988
src/main/activemq/commands/ ProducerId.h	2988
src/main/activemq/commands/ ProducerInfo.h	2988
src/main/activemq/commands/ RemoveInfo.h	2989
src/main/activemq/commands/ RemoveSubscriptionInfo.h	2989
src/main/activemq/commands/ ReplayCommand.h	2990
src/main/activemq/commands/ Response.h	2990
src/main/activemq/commands/ SessionId.h	2991
src/main/activemq/commands/ SessionInfo.h	2991
src/main/activemq/commands/ ShutdownInfo.h	2992
src/main/activemq/commands/ SubscriptionInfo.h	2992
src/main/activemq/commands/ TransactionId.h	2992
src/main/activemq/commands/ TransactionInfo.h	2993
src/main/activemq/commands/ WireFormatInfo.h	2993

src/main/activemq/commands/XATransactionId.h	2994
src/main/activemq/core/ActiveMQAckHandler.h	2994
src/main/activemq/core/ActiveMQConnection.h	2995
src/main/activemq/core/ActiveMQConnectionFactory.h	2995
src/main/activemq/core/ActiveMQConnectionMetaData.h	2996
src/main/activemq/core/ActiveMQConstants.h	2996
src/main/activemq/core/ActiveMQConsumer.h	2997
src/main/activemq/core/ActiveMQProducer.h	2997
src/main/activemq/core/ActiveMQQueueBrowser.h	2998
src/main/activemq/core/ActiveMQSession.h	2998
src/main/activemq/core/ActiveMQSessionExecutor.h	2999
src/main/activemq/core/ActiveMQTransactionContext.h	3000
src/main/activemq/core/ActiveMQXAConnection.h	3000
src/main/activemq/core/ActiveMQXAConnectionFactory.h	3001
src/main/activemq/core/ActiveMQXASession.h	3001
src/main/activemq/core/DispatchData.h	3001
src/main/activemq/core/Dispatcher.h	3002
src/main/activemq/core/FifoMessageDispatchChannel.h	3002
src/main/activemq/core/MessageDispatchChannel.h	3003
src/main/activemq/core/PrefetchPolicy.h	3004
src/main/activemq/core/RedeliveryPolicy.h	3005
src/main/activemq/core/SimplePriorityMessageDispatchChannel.h	3005
src/main/activemq/core/Synchronization.h	3005
src/main/activemq/core/policies/DefaultPrefetchPolicy.h	3003
src/main/activemq/core/policies/DefaultRedeliveryPolicy.h	3004
src/main/activemq/exceptions/ActiveMQException.h	3006
src/main/activemq/exceptions/BrokerException.h	3006
src/main/activemq/exceptions/ConnectionFailedException.h	3007
src/main/activemq/exceptions/ExceptionDefines.h	3007
src/main/activemq/io/LoggingInputStream.h	3012
src/main/activemq/io/LoggingOutputStream.h	3012
src/main/activemq/library/ActiveMQCPP.h	3012
src/main/activemq/state/CommandVisitor.h	3013
src/main/activemq/state/CommandVisitorAdapter.h	3013
src/main/activemq/state/ConnectionState.h	3014
src/main/activemq/state/ConnectionStateTracker.h	3015
src/main/activemq/state/ConsumerState.h	3015
src/main/activemq/state/ProducerState.h	3016
src/main/activemq/state/SessionState.h	3016
src/main/activemq/state/Tracked.h	3017
src/main/activemq/state/TransactionState.h	3017
src/main/activemq/threads/CompositeTask.h	3018
src/main/activemq/threads/CompositeTaskRunner.h	3018
src/main/activemq/threads/DedicatedTaskRunner.h	3019
src/main/activemq/threads/Scheduler.h	3019
src/main/activemq/threads/SchedulerTimerTask.h	3020
src/main/activemq/threads/Task.h	3020
src/main/activemq/threads/TaskRunner.h	3020
src/main/activemq/transport/AbstractTransportFactory.h	3021
src/main/activemq/transport/CompositeTransport.h	3021

src/main/activemq/transport/ DefaultTransportListener.h	3023
src/main/activemq/transport/ IOTransport.h	3029
src/main/activemq/transport/ Transport.h	3034
src/main/activemq/transport/ TransportFactory.h	3035
src/main/activemq/transport/ TransportFilter.h	3035
src/main/activemq/transport/ TransportListener.h	3036
src/main/activemq/transport/ TransportRegistry.h	3036
src/main/activemq/transport/correlator/ FutureResponse.h	3022
src/main/activemq/transport/correlator/ ResponseCorrelator.h	3022
src/main/activemq/transport/failover/ BackupTransport.h	3024
src/main/activemq/transport/failover/ BackupTransportPool.h	3024
src/main/activemq/transport/failover/ CloseTransportsTask.h	3025
src/main/activemq/transport/failover/ FailoverTransport.h	3025
src/main/activemq/transport/failover/ FailoverTransportFactory.h	3026
src/main/activemq/transport/failover/ FailoverTransportListener.h	3026
src/main/activemq/transport/failover/ URIPool.h	3027
src/main/activemq/transport/inactivity/ InactivityMonitor.h	3027
src/main/activemq/transport/inactivity/ ReadChecker.h	3028
src/main/activemq/transport/inactivity/ WriteChecker.h	3028
src/main/activemq/transport/logging/ LoggingTransport.h	3029
src/main/activemq/transport/mock/ InternalCommandListener.h	3030
src/main/activemq/transport/mock/ MockTransport.h	3030
src/main/activemq/transport/mock/ MockTransportFactory.h	3031
src/main/activemq/transport/mock/ ResponseBuilder.h	3032
src/main/activemq/transport/tcp/ SslTransport.h	3032
src/main/activemq/transport/tcp/ SslTransportFactory.h	3033
src/main/activemq/transport/tcp/ TcpTransport.h	3033
src/main/activemq/transport/tcp/ TcpTransportFactory.h	3034
src/main/activemq/util/ ActiveMQProperties.h	3037
src/main/activemq/util/ CMSExceptionSupport.h	3037
src/main/activemq/util/ CompositeData.h	3039
src/main/activemq/util/ Config.h	3039
src/main/activemq/util/ IdGenerator.h	3040
src/main/activemq/util/ LongSequenceGenerator.h	3041
src/main/activemq/util/ MarshallingSupport.h	3041
src/main/activemq/util/ MemoryUsage.h	3041
src/main/activemq/util/ PrimitiveList.h	3042
src/main/activemq/util/ PrimitiveMap.h	3042
src/main/activemq/util/ PrimitiveValueConverter.h	3043
src/main/activemq/util/ PrimitiveValueNode.h	3043
src/main/activemq/util/ Service.h	3044
src/main/activemq/util/ ServiceListener.h	3044
src/main/activemq/util/ ServiceStopper.h	3045
src/main/activemq/util/ ServiceSupport.h	3045
src/main/activemq/util/ URISupport.h	3046
src/main/activemq/util/ Usage.h	3046
src/main/activemq/wireformat/ MarshalAware.h	3046
src/main/activemq/wireformat/ WireFormat.h	3094
src/main/activemq/wireformat/ WireFormatFactory.h	3094
src/main/activemq/wireformat/ WireFormatNegotiator.h	3095

src/main/activemq/wireformat/ WireFormatRegistry.h	3095
src/main/activemq/wireformat/openwire/ OpenWireFormat.h	3088
src/main/activemq/wireformat/openwire/ OpenWireFormatFactory.h	3088
src/main/activemq/wireformat/openwire/ OpenWireFormatNegotiator.h	3089
src/main/activemq/wireformat/openwire/ OpenWireResponseBuilder.h	3089
src/main/activemq/wireformat/openwire/marshal/ BaseDataStreamMarshaller.h	3047
src/main/activemq/wireformat/openwire/marshal/ DataStreamMarshaller.h	3047
src/main/activemq/wireformat/openwire/marshal/ PrimitiveTypesMarshaller.h	3087
src/main/activemq/wireformat/openwire/marshal/generated/ ActiveMQBlobMessageMarshaller.h	3048
src/main/activemq/wireformat/openwire/marshal/generated/ ActiveMQBytesMessageMarshaller.h	3049
src/main/activemq/wireformat/openwire/marshal/generated/ ActiveMQDestinationMarshaller.h	3049
src/main/activemq/wireformat/openwire/marshal/generated/ ActiveMQMapMessageMarshaller.h	3050
src/main/activemq/wireformat/openwire/marshal/generated/ ActiveMQObjectMessageMarshaller.h	3051
src/main/activemq/wireformat/openwire/marshal/generated/ ActiveMQQueueMarshaller.h	3052
src/main/activemq/wireformat/openwire/marshal/generated/ ActiveMQStreamMessageMarshaller.h	3052
src/main/activemq/wireformat/openwire/marshal/generated/ ActiveMQTempDestinationMarshaller.h	3053
src/main/activemq/wireformat/openwire/marshal/generated/ ActiveMQTempQueueMarshaller.h	3054
src/main/activemq/wireformat/openwire/marshal/generated/ ActiveMQTempTopicMarshaller.h	3054
src/main/activemq/wireformat/openwire/marshal/generated/ ActiveMQTextMessageMarshaller.h	3055
src/main/activemq/wireformat/openwire/marshal/generated/ ActiveMQTopicMarshaller.h	3056
src/main/activemq/wireformat/openwire/marshal/generated/ BaseCommandMarshaller.h	3056
src/main/activemq/wireformat/openwire/marshal/generated/ BrokerIdMarshaller.h	3057
src/main/activemq/wireformat/openwire/marshal/generated/ BrokerInfoMarshaller.h	3058
src/main/activemq/wireformat/openwire/marshal/generated/ ConnectionControlMarshaller.h	3058
src/main/activemq/wireformat/openwire/marshal/generated/ ConnectionErrorMarshaller.h	3059
src/main/activemq/wireformat/openwire/marshal/generated/ ConnectionIdMarshaller.h	3060
src/main/activemq/wireformat/openwire/marshal/generated/ ConnectionInfoMarshaller.h	3060

src/main/activemq/wireformat/openwire/marshall/generated/ Consumer- ControlMarshaller.h	3061
src/main/activemq/wireformat/openwire/marshall/generated/ ConsumerId- Marshaller.h	3061
src/main/activemq/wireformat/openwire/marshall/generated/ ConsumerInfo- Marshaller.h	3062
src/main/activemq/wireformat/openwire/marshall/generated/ ControlCommand- Marshaller.h	3063
src/main/activemq/wireformat/openwire/marshall/generated/ DataArray- ResponseMarshaller.h	3063
src/main/activemq/wireformat/openwire/marshall/generated/ DataResponse- Marshaller.h	3064
src/main/activemq/wireformat/openwire/marshall/generated/ DestinationInfo- Marshaller.h	3065
src/main/activemq/wireformat/openwire/marshall/generated/ DiscoveryEvent- Marshaller.h	3065
src/main/activemq/wireformat/openwire/marshall/generated/ Exception- ResponseMarshaller.h	3066
src/main/activemq/wireformat/openwire/marshall/generated/ FlushCommand- Marshaller.h	3066
src/main/activemq/wireformat/openwire/marshall/generated/ IntegerResponse- Marshaller.h	3067
src/main/activemq/wireformat/openwire/marshall/generated/ JournalQueue- AckMarshaller.h	3068
src/main/activemq/wireformat/openwire/marshall/generated/ JournalTopic- AckMarshaller.h	3068
src/main/activemq/wireformat/openwire/marshall/generated/ JournalTrace- Marshaller.h	3069
src/main/activemq/wireformat/openwire/marshall/generated/ JournalTransaction- Marshaller.h	3070
src/main/activemq/wireformat/openwire/marshall/generated/ KeepAliveInfo- Marshaller.h	3070
src/main/activemq/wireformat/openwire/marshall/generated/ LastPartial- CommandMarshaller.h	3071
src/main/activemq/wireformat/openwire/marshall/generated/ LocalTransaction- IdMarshaller.h	3071
src/main/activemq/wireformat/openwire/marshall/generated/ Marshaller- Factory.h	3072
src/main/activemq/wireformat/openwire/marshall/generated/ MessageAck- Marshaller.h	3073
src/main/activemq/wireformat/openwire/marshall/generated/ MessageDispatch- Marshaller.h	3073
src/main/activemq/wireformat/openwire/marshall/generated/ MessageDispatch- NotificationMarshaller.h	3074
src/main/activemq/wireformat/openwire/marshall/generated/ MessageId- Marshaller.h	3074
src/main/activemq/wireformat/openwire/marshall/generated/ MessageMarshaller- h	3075
src/main/activemq/wireformat/openwire/marshall/generated/ MessagePull- Marshaller.h	3076

src/main/activemq/wireformat/openwire/marshal/generated/ NetworkBridge-FilterMarshaller.h	3076
src/main/activemq/wireformat/openwire/marshal/generated/ PartialCommand-Marshaller.h	3077
src/main/activemq/wireformat/openwire/marshal/generated/ ProducerAck-Marshaller.h	3078
src/main/activemq/wireformat/openwire/marshal/generated/ ProducerId-Marshaller.h	3078
src/main/activemq/wireformat/openwire/marshal/generated/ ProducerInfo-Marshaller.h	3079
src/main/activemq/wireformat/openwire/marshal/generated/ RemoveInfo-Marshaller.h	3079
src/main/activemq/wireformat/openwire/marshal/generated/ RemoveSubscription-InfoMarshaller.h	3080
src/main/activemq/wireformat/openwire/marshal/generated/ ReplayCommand-Marshaller.h	3081
src/main/activemq/wireformat/openwire/marshal/generated/ Response-Marshaller.h	3081
src/main/activemq/wireformat/openwire/marshal/generated/ SessionId-Marshaller.h	3082
src/main/activemq/wireformat/openwire/marshal/generated/ SessionInfo-Marshaller.h	3083
src/main/activemq/wireformat/openwire/marshal/generated/ ShutdownInfo-Marshaller.h	3083
src/main/activemq/wireformat/openwire/marshal/generated/ Subscription-InfoMarshaller.h	3084
src/main/activemq/wireformat/openwire/marshal/generated/ TransactionId-Marshaller.h	3084
src/main/activemq/wireformat/openwire/marshal/generated/ TransactionInfo-Marshaller.h	3085
src/main/activemq/wireformat/openwire/marshal/generated/ WireFormatInfo-Marshaller.h	3086
src/main/activemq/wireformat/openwire/marshal/generated/ XATransaction-IdMarshaller.h	3086
src/main/activemq/wireformat/openwire/utis/ BooleanStream.h	3090
src/main/activemq/wireformat/openwire/utis/ HexTable.h	3090
src/main/activemq/wireformat/openwire/utis/ MessagePropertyInterceptor.h	3091
src/main/activemq/wireformat/stomp/ StompCommandConstants.h	3091
src/main/activemq/wireformat/stomp/ StompFrame.h	3092
src/main/activemq/wireformat/stomp/ StompHelper.h	3092
src/main/activemq/wireformat/stomp/ StompWireFormat.h	3093
src/main/activemq/wireformat/stomp/ StompWireFormatFactory.h	3093
src/main/cms/ BytesMessage.h	3096
src/main/cms/ Closeable.h	3096
src/main/cms/ CMSException.h	3097
src/main/cms/ CMSProperties.h	3098
src/main/cms/ CMSSecurityException.h	3098
src/main/cms/ Config.h	3039
src/main/cms/ Connection.h	3098
src/main/cms/ ConnectionFactory.h	3099

src/main/cms/ ConnectionMetaData.h	3099
src/main/cms/ DeliveryMode.h	3100
src/main/cms/ Destination.h	3100
src/main/cms/ ExceptionListener.h	3100
src/main/cms/ IllegalStateException.h	3101
src/main/cms/ InvalidClientIdException.h	3102
src/main/cms/ InvalidDestinationException.h	3102
src/main/cms/ InvalidSelectorException.h	3103
src/main/cms/ MapMessage.h	3103
src/main/cms/ Message.h	2984
src/main/cms/ MessageConsumer.h	3103
src/main/cms/ MessageEnumeration.h	3104
src/main/cms/ MessageEOFException.h	3104
src/main/cms/ MessageFormatException.h	3105
src/main/cms/ MessageListener.h	3105
src/main/cms/ MessageNotReadableException.h	3105
src/main/cms/ MessageNotWriteableException.h	3106
src/main/cms/ MessageProducer.h	3106
src/main/cms/ ObjectMessage.h	3107
src/main/cms/ Queue.h	3107
src/main/cms/ QueueBrowser.h	3108
src/main/cms/ Session.h	3109
src/main/cms/ Startable.h	3109
src/main/cms/ Stoppable.h	3109
src/main/cms/ StreamMessage.h	3110
src/main/cms/ TemporaryQueue.h	3110
src/main/cms/ TemporaryTopic.h	3111
src/main/cms/ TextMessage.h	3111
src/main/cms/ Topic.h	3112
src/main/cms/ TransactionInProgressException.h	3112
src/main/cms/ TransactionRolledBackException.h	3112
src/main/cms/ UnsupportedOperationException.h	3113
src/main/cms/ XAConnection.h	3114
src/main/cms/ XAConnectionFactory.h	3114
src/main/cms/ XAException.h	3115
src/main/cms/ XAResource.h	3115
src/main/cms/ XASession.h	3115
src/main/cms/ Xid.h	3116
src/main/decaf/internal/ AprPool.h	3116
src/main/decaf/internal/ DecafRuntime.h	3117
src/main/decaf/internal/io/ StandardErrorOutputStream.h	3117
src/main/decaf/internal/io/ StandardInputStream.h	3118
src/main/decaf/internal/io/ StandardOutputStream.h	3118
src/main/decaf/internal/net/ DefaultServerSocketFactory.h	3119
src/main/decaf/internal/net/ DefaultSocketFactory.h	3119
src/main/decaf/internal/net/ Network.h	3120
src/main/decaf/internal/net/ SocketFileDescriptor.h	3120
src/main/decaf/internal/net/ URIEncoderDecoder.h	3128
src/main/decaf/internal/net/ URIHelper.h	3129
src/main/decaf/internal/net/ URIType.h	3129

src/main/decaf/internal/net/ssl/ DefaultSSLContext.h	3121
src/main/decaf/internal/net/ssl/ DefaultSSLServerSocketFactory.h	3121
src/main/decaf/internal/net/ssl/ DefaultSSLSocketFactory.h	3122
src/main/decaf/internal/net/ssl/openssl/ OpenSSLContextSpi.h	3122
src/main/decaf/internal/net/ssl/openssl/ OpenSSLParameters.h	3123
src/main/decaf/internal/net/ssl/openssl/ OpenSSLServerSocket.h	3123
src/main/decaf/internal/net/ssl/openssl/ OpenSSLServerSocketFactory.h	3124
src/main/decaf/internal/net/ssl/openssl/ OpenSSLSocket.h	3124
src/main/decaf/internal/net/ssl/openssl/ OpenSSLSocketException.h	3125
src/main/decaf/internal/net/ssl/openssl/ OpenSSLSocketFactory.h	3125
src/main/decaf/internal/net/ssl/openssl/ OpenSSLSocketInputStream.h	3126
src/main/decaf/internal/net/ssl/openssl/ OpenSSLSocketOutputStream.h	3126
src/main/decaf/internal/net/tcp/ TcpSocket.h	3127
src/main/decaf/internal/net/tcp/ TcpSocketInputStream.h	3127
src/main/decaf/internal/net/tcp/ TcpSocketOutputStream.h	3128
src/main/decaf/internal/nio/ BufferFactory.h	3129
src/main/decaf/internal/nio/ ByteBuffer.h	3130
src/main/decaf/internal/nio/ CharArrayBuffer.h	3131
src/main/decaf/internal/nio/ DoubleArrayBuffer.h	3131
src/main/decaf/internal/nio/ FloatArrayBuffer.h	3132
src/main/decaf/internal/nio/ IntArrayBuffer.h	3132
src/main/decaf/internal/nio/ LongArrayBuffer.h	3133
src/main/decaf/internal/nio/ ShortArrayBuffer.h	3133
src/main/decaf/internal/security/unix/ SecureRandomImpl.h	3134
src/main/decaf/internal/security/windows/ SecureRandomImpl.h	3134
src/main/decaf/internal/util/ ByteArrayAdapter.h	3135
src/main/decaf/internal/util/ GenericResource.h	3140
src/main/decaf/internal/util/ HexStringParser.h	3141
src/main/decaf/internal/util/ Resource.h	3141
src/main/decaf/internal/util/ ResourceLifecycleManager.h	2962
src/main/decaf/internal/util/ TimerTaskHeap.h	3141
src/main/decaf/internal/util/concurrent/ ConditionImpl.h	3135
src/main/decaf/internal/util/concurrent/ MutexImpl.h	3136
src/main/decaf/internal/util/concurrent/ SynchronizableImpl.h	3136
src/main/decaf/internal/util/concurrent/ Transferer.h	3137
src/main/decaf/internal/util/concurrent/ TransferQueue.h	3137
src/main/decaf/internal/util/concurrent/ TransferStack.h	3138
src/main/decaf/internal/util/concurrent/unix/ ConditionHandle.h	3138
src/main/decaf/internal/util/concurrent/unix/ MutexHandle.h	3139
src/main/decaf/internal/util/concurrent/windows/ ConditionHandle.h	3139
src/main/decaf/internal/util/concurrent/windows/ MutexHandle.h	3140
src/main/decaf/internal/util/zip/ crc32.h	3142
src/main/decaf/internal/util/zip/ deflate.h	3142
src/main/decaf/internal/util/zip/ gzguts.h	3146
src/main/decaf/internal/util/zip/ inffast.h	3148
src/main/decaf/internal/util/zip/ inffixed.h	3148
src/main/decaf/internal/util/zip/ inflate.h	3148
src/main/decaf/internal/util/zip/ infrees.h	3149
src/main/decaf/internal/util/zip/ trees.h	3150
src/main/decaf/internal/util/zip/ zconf.h	3152

src/main/decaf/internal/util/zip/zlib.h	3153
src/main/decaf/internal/util/zip/zutil.h	3161
src/main/decaf/io/BlockingByteArrayInputStream.h	3164
src/main/decaf/io/BufferedInputStream.h	3164
src/main/decaf/io/BufferedOutputStream.h	3164
src/main/decaf/io/ByteArrayInputStream.h	3165
src/main/decaf/io/ByteArrayOutputStream.h	3165
src/main/decaf/io/Closeable.h	3097
src/main/decaf/io/DataInput.h	3166
src/main/decaf/io/DataInputStream.h	3166
src/main/decaf/io/DataOutput.h	3167
src/main/decaf/io/DataOutputStream.h	3167
src/main/decaf/io/EOFException.h	3168
src/main/decaf/io/FileDescriptor.h	3168
src/main/decaf/io/FilterInputStream.h	3169
src/main/decaf/io/FilterOutputStream.h	3169
src/main/decaf/io/Flushable.h	3170
src/main/decaf/io/InputStream.h	3170
src/main/decaf/io/InputStreamReader.h	3171
src/main/decaf/io/InterruptedIOException.h	3171
src/main/decaf/io/IOException.h	3171
src/main/decaf/io/OutputStream.h	3172
src/main/decaf/io/OutputStreamWriter.h	3172
src/main/decaf/io/PushbackInputStream.h	3173
src/main/decaf/io/Reader.h	3173
src/main/decaf/io/UnsupportedEncodingException.h	3174
src/main/decaf/io/UTFDataFormatException.h	3174
src/main/decaf/io/Writer.h	3174
src/main/decaf/lang/Appendable.h	3175
src/main/decaf/lang/ArrayPointer.h	3175
src/main/decaf/lang/Boolean.h	3176
src/main/decaf/lang/Byte.h	3177
src/main/decaf/lang/Character.h	3177
src/main/decaf/lang/CharSequence.h	3177
src/main/decaf/lang/Comparable.h	3178
src/main/decaf/lang/Double.h	3178
src/main/decaf/lang/Exception.h	3179
src/main/decaf/lang/Float.h	3183
src/main/decaf/lang/Integer.h	3184
src/main/decaf/lang/Iterable.h	3184
src/main/decaf/lang/Long.h	3185
src/main/decaf/lang/Math.h	3185
src/main/decaf/lang/Number.h	3186
src/main/decaf/lang/Pointer.h	3186
src/main/decaf/lang/Readable.h	3187
src/main/decaf/lang/Runnable.h	3187
src/main/decaf/lang/Runtime.h	3188
src/main/decaf/lang/Short.h	3188
src/main/decaf/lang/String.h	3189
src/main/decaf/lang/System.h	3189

src/main/decaf/lang/ Thread.h	3190
src/main/decaf/lang/ ThreadGroup.h	3190
src/main/decaf/lang/ Throwable.h	3191
src/main/decaf/lang/exceptions/ ClassCastException.h	3179
src/main/decaf/lang/exceptions/ ExceptionDefines.h	3009
src/main/decaf/lang/exceptions/ IllegalArgumentException.h	3180
src/main/decaf/lang/exceptions/ IllegalMonitorStateException.h	3180
src/main/decaf/lang/exceptions/ IllegalStateException.h	3101
src/main/decaf/lang/exceptions/ IllegalThreadStateException.h	3180
src/main/decaf/lang/exceptions/ IndexOutOfBoundsException.h	3181
src/main/decaf/lang/exceptions/ InterruptedException.h	3181
src/main/decaf/lang/exceptions/ InvalidStateException.h	3182
src/main/decaf/lang/exceptions/ NullPointerException.h	3182
src/main/decaf/lang/exceptions/ NumberFormatException.h	3183
src/main/decaf/lang/exceptions/ RuntimeException.h	3183
src/main/decaf/lang/exceptions/ UnsupportedOperationException.h	3113
src/main/decaf/net/ BindException.h	3191
src/main/decaf/net/ ConnectException.h	3192
src/main/decaf/net/ DatagramPacket.h	3192
src/main/decaf/net/ HttpRetryException.h	3192
src/main/decaf/net/ Inet4Address.h	3193
src/main/decaf/net/ Inet6Address.h	3193
src/main/decaf/net/ InetAddress.h	3194
src/main/decaf/net/ InetSocketAddress.h	3194
src/main/decaf/net/ MalformedURLException.h	3194
src/main/decaf/net/ NoRouteToHostException.h	3195
src/main/decaf/net/ PortUnreachableException.h	3195
src/main/decaf/net/ ProtocolException.h	3196
src/main/decaf/net/ ServerSocket.h	3196
src/main/decaf/net/ ServerSocketFactory.h	3196
src/main/decaf/net/ Socket.h	3197
src/main/decaf/net/ SocketAddress.h	3197
src/main/decaf/net/ SocketError.h	3198
src/main/decaf/net/ SocketException.h	3198
src/main/decaf/net/ SocketFactory.h	3199
src/main/decaf/net/ SocketImpl.h	3199
src/main/decaf/net/ SocketImplFactory.h	3200
src/main/decaf/net/ SocketOptions.h	3200
src/main/decaf/net/ SocketTimeoutException.h	3200
src/main/decaf/net/ UnknownHostException.h	3204
src/main/decaf/net/ UnknownServiceException.h	3204
src/main/decaf/net/ URI.h	3205
src/main/decaf/net/ URISyntaxException.h	3205
src/main/decaf/net/ URL.h	3206
src/main/decaf/net/ URLDecoder.h	3206
src/main/decaf/net/ URLEncoder.h	3206
src/main/decaf/net/ssl/ SSLContext.h	3201
src/main/decaf/net/ssl/ SSLContextSpi.h	3201
src/main/decaf/net/ssl/ SSLParameters.h	3202
src/main/decaf/net/ssl/ SSLServerSocket.h	3202

src/main/decaf/net/ssl/SSLServerSocketFactory.h	3203
src/main/decaf/net/ssl/SSLSocket.h	3203
src/main/decaf/net/ssl/SSLSocketFactory.h	3203
src/main/decaf/nio/Buffer.h	3207
src/main/decaf/nio/BufferOverflowException.h	3207
src/main/decaf/nio/BufferUnderflowException.h	3208
src/main/decaf/nio/ByteBuffer.h	3208
src/main/decaf/nio/CharBuffer.h	3208
src/main/decaf/nio/DoubleBuffer.h	3209
src/main/decaf/nio/FloatBuffer.h	3210
src/main/decaf/nio/IntBuffer.h	3210
src/main/decaf/nio/InvalidMarkException.h	3211
src/main/decaf/nio/LongBuffer.h	3211
src/main/decaf/nio/ReadOnlyBufferException.h	3211
src/main/decaf/nio/ShortBuffer.h	3212
src/main/decaf/security/GeneralSecurityException.h	3216
src/main/decaf/security/InvalidKeyException.h	3216
src/main/decaf/security/Key.h	3217
src/main/decaf/security/KeyException.h	3217
src/main/decaf/security/KeyManagementException.h	3218
src/main/decaf/security/NoSuchAlgorithmException.h	3218
src/main/decaf/security/NoSuchProviderException.h	3218
src/main/decaf/security/Principal.h	3219
src/main/decaf/security/PublicKey.h	3219
src/main/decaf/security/SecureRandom.h	3220
src/main/decaf/security/SecureRandomSpi.h	3220
src/main/decaf/security/SignatureException.h	3221
src/main/decaf/security/auth/x500/X500Principal.h	3212
src/main/decaf/security/cert/Certificate.h	3213
src/main/decaf/security/cert/CertificateEncodingException.h	3213
src/main/decaf/security/cert/CertificateException.h	3214
src/main/decaf/security/cert/CertificateExpiredException.h	3214
src/main/decaf/security/cert/CertificateNotYetValidException.h	3215
src/main/decaf/security/cert/CertificateParsingException.h	3215
src/main/decaf/security/cert/X509Certificate.h	3216
src/main/decaf/util/AbstractCollection.h	3221
src/main/decaf/util/AbstractList.h	3222
src/main/decaf/util/AbstractMap.h	3222
src/main/decaf/util/AbstractQueue.h	3223
src/main/decaf/util/AbstractSequentialList.h	3223
src/main/decaf/util/AbstractSet.h	3224
src/main/decaf/util/ArrayList.h	3224
src/main/decaf/util/Arrays.h	3225
src/main/decaf/util/Collection.h	3225
src/main/decaf/util/Comparator.h	3226
src/main/decaf/util/ConcurrentModificationException.h	3247
src/main/decaf/util/Config.h	3040
src/main/decaf/util/Date.h	3248
src/main/decaf/util/Deque.h	3248
src/main/decaf/util/Iterator.h	3248

src/main/decaf/util/ LinkedList.h	3249
src/main/decaf/util/ List.h	3250
src/main/decaf/util/ ListIterator.h	3250
src/main/decaf/util/ Map.h	3260
src/main/decaf/util/ NoSuchElementException.h	3260
src/main/decaf/util/ PriorityQueue.h	3261
src/main/decaf/util/ Properties.h	3261
src/main/decaf/util/ Queue.h	3108
src/main/decaf/util/ Random.h	3262
src/main/decaf/util/ Set.h	3262
src/main/decaf/util/ StlList.h	3263
src/main/decaf/util/ StlMap.h	3263
src/main/decaf/util/ StlQueue.h	3264
src/main/decaf/util/ StlSet.h	3264
src/main/decaf/util/ StringTokenizer.h	3265
src/main/decaf/util/ Timer.h	3265
src/main/decaf/util/ TimerTask.h	3266
src/main/decaf/util/ UUID.h	3266
src/main/decaf/util/comparators/ Less.h	3226
src/main/decaf/util/concurrent/ AbstractExecutorService.h	3227
src/main/decaf/util/concurrent/ BlockingQueue.h	3229
src/main/decaf/util/concurrent/ BrokenBarrierException.h	3230
src/main/decaf/util/concurrent/ Callable.h	3230
src/main/decaf/util/concurrent/ CancellationException.h	3230
src/main/decaf/util/concurrent/ Concurrent.h	3231
src/main/decaf/util/concurrent/ ConcurrentMap.h	3232
src/main/decaf/util/concurrent/ ConcurrentStlMap.h	3232
src/main/decaf/util/concurrent/ CopyOnWriteArrayList.h	3233
src/main/decaf/util/concurrent/ CopyOnWriteArraySet.h	3234
src/main/decaf/util/concurrent/ CountDownLatch.h	3234
src/main/decaf/util/concurrent/ Delayed.h	3235
src/main/decaf/util/concurrent/ ExecutionException.h	3235
src/main/decaf/util/concurrent/ Executor.h	3236
src/main/decaf/util/concurrent/ Executors.h	3236
src/main/decaf/util/concurrent/ ExecutorService.h	3237
src/main/decaf/util/concurrent/ Future.h	3237
src/main/decaf/util/concurrent/ LinkedBlockingQueue.h	3238
src/main/decaf/util/concurrent/ Lock.h	3238
src/main/decaf/util/concurrent/ Mutex.h	3242
src/main/decaf/util/concurrent/ RejectedExecutionException.h	3242
src/main/decaf/util/concurrent/ RejectedExecutionHandler.h	3243
src/main/decaf/util/concurrent/ Semaphore.h	3243
src/main/decaf/util/concurrent/ Synchronizable.h	3244
src/main/decaf/util/concurrent/ SynchronousQueue.h	3244
src/main/decaf/util/concurrent/ ThreadFactory.h	3245
src/main/decaf/util/concurrent/ ThreadPoolExecutor.h	3245
src/main/decaf/util/concurrent/ TimeoutException.h	3246
src/main/decaf/util/concurrent/ TimeUnit.h	3247
src/main/decaf/util/concurrent/atomic/ AtomicBoolean.h	3227
src/main/decaf/util/concurrent/atomic/ AtomicInteger.h	3228

src/main/decaf/util/concurrent/atomic/ AtomicRefCounter.h	3228
src/main/decaf/util/concurrent/atomic/ AtomicReference.h	3229
src/main/decaf/util/concurrent/locks/ AbstractOwnableSynchronizer.h	3239
src/main/decaf/util/concurrent/locks/ Condition.h	3240
src/main/decaf/util/concurrent/locks/ Lock.h	3239
src/main/decaf/util/concurrent/locks/ LockSupport.h	3240
src/main/decaf/util/concurrent/locks/ ReadWriteLock.h	3241
src/main/decaf/util/concurrent/locks/ ReentrantLock.h	3241
src/main/decaf/util/logging/ ConsoleHandler.h	3251
src/main/decaf/util/logging/ ErrorManager.h	3251
src/main/decaf/util/logging/ Filter.h	3251
src/main/decaf/util/logging/ Formatter.h	3252
src/main/decaf/util/logging/ Handler.h	3252
src/main/decaf/util/logging/ Level.h	3253
src/main/decaf/util/logging/ Logger.h	3253
src/main/decaf/util/logging/ LoggerCommon.h	3254
src/main/decaf/util/logging/ LoggerDefines.h	3254
src/main/decaf/util/logging/ LoggerHierarchy.h	3255
src/main/decaf/util/logging/ LogManager.h	3256
src/main/decaf/util/logging/ LogRecord.h	3256
src/main/decaf/util/logging/ LogWriter.h	3257
src/main/decaf/util/logging/ MarkBlockLogger.h	3257
src/main/decaf/util/logging/ PropertiesChangeListener.h	3258
src/main/decaf/util/logging/ SimpleFormatter.h	3258
src/main/decaf/util/logging/ SimpleLogger.h	3259
src/main/decaf/util/logging/ StreamHandler.h	3259
src/main/decaf/util/logging/ XMLFormatter.h	3260
src/main/decaf/util/zip/ Adler32.h	3267
src/main/decaf/util/zip/ CheckedInputStream.h	3267
src/main/decaf/util/zip/ CheckedOutputStream.h	3268
src/main/decaf/util/zip/ Checksum.h	3268
src/main/decaf/util/zip/ CRC32.h	3269
src/main/decaf/util/zip/ DataFormatException.h	3269
src/main/decaf/util/zip/ Deflater.h	3270
src/main/decaf/util/zip/ DeflaterOutputStream.h	3270
src/main/decaf/util/zip/ Inflater.h	3271
src/main/decaf/util/zip/ InflaterInputStream.h	3271
src/main/decaf/util/zip/ ZipException.h	3272

Chapter 5

Namespace Documentation

5.1 activemq Namespace Reference

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

Namespaces

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

5.1.1 Detailed Description

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

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

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

5.2 activemq::cmsutil Namespace Reference

Data Structures

- class **CachedConsumer**
A cached message consumer contained within a pooled session.
- class **CachedProducer**
A cached message producer contained within a pooled session.
- class **CmsAccessor**
*Base class for **activemq.cmsutil.CmsTemplate** (p. 836) and other CMS-accessing gateway helpers, defining common properties such as the CMS **cms.ConnectionFactory** (p. 955) to operate on.*
- class **CmsDestinationAccessor**
*Extends the **CmsAccessor** (p. 819) to add support for resolving destination names.*
- class **CmsTemplate**
***CmsTemplate** (p. 836) simplifies performing synchronous CMS operations.*
- class **DestinationResolver**
*Resolves a CMS destination name to a *Destination*.*
- class **DynamicDestinationResolver**
*Resolves a CMS destination name to a *Destination*.*
- class **MessageCreator**
*Creates the user-defined message to be sent by the **CmsTemplate** (p. 836).*
- class **PooledSession**
A pooled session object that wraps around a delegate session.
- class **ProducerCallback**
Callback for sending a message to a CMS destination.
- class **ResourceLifecycleManager**
Manages the lifecycle of a set of CMS resources.
- class **SessionCallback**
Callback for executing any number of operations on a provided CMS Session.
- class **SessionPool**
A pool of CMS sessions from the same connection and with the same acknowledge mode.

5.3 activemq::commands Namespace Reference

Data Structures

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

This class represents an Exception sent from the Broker.

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

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

5.4 activemq::core Namespace Reference

Namespaces

- namespace **policies**

Data Structures

- class **ActiveMQAckHandler**
Interface class that is used to give CMS Messages an interface to Ack themselves with.
- class **ActiveMQConnection**
Concrete connection used for all connectors to the ActiveMQ broker.
- class **ActiveMQConnectionFactory**
- class **ActiveMQConnectionMetaData**
*This class houses all the various settings and information that is used by an instance of an **ActiveMQConnection** (p. 187) class.*

- class **ActiveMQConstants**
Class holding constant values for various ActiveMQ specific things Each constant is defined as an enumeration and has functions that convert back and forth between string and enum values.
- class **ActiveMQConsumer**
- class **ActiveMQProducer**
- class **ActiveMQQueueBrowser**
- class **ActiveMQSession**
- class **ActiveMQSessionExecutor**
Delegate dispatcher for a single session.
- class **ActiveMQTransactionContext**
Transaction Management class, hold messages that are to be redelivered upon a request to roll-back.
- class **ActiveMQXAConnection**
- class **ActiveMQXAConnectionFactory**
- class **ActiveMQXASession**
- class **DispatchData**
Simple POCO that contains the information necessary to route a message to a specified consumer.
- class **Dispatcher**
Interface for an object responsible for dispatching messages to consumers.
- class **FifoMessageDispatchChannel**
- class **MessageDispatchChannel**
- class **PrefetchPolicy**
Interface for a Policy object that controls message Prefetching on various destination types in ActiveMQ-CPP.
- class **RedeliveryPolicy**
*Interface for a **RedeliveryPolicy** (p.2250) object that controls how message - Redelivery is handled in ActiveMQ-CPP when a transaction is rolled back.*
- class **SimplePriorityMessageDispatchChannel**
- class **Synchronization**
*Transacted Object **Synchronization** (p.2654), used to sync the events of a - Transaction with the items in the Transaction.*

5.5 activemq::core::policies Namespace Reference

Data Structures

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

5.6 activemq::exceptions Namespace Reference

Data Structures

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

5.7 activemq::io Namespace Reference

Data Structures

- class **LoggingInputStream**
- class **LoggingOutputStream**
OutputStream filter that just logs the data being written.

5.8 activemq::library Namespace Reference

Data Structures

- class **ActiveMQCPP**

5.9 activemq::state Namespace Reference

Data Structures

- class **CommandVisitor**
Interface for an Object that can visit the various Command Objects that are sent from and to this client.
- class **CommandVisitorAdapter**
*Default Implementation of a **CommandVisitor** (p. 872) that returns NULL for all calls.*
- class **ConnectionState**
- class **ConnectionStateTracker**
- class **ConsumerState**
- class **ProducerState**
- class **SessionState**
- class **Tracked**
- class **TransactionState**

5.10 activemq::threads Namespace Reference

Data Structures

- class **CompositeTask**
*Represents a single task that can be part of a set of Tasks that are contained in a **CompositeTaskRunner** (p. 893).*
- class **CompositeTaskRunner**
*A **Task** (p. 2676) Runner that can contain one or more CompositeTasks that are each checked for pending work and run if any is present in the order that the tasks were added.*
- class **DedicatedTaskRunner**
- class **Scheduler**
***Scheduler** (p. 2317) class for use in executing Runnable Tasks either periodically or one time only with optional delay.*
- class **SchedulerTimerTask**
Extension of the Decaf TimerTask that adds a Runnable instance which is the target of this task.
- class **Task**
Represents a unit of work that requires one or more iterations to complete.
- class **TaskRunner**

5.11 activemq::transport Namespace Reference

Namespaces

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

Data Structures

- class **AbstractTransportFactory**
*Abstract implementation of the **TransportFactory** (p. 2798) interface, providing the base functionality that's common to most of the **TransportFactory** (p. 2798) instances.*
- class **CompositeTransport**
*A Composite **Transport** (p. 2790) is a **Transport** (p. 2790) implementation that is composed of several Transports.*
- class **DefaultTransportListener**
*A Utility class that create empty implementations for the **TransportListener** (p. 2810) interface so that a subclass only needs to override the one's its interested.*

- class **IOTransport**
*Implementation of the **Transport** (p. 2790) interface that performs marshaling of commands to IO streams.*
- class **Transport**
Interface for a transport layer for command objects.
- class **TransportFactory**
Defines the interface for Factories that create Transports or TransportFilters.
- class **TransportFilter**
A filter on the transport layer.
- class **TransportListener**
A listener of asynchronous exceptions from a command transport object.
- class **TransportRegistry**
*Registry of all **Transport** (p. 2790) Factories that are available to the client at runtime.*

5.12 activemq::transport::correlator Namespace Reference

Data Structures

- class **FutureResponse**
A container that holds a response object.
- class **ResponseCorrelator**
This type of transport filter is responsible for correlating asynchronous responses with requests.

5.13 activemq::transport::failover Namespace Reference

Data Structures

- class **BackupTransport**
- class **BackupTransportPool**
- class **CloseTransportsTask**
- class **FailoverTransport**
- class **FailoverTransportFactory**
*Creates an instance of a **FailoverTransport** (p. 1305).*
- class **FailoverTransportListener**
*Utility class used by the **Transport** (p. 2790) to perform the work of responding to events from the active **Transport** (p. 2790).*
- class **URIPool**

5.14 activemq::transport::inactivity Namespace Reference

Data Structures

- class **InactivityMonitor**
- class **ReadChecker**
Runnable class that is used by the {}.
- class **WriteChecker**
Runnable class used by the {}.

5.15 activemq::transport::logging Namespace Reference

Data Structures

- class **LoggingTransport**
A transport filter that logs commands as they are sent/received.

5.16 activemq::transport::mock Namespace Reference

Data Structures

- class **InternalCommandListener**
*Listens for Commands sent from the **MockTransport** (p. 1948).*
- class **MockTransport**
*The **MockTransport** (p. 1948) defines a base level **Transport** (p. 2790) class that is intended to be used in place of an a regular protocol **Transport** (p. 2790) such as TCP.*
- class **MockTransportFactory**
Manufactures MockTransports, which are objects that read from input streams and write to output streams.
- class **ResponseBuilder**
Interface for all Protocols to implement that defines the behavior of the Broker in response to messages of that protocol.

5.17 activemq::transport::tcp Namespace Reference

Data Structures

- class **SslTransport**
***Transport** (p. 2790) for connecting to a Broker using an SSL Socket.*
- class **SslTransportFactory**
- class **TcpTransport**

*Implements a TCP/IP based transport filter, this transport is meant to wrap an instance of an **IOTransport** (p. 1548).*

- class **TcpTransportFactory**

*Factory Responsible for creating the **TcpTransport** (p. 2693).*

5.18 activemq::util Namespace Reference

Data Structures

- class **ActiveMQProperties**

*Implementation of the CMSProperties interface that delegates to a **decaf::util::Properties** (p. 2200) object.*

- class **CMSExceptionSupport**

- class **CompositeData**

Represents a Composite URI.

- class **IdGenerator**

- class **LongSequenceGenerator**

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

- class **MarshallingSupport**

- class **MemoryUsage**

- class **PrimitiveList**

List of primitives.

- class **PrimitiveMap**

Map of named primitives.

- class **PrimitiveValueConverter**

*Class controls the conversion of data contained in a **PrimitiveValueNode** (p. 2145) from one type to another.*

- class **PrimitiveValueNode**

Class that wraps around a single value of one of the many types.

- class **Service**

*Base interface for all classes that run as a **Service** (p. 2355) inside the application.*

- class **ServiceListener**

*Listener interface for observers of **Service** (p. 2355) related events.*

- class **ServiceStopper**

- class **ServiceSupport**

*Provides a base class for **Service** (p. 2355) implementations.*

- class **URISupport**

- class **Usage**

5.19 activemq::wireformat Namespace Reference

Namespaces

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

Data Structures

- class **MarshalAware**
- class **WireFormat**
Provides a mechanism to marshal commands into and out of packets or into and out of streams, Channels and Datagrams.
- class **WireFormatFactory**
*The **WireFormatFactory** (p.2888) is the interface that all **WireFormatFactory** (p.2888) classes must extend.*
- class **WireFormatNegotiator**
*Defines a **WireFormatNegotiator** (p.2904) which allows a **WireFormat** (p.2884) to.*
- class **WireFormatRegistry**
*Registry of all **WireFormat** (p.2884) Factories that are available to the client at run-time.*

5.20 activemq::wireformat::openwire Namespace Reference

Namespaces

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

Data Structures

- class **OpenWireFormat**
- class **OpenWireFormatFactory**
- class **OpenWireFormatNegotiator**
- class **OpenWireResponseBuilder**
Used to allow a MockTransport to generate response commands to OpenWire - Commands.

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

Namespaces

- namespace **generated**

Data Structures

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

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

Data Structures

- class **ActiveMQBlobMessageMarshaller**
*Marshaling code for Open Wire Format for **ActiveMQBlobMessageMarshaller** (p. 161).*
- class **ActiveMQBytesMessageMarshaller**
*Marshaling code for Open Wire Format for **ActiveMQBytesMessageMarshaller** (p. 183).*
- class **ActiveMQDestinationMarshaller**
*Marshaling code for Open Wire Format for **ActiveMQDestinationMarshaller** (p. 258).*
- class **ActiveMQMapMessageMarshaller**
*Marshaling code for Open Wire Format for **ActiveMQMapMessageMarshaller** (p. 284).*
- class **ActiveMQMessageMarshaller**
*Marshaling code for Open Wire Format for **ActiveMQMessageMarshaller** (p. 291).*
- class **ActiveMQObjectMessageMarshaller**
*Marshaling code for Open Wire Format for **ActiveMQObjectMessageMarshaller** (p. 304).*
- class **ActiveMQQueueMarshaller**
*Marshaling code for Open Wire Format for **ActiveMQQueueMarshaller** (p. 329).*
- class **ActiveMQStreamMessageMarshaller**
*Marshaling code for Open Wire Format for **ActiveMQStreamMessageMarshaller** (p. 375).*
- class **ActiveMQTempDestinationMarshaller**
*Marshaling code for Open Wire Format for **ActiveMQTempDestinationMarshaller** (p. 382).*

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

- class **ActiveMQTempQueueMarshaller**
*Marshaling code for Open Wire Format for **ActiveMQTempQueueMarshaller** (p. 391).*
- class **ActiveMQTempTopicMarshaller**
*Marshaling code for Open Wire Format for **ActiveMQTempTopicMarshaller** (p. 401).*
- class **ActiveMQTextMessageMarshaller**
*Marshaling code for Open Wire Format for **ActiveMQTextMessageMarshaller** (p. 410).*
- class **ActiveMQTopicMarshaller**
*Marshaling code for Open Wire Format for **ActiveMQTopicMarshaller** (p. 419).*
- class **BaseCommandMarshaller**
*Marshaling code for Open Wire Format for **BaseCommandMarshaller** (p. 499).*
- class **BrokerIdMarshaller**
*Marshaling code for Open Wire Format for **BrokerIdMarshaller** (p. 567).*
- class **BrokerInfoMarshaller**
*Marshaling code for Open Wire Format for **BrokerInfoMarshaller** (p. 578).*
- class **ConnectionControlMarshaller**
*Marshaling code for Open Wire Format for **ConnectionControlMarshaller** (p. 943).*
- class **ConnectionErrorMarshaller**
*Marshaling code for Open Wire Format for **ConnectionErrorMarshaller** (p. 951).*
- class **ConnectionIdMarshaller**
*Marshaling code for Open Wire Format for **ConnectionIdMarshaller** (p. 963).*
- class **ConnectionInfoMarshaller**
*Marshaling code for Open Wire Format for **ConnectionInfoMarshaller** (p. 974).*
- class **ConsumerControlMarshaller**
*Marshaling code for Open Wire Format for **ConsumerControlMarshaller** (p. 996).*
- class **ConsumerIdMarshaller**
*Marshaling code for Open Wire Format for **ConsumerIdMarshaller** (p. 1005).*
- class **ConsumerInfoMarshaller**
*Marshaling code for Open Wire Format for **ConsumerInfoMarshaller** (p. 1017).*
- class **ControlCommandMarshaller**
*Marshaling code for Open Wire Format for **ControlCommandMarshaller** (p. 1025).*
- class **DataArrayResponseMarshaller**
*Marshaling code for Open Wire Format for **DataArrayResponseMarshaller** (p. 1072).*
- class **DataResponseMarshaller**
*Marshaling code for Open Wire Format for **DataResponseMarshaller** (p. 1115).*
- class **DestinationInfoMarshaller**
*Marshaling code for Open Wire Format for **DestinationInfoMarshaller** (p. 1218).*
- class **DiscoveryEventMarshaller**
*Marshaling code for Open Wire Format for **DiscoveryEventMarshaller** (p. 1230).*
- class **ExceptionResponseMarshaller**
*Marshaling code for Open Wire Format for **ExceptionResponseMarshaller** (p. 1290).*
- class **FlushCommandMarshaller**
*Marshaling code for Open Wire Format for **FlushCommandMarshaller** (p. 1383).*

- class **IntegerResponseMarshaller**
*Marshaling code for Open Wire Format for **IntegerResponseMarshaller** (p. 1519).*
- class **JournalQueueAckMarshaller**
*Marshaling code for Open Wire Format for **JournalQueueAckMarshaller** (p. 1564).*
- class **JournalTopicAckMarshaller**
*Marshaling code for Open Wire Format for **JournalTopicAckMarshaller** (p. 1572).*
- class **JournalTraceMarshaller**
*Marshaling code for Open Wire Format for **JournalTraceMarshaller** (p. 1579).*
- class **JournalTransactionMarshaller**
*Marshaling code for Open Wire Format for **JournalTransactionMarshaller** (p. 1587).*
- class **KeepAliveInfoMarshaller**
*Marshaling code for Open Wire Format for **KeepAliveInfoMarshaller** (p. 1594).*
- class **LastPartialCommandMarshaller**
*Marshaling code for Open Wire Format for **LastPartialCommandMarshaller** (p. 1608).*
- class **LocalTransactionIdMarshaller**
*Marshaling code for Open Wire Format for **LocalTransactionIdMarshaller** (p. 1678).*
- class **MarshallerFactory**
Used to create marshallers for a specific version of the wire protocol.
- class **MessageAckMarshaller**
*Marshaling code for Open Wire Format for **MessageAckMarshaller** (p. 1873).*
- class **MessageDispatchMarshaller**
*Marshaling code for Open Wire Format for **MessageDispatchMarshaller** (p. 1890).*
- class **MessageDispatchNotificationMarshaller**
*Marshaling code for Open Wire Format for **MessageDispatchNotificationMarshaller** (p. 1899).*
- class **MessageIdMarshaller**
*Marshaling code for Open Wire Format for **MessageIdMarshaller** (p. 1912).*
- class **MessageMarshaller**
*Marshaling code for Open Wire Format for **MessageMarshaller** (p. 1917).*
- class **MessagePullMarshaller**
*Marshaling code for Open Wire Format for **MessagePullMarshaller** (p. 1944).*
- class **NetworkBridgeFilterMarshaller**
*Marshaling code for Open Wire Format for **NetworkBridgeFilterMarshaller** (p. 1974).*
- class **PartialCommandMarshaller**
*Marshaling code for Open Wire Format for **PartialCommandMarshaller** (p. 2079).*
- class **ProducerAckMarshaller**
*Marshaling code for Open Wire Format for **ProducerAckMarshaller** (p. 2175).*
- class **ProducerIdMarshaller**
*Marshaling code for Open Wire Format for **ProducerIdMarshaller** (p. 2186).*
- class **ProducerInfoMarshaller**
*Marshaling code for Open Wire Format for **ProducerInfoMarshaller** (p. 2195).*
- class **RemoveInfoMarshaller**

*Marshaling code for Open Wire Format for **RemoveInfoMarshaller** (p. 2271).*

- class **RemoveSubscriptionInfoMarshaller**

*Marshaling code for Open Wire Format for **RemoveSubscriptionInfoMarshaller** (p. 2280).*

- class **ReplayCommandMarshaller**

*Marshaling code for Open Wire Format for **ReplayCommandMarshaller** (p. 2287).*

- class **ResponseMarshaller**

*Marshaling code for Open Wire Format for **ResponseMarshaller** (p. 2307).*

- class **SessionIdMarshaller**

*Marshaling code for Open Wire Format for **SessionIdMarshaller** (p. 2382).*

- class **SessionInfoMarshaller**

*Marshaling code for Open Wire Format for **SessionInfoMarshaller** (p. 2390).*

- class **ShutdownInfoMarshaller**

*Marshaling code for Open Wire Format for **ShutdownInfoMarshaller** (p. 2434).*

- class **SubscriptionInfoMarshaller**

*Marshaling code for Open Wire Format for **SubscriptionInfoMarshaller** (p. 2635).*

- class **TransactionIdMarshaller**

*Marshaling code for Open Wire Format for **TransactionIdMarshaller** (p. 2770).*

- class **TransactionInfoMarshaller**

*Marshaling code for Open Wire Format for **TransactionInfoMarshaller** (p. 2778).*

- class **WireFormatInfoMarshaller**

*Marshaling code for Open Wire Format for **WireFormatInfoMarshaller** (p. 2900).*

- class **XATransactionIdMarshaller**

*Marshaling code for Open Wire Format for **XATransactionIdMarshaller** (p. 2942).*

5.23 activemq::wireformat::openwire::utils Namespace Reference

Data Structures

- class **BooleanStream**

Manages the writing and reading of boolean data streams to and from a data source such as a `DataStream` or `DataOutputStream`.

- class **HexTable**

*The **HexTable** (p. 1407) class maps hexadecimal strings to the value of an index into the table, i.e.*

- class **MessagePropertyInterceptor**

Used the base `ActiveMQMessage` class to intercept calls to get and set properties in order to capture the calls that use the reserved JMS properties and get and set them in the `OpenWire` Message properties.

5.24 activemq::wireformat::stomp Namespace Reference

Data Structures

- class **StompCommandConstants**
- class **StompFrame**

A Stomp-level message frame that encloses all messages to and from the broker.
- class **StompHelper**

*Utility Methods used when marshaling to and from **StompFrame** (p. 2587)'s.*
- class **StompWireFormat**
- class **StompWireFormatFactory**

Factory used to create the Stomp Wire Format instance.

5.25 cms Namespace Reference

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

Data Structures

- class **BytesMessage**

*A **BytesMessage** (p. 718) object is used to send a message containing a stream of unsigned bytes.*
- class **Closeable**

Interface for a class that implements the close method.
- class **CMSException**

CMS API Exception that is the base for all exceptions thrown from CMS classes.
- class **CMSProperties**

Interface for a Java-like properties object.
- class **CMSSecurityException**

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

The client's connection to its provider.
- class **ConnectionFactory**

*Defines the interface for a factory that creates connection objects, the **Connection** (p. 933) objects returned implement the CMS **Connection** (p. 933) interface and hide the CMS Provider specific implementation details behind that interface.*
- class **ConnectionMetaData**

*A **ConnectionMetaData** (p. 978) object provides information describing the **Connection** (p. 933) object.*
- class **DeliveryMode**

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

- class **Destination**
*A **Destination** (p. 1210) object encapsulates a provider-specific address.*
- class **ExceptionListener**
*If a CMS provider detects a serious problem, it notifies the client application through an **ExceptionListener** (p. 1286) that is registered with the **Connection** (p. 933).*
- class **IllegalStateException**
This exception is thrown when a method is invoked at an illegal or inappropriate time or if the provider is not in an appropriate state for the requested operation.
- class **InvalidClientIdException**
This exception must be thrown when a client attempts to set a connection's client ID to a value that is rejected by a provider.
- class **InvalidDestinationException**
This exception must be thrown when a destination either is not understood by a provider or is no longer valid.
- class **InvalidSelectorException**
This exception must be thrown when a CMS client attempts to give a provider a message selector with invalid syntax.
- class **MapMessage**
*A **MapMessage** (p. 1779) object is used to send a set of name-value pairs.*
- class **Message**
Root of all messages.
- class **MessageConsumer**
*A client uses a **MessageConsumer** (p. 1877) to receive messages from a destination.*
- class **MessageEnumeration**
Defines an object that enumerates a collection of Messages.
- class **MessageEOFException**
*This exception must be thrown when an unexpected end of stream has been reached when a **StreamMessage** (p. 2606) or **BytesMessage** (p. 718) is being read.*
- class **MessageFormatException**
This exception must be thrown when a CMS client attempts to use a data type not supported by a message or attempts to read data in a message as the wrong type.
- class **MessageListener**
*A **MessageListener** (p. 1916) object is used to receive asynchronously delivered messages.*
- class **MessageNotReadableException**
This exception must be thrown when a CMS client attempts to read a write-only message.
- class **MessageNotWriteableException**
This exception must be thrown when a CMS client attempts to write to a read-only message.
- class **MessageProducer**
*A client uses a **MessageProducer** (p. 1924) object to send messages to a **Destination** (p. 1210).*
- class **ObjectMessage**

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

- class **Queue**

An interface encapsulating a provider-specific queue name.

- class **QueueBrowser**

*This class implements in interface for browsing the messages in a **Queue** (p. 2221) without removing them.*

- class **Session**

*A **Session** (p. 2361) object is a single-threaded context for producing and consuming messages.*

- class **Startable**

Interface for a class that implements the start method.

- class **Stoppable**

Interface for a class that implements the stop method.

- class **StreamMessage**

*Interface for a **StreamMessage** (p. 2606).*

- class **TemporaryQueue**

*Defines a Temporary **Queue** (p. 2221) based **Destination** (p. 1210).*

- class **TemporaryTopic**

*Defines a Temporary **Topic** (p. 2764) based **Destination** (p. 1210).*

- class **TextMessage**

Interface for a text message.

- class **Topic**

An interface encapsulating a provider-specific topic name.

- class **TransactionInProgressException**

This exception is thrown when an operation is invalid because a transaction is in progress.

- class **TransactionRolledBackException**

*This exception must be thrown when a call to **Session.commit** (p. 2366) results in a rollback of the current transaction.*

- class **UnsupportedOperationException**

This exception must be thrown when a CMS client attempts use a CMS method that is not implemented or not supported by the CMS Provider in use.

- class **XAConnection**

*The **XAConnection** (p. 2917) interface defines an extended **Connection** (p. 933) type that is used to create **XASession** (p. 2935) objects.*

- class **XAConnectionFactory**

*The **XAConnectionFactory** (p. 2918) interface is specialized interface that defines an **ConnectionFactory** (p. 955) that creates **Connection** (p. 933) instance that will participate in XA Transactions.*

- class **XAException**

*The **XAException** (p. 2921) is thrown by the Resource Manager (RM) to inform the Transaction Manager of an error encountered by the involved transaction.*

- class **XAResource**

The **XAResource** (p. 2926) interface is a CMS mapping of the industry standard XA interface based on the X/Open CAE Specification (Distributed Transaction Processing: The XA Specification).

- class **XASession**

The **XASession** (p. 2935) interface extends the capability of **Session** (p. 2361) by adding access to a CMS provider's support for the operating inside an XA Transaction (optional).

- class **Xid**

An interface which provides a mapping for the X/Open XID transaction identifier structure.

5.25.1 Detailed Description

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

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

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

5.26 decaf Namespace Reference

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

Namespaces

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

5.26.1 Detailed Description

Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements. See the NOTICE file distributed with this work for additional information regarding copyright ownership. The ASF licenses this file to You under the Apache

License, Version 2.0 (the "License"); you may not use this file except in compliance with the License. You may obtain a copy of the License at

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

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

5.27 decaf::internal Namespace Reference

Namespaces

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

Data Structures

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

5.28 decaf::internal::io Namespace Reference

Data Structures

- class **StandardErrorOutputStream**
Wrapper Around the Standard error Output facility on the current platform.
- class **StandardInputStream**
- class **StandardOutputStream**

5.29 decaf::internal::net Namespace Reference

Namespaces

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

Data Structures

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

5.30 decaf::internal::net::ssl Namespace Reference

Namespaces

- namespace **openssl**

Data Structures

- class **DefaultSSLContext**
Default SSLContext manager for the Decaf library.
- class **DefaultSSLServerSocketFactory**
Default implementation of the SSLServerSocketFactory, this factory throws an - Exception from all its create methods to indicate that SSL is not supported, this factory is used when OpenSSL is not enabled in the builds.
- class **DefaultSSLSocketFactory**
Default implementation of the SSLSocketFactory, this factory throws an Exception from all its create methods to indicate that SSL is not supported, this factory is used when OpenSSL is not enabled in the builds.

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

Data Structures

- class **OpenSSLContextSpi**
Provides an SSLContext that wraps the OpenSSL API.

- class **OpenSSLParameters**
Container class for parameters that are Common to OpenSSL socket classes.
- class **OpenSSLServerSocket**
SSLServerSocket based on OpenSSL library code.
- class **OpenSSLServerSocketFactory**
SSLServerSocketFactory that creates Server Sockets that use OpenSSL.
- class **OpenSSLSocket**
Wraps a a Normal Socket object and extends or overrides functions in that class to make use of the OpenSSL Socket API.
- class **OpenSSLSocketException**
Subclass of the standard SocketException that knows how to produce an error message from the OpenSSL error stack.
- class **OpenSSLSocketFactory**
Client Socket Factory that creates SSL based client sockets using the OpenSSL library.
- class **OpenSSLSocketInputStream**
An output stream for reading data from an OpenSSL Socket instance.
- class **OpenSSLSocketOutputStream**
*OutputStream implementation used to write data to an **OpenSSLSocket** (p. 2014) instance.*

5.32 decaf::internal::net::tcp Namespace Reference

Data Structures

- class **TcpSocket**
Platform-independent implementation of the socket interface.
- class **TcpSocketInputStream**
Input stream for performing reads on a socket.
- class **TcpSocketOutputStream**
Output stream for performing write operations on a socket.

5.33 decaf::internal::nio Namespace Reference

Data Structures

- class **BufferFactory**
*Factory class used by static methods in the **decaf::nio** (p. 94) package to create the various default version of the NIO interfaces.*
- class **ByteArrayBuffer**
This class defines six categories of operations upon byte buffers:
- class **CharArrayBuffer**
- class **DoubleArrayBuffer**

- class **FloatArrayBuffer**
- class **IntArrayBuffer**
- class **LongArrayBuffer**
- class **ShortArrayBuffer**

5.34 decaf::internal::security Namespace Reference

Data Structures

- class **SecureRandomImpl**
Secure Random Number Generator for Unix based platforms that attempts to obtain secure bytes with high entropy from known sources.

5.35 decaf::internal::util Namespace Reference

Namespaces

- namespace **concurrent**

Data Structures

- class **ByteArrayAdapter**
This class adapts primitive type arrays to a base byte array so that the classes can inter-operate on the same base byte array without copying data.
- class **GenericResource**
*A Generic **Resource** (p. 2293) wraps some type and will delete it when the **Resource** (p. 2293) itself is deleted.*
- class **HexStringParser**
- class **Resource**
Interface for all Managed Resources in Decaf, these objects are added to the Runtime System and are destroyed at shutdown.
- class **ResourceLifecycleManager**
- class **TimerTaskHeap**
A Binary Heap implemented specifically for the Timer class in Decaf Util.

5.36 decaf::internal::util::concurrent Namespace Reference

Data Structures

- class **ConditionImpl**
- class **MutexImpl**
- class **SynchronizableImpl**

A convenience class used by some Decaf classes to implement the Synchronizable interface when there is no issues related to multiple inheritance.

- class **Transferer**

Shared internal API for dual stacks and queues.

- class **TransferQueue**

This extends Scherer-Scott dual queue algorithm, differing, among other ways, by using modes within nodes rather than marked pointers.

- class **TransferStack**

5.37 decaf::io Namespace Reference

Data Structures

- class **BlockingByteArrayInputStream**

This is a blocking version of a byte buffer stream.

- class **BufferedInputStream**

A wrapper around another input stream that performs a buffered read, where it reads more data than it needs in order to reduce the number of io operations on the input stream.

- class **BufferedOutputStream**

Wrapper around another output stream that buffers output before writing to the target output stream.

- class **ByteArrayInputStream**

*A **ByteArrayInputStream** (p. 679) contains an internal buffer that contains bytes that may be read from the stream.*

- class **ByteArrayOutputStream**

- class **Closeable**

Interface for a class that implements the close method.

- class **DataInput**

*The **DataInput** (p. 1086) interface provides for reading bytes from a binary stream and reconstructing from them data in any of the C++ primitive types.*

- class **DataInputStream**

A data input stream lets an application read primitive Java data types from an underlying input stream in a machine-independent way.

- class **DataOutput**

*The **DataOutput** (p. 1103) interface provides for converting data from any of the C++ primitive types to a series of bytes and writing these bytes to a binary stream.*

- class **DataOutputStream**

A data output stream lets an application write primitive Java data types to an output stream in a portable way.

- class **EOFException**

- class **FileDescriptor**

This class serves as an opaque wrapper around an underlying OS level resource that can be used as a source / sink for byte level data such as sockets and files.

- class **FilterInputStream**

A **FilterInputStream** (p. 1334) contains some other input stream, which it uses as its basic source of data, possibly transforming the data along the way or providing additional functionality.

- class **FilterOutputStream**

This class is the superclass of all classes that filter output streams.

- class **Flushable**

A **Flushable** (p. 1380) is a destination of data that can be flushed.

- class **InputStream**

A base class that must be implemented by all classes wishing to provide a class that reads in a stream of bytes.

- class **InputStreamReader**

An **InputStreamReader** (p. 1475) is a bridge from byte streams to character streams.

- class **InterruptedIOException**

- class **IOException**

- class **OutputStream**

Base interface for any class that wants to represent an output stream of bytes.

- class **OutputStreamWriter**

A class for turning a character stream into a byte stream.

- class **PushbackInputStream**

A **PushbackInputStream** (p. 2214) adds functionality to another input stream, namely the ability to "push back" or "unread" one byte.

- class **Reader**

- class **UnsupportedEncodingException**

Thrown when the the Character Encoding is not supported.

- class **UTFDataFormatException**

Thrown from classes that attempt to read or write a UTF-8 encoded string and an encoding error is encountered.

- class **Writer**

5.38 decaf::lang Namespace Reference

Namespaces

- namespace **exceptions**

Data Structures

- class **Appendable**

An object to which char sequences and values can be appended.

- class **ArrayPointer**

*Decaf's implementation of a Smart **Pointer** (p. 2083) that is a template on a Type and is **Thread** (p. 2703) Safe if the default Reference Counter is used.*

- class **ArrayPointerComparator**

*This implementation of **Comparator** is designed to allows objects in a **Collection** to be sorted or tested for equality based on the value of the value of the actual pointer to the array being contained in this **ArrayPointer** (p. 462).*

- class **Boolean**
- class **Byte**
- class **Character**
- class **CharSequence**

*A **CharSequence** (p. 803) is a readable sequence of char values.*

- class **Comparable**

This interface imposes a total ordering on the objects of each class that implements it.

- class **Double**
- class **Exception**
- class **Float**
- class **Integer**
- class **Iterable**

*Implementing this interface allows an object to be cast to an **Iterable** (p. 1556) type for generic collections API calls.*

- class **Long**
- class **Math**

*The class **Math** (p. 1800) contains methods for performing basic numeric operations such as the elementary exponential, logarithm, square root, and trigonometric functions.*

- class **Number**

*The abstract class **Number** (p. 1992) is the superclass of classes **Byte** (p. 614), - **Double** (p. 1235), **Float** (p. 1344), **Integer** (p. 1500), **Long** (p. 1726), and **Short** (p. 2398).*

- struct **STATIC_CAST_TOKEN**
- struct **DYNAMIC_CAST_TOKEN**
- class **Pointer**

*Decaf's implementation of a Smart **Pointer** (p. 2083) that is a template on a **Type** and is **Thread** (p. 2703) Safe if the default Reference Counter is used.*

- class **PointerComparator**

*This implementation of **Comparator** is designed to allows objects in a **Collection** to be sorted or tested for equality based on the value of the **Object** being **Pointed** to and not the value of the contained pointer in the **Pointer** (p. 2083) instance.*

- class **Readable**

*A **Readable** (p. 2235) is a source of characters.*

- class **Runnable**

Interface for a runnable object - defines a task that can be run by a thread.

- class **Runtime**
- class **Short**
- class **String**

*The **String** (p. 2620) class represents an immutable sequence of chars.*

- class **System**

*The **System** (p. 2665) class provides static methods for accessing system level resources and performing some system dependent tasks such as looking up environment values and copying memory and arrays.*

- class **Thread**

*A **Thread** (p. 2703) is a concurrent unit of execution.*

- class **ThreadGroup**
- class **Throwable**

This class represents an error that has occurred.

Functions

- template<typename T, typename R, typename U >
bool **operator==** (const **ArrayPointer**< T, R > &left, const U *right)
- template<typename T, typename R, typename U >
bool **operator==** (const U *left, const **ArrayPointer**< T, R > &right)
- template<typename T, typename R, typename U >
bool **operator!=** (const **ArrayPointer**< T, R > &left, const U *right)
- template<typename T, typename R, typename U >
bool **operator!=** (const U *left, const **ArrayPointer**< T, R > &right)
- template<typename T, typename R, typename U >
bool **operator==** (const **Pointer**< T, R > &left, const U *right)
- template<typename T, typename R, typename U >
bool **operator==** (const U *left, const **Pointer**< T, R > &right)
- template<typename T, typename R, typename U >
bool **operator!=** (const **Pointer**< T, R > &left, const U *right)
- template<typename T, typename R, typename U >
bool **operator!=** (const U *left, const **Pointer**< T, R > &right)

5.38.1 Function Documentation

- 5.38.1.1 template<typename T, typename R, typename U > bool decaf::lang::operator!= (const **Pointer**< T, R > & *left*, const U * *right*) [inline]

References decaf::lang::Pointer< T, REFCOUNTER >::get().

- 5.38.1.2 template<typename T, typename R, typename U > bool decaf::lang::operator!= (const U * *left*, const **Pointer**< T, R > & *right*) [inline]

References decaf::lang::Pointer< T, REFCOUNTER >::get().

- 5.38.1.3 template<typename T, typename R, typename U > bool decaf::lang::operator!= (const **ArrayPointer**< T, R > & *left*, const U * *right*) [inline]

References decaf::lang::ArrayPointer< T, REFCOUNTER >::get().

5.38.1.4 `template<typename T , typename R , typename U > bool decaf::lang::operator!= (const U * left, const ArrayPointer< T, R > & right) [inline]`

References `decaf::lang::ArrayPointer< T, REFCOUNTER >::get()`.

5.38.1.5 `template<typename T , typename R , typename U > bool decaf::lang::operator== (const Pointer< T, R > & left, const U * right) [inline]`

References `decaf::lang::Pointer< T, REFCOUNTER >::get()`.

5.38.1.6 `template<typename T , typename R , typename U > bool decaf::lang::operator== (const U * left, const Pointer< T, R > & right) [inline]`

References `decaf::lang::Pointer< T, REFCOUNTER >::get()`.

5.38.1.7 `template<typename T , typename R , typename U > bool decaf::lang::operator== (const ArrayPointer< T, R > & left, const U * right) [inline]`

References `decaf::lang::ArrayPointer< T, REFCOUNTER >::get()`.

5.38.1.8 `template<typename T , typename R , typename U > bool decaf::lang::operator== (const U * left, const ArrayPointer< T, R > & right) [inline]`

References `decaf::lang::ArrayPointer< T, REFCOUNTER >::get()`.

5.39 decaf::lang::exceptions Namespace Reference

Data Structures

- class **ClassCastException**
- class **IllegalArgumentException**
- class **IllegalMonitorStateException**
- class **IllegalStateException**
- class **IllegalThreadStateException**
- class **IndexOutOfBoundsException**
- class **InterruptedException**
- class **InvalidStateException**
- class **NullPointerException**
- class **NumberFormatException**
- class **RuntimeException**
- class **UnsupportedOperationException**

5.40 decaf::net Namespace Reference

Namespaces

- namespace **ssl**

Data Structures

- class **BindException**
- class **ConnectException**
- class **DatagramPacket**
Class that represents a single datagram packet.
- class **HttpRetryException**
- class **Inet4Address**
- class **Inet6Address**
- class **InetAddress**
Represents an IP address.
- class **InetSocketAddress**
- class **MalformedURLException**
- class **NoRouteToHostException**
- class **PortUnreachableException**
- class **ProtocolException**
- class **ServerSocket**
This class implements server sockets.
- class **ServerSocketFactory**
Class used to create Server Sockets, subclasses can be created that create certain types of server sockets according to specific policies.
- class **Socket**
- class **SocketAddress**
*Base class for protocol specific **Socket** (p. 2452) addresses.*
- class **SocketError**
Static utility class to simplify handling of error codes for socket operations.
- class **SocketException**
Exception for errors when manipulating sockets.
- class **SocketFactory**
*The **SocketFactory** (p. 2473) is used to create **Socket** (p. 2452) objects and can be sub-classed to provide other types of Sockets or Sockets with varying configurations.*
- class **SocketImpl**
*Acts as a base class for all physical **Socket** (p. 2452) implementations.*
- class **SocketImplFactory**
Factory class interface for a Factory that creates SocketImpl objects.
- class **SocketOptions**
- class **SocketTimeoutException**
- class **UnknownHostException**

- class **UnknownServiceException**
- class **URI**

*This class represents an instance of a **URI** (p. 2828) as defined by RFC 2396.*

- class **URISyntaxException**
- class **URL**

*Class **URL** (p. 2868) represents a Uniform Resource Locator, a pointer to a "resource" on the World Wide Web.*

- class **URLDecoder**
- class **URLEncoder**

5.41 decaf::net::ssl Namespace Reference

Data Structures

- class **SSLContext**
*Represents an implementation of the Secure **Socket** (p. 2452) Layer for streaming based sockets.*
- class **SSLContextSpi**
*Defines the interface that should be provided by an **SSLContext** (p. 2495) provider.*
- class **SSLParameters**
- class **SSLServerSocket**
Represents a server socket that is used to accept connections from clients using the Secure Sockets protocol or the Top Level Security protocol.
- class **SSLServerSocketFactory**
Factory class interface that provides methods to create SSL Server Sockets.
- class **SSLSocket**
- class **SSLSocketFactory**
*Factory class interface for a **SocketFactory** (p. 2473) that can create **SSLSocket** (p. 2513) objects.*

5.42 decaf::nio Namespace Reference

Data Structures

- class **Buffer**
A container for data of a specific primitive type.
- class **BufferOverflowException**
- class **BufferUnderflowException**
- class **ByteBuffer**
This class defines six categories of operations upon byte buffers:
- class **CharBuffer**
This class defines four categories of operations upon character buffers:
- class **DoubleBuffer**

This class defines four categories of operations upon double buffers:

- class **FloatBuffer**

This class defines four categories of operations upon float buffers:

- class **IntBuffer**

This class defines four categories of operations upon int buffers:

- class **InvalidMarkException**
- class **LongBuffer**

This class defines four categories of operations upon long long buffers:

- class **ReadOnlyBufferException**
- class **ShortBuffer**

This class defines four categories of operations upon short buffers:

5.43 decaf::security Namespace Reference

Namespaces

- namespace **auth**
- namespace **cert**

Data Structures

- class **GeneralSecurityException**
- class **InvalidKeyException**
- class **Key**

*The **Key** (p. 1598) interface is the top-level interface for all keys.*

- class **KeyException**
- class **KeyManagementException**
- class **NoSuchAlgorithmException**
- class **NoSuchProviderException**
- class **Principal**

Base interface for a principal, which can represent an individual or organization.

- class **PublicKey**

A public key.

- class **SecureRandom**
- class **SecureRandomSpi**

Interface class used by Security Service Providers to implement a source of secure random bytes.

- class **SignatureException**

5.44 decaf::security::auth Namespace Reference

Namespaces

- namespace **x500**

5.45 decaf::security::auth::x500 Namespace Reference

Data Structures

- class **X500Principal**

5.46 decaf::security::cert Namespace Reference

Data Structures

- class **Certificate**
Base interface for all identity certificates.
- class **CertificateEncodingException**
- class **CertificateException**
- class **CertificateExpiredException**
- class **CertificateNotYetValidException**
- class **CertificateParsingException**
- class **X509Certificate**
Base interface for all identity certificates.

5.47 decaf::util Namespace Reference

Namespaces

- namespace **comparators**
- namespace **concurrent**
- namespace **logging**
- namespace **zip**

Data Structures

- class **AbstractCollection**
*This class provides a skeletal implementation of the **Collection** (p. 851) interface, to minimize the effort required to implement this interface.*
- class **AbstractList**
*This class provides a skeletal implementation of the **List** (p. 1658) interface to minimize the effort required to implement this interface backed by a "random access" data store (such as an array).*
- class **AbstractMap**
*This class provides a skeletal implementation of the **Map** (p. 1768) interface, to minimize the effort required to implement this interface.*
- class **AbstractQueue**
*This class provides skeletal implementations of some **Queue** (p. 2222) operations.*

- class **AbstractSequentialList**
*This class provides a skeletal implementation of the **List** (p. 1658) interface to minimize the effort required to implement this interface backed by a "sequential access" data store (such as a linked list).*
- class **AbstractSet**
*This class provides a skeletal implementation of the **Set** (p. 2397) interface to minimize the effort required to implement this interface.*
- class **ArrayList**
- class **Arrays**
- class **Collection**
The root interface in the collection hierarchy.
- class **Comparator**
A comparison function, which imposes a total ordering on some collection of objects.
- class **ConcurrentModificationException**
- class **Date**
Wrapper class around a time value in milliseconds.
- class **Deque**
*Defines a 'Double ended **Queue** (p. 2222)' interface that allows for insertion and removal of elements from both ends.*
- class **Iterator**
Defines an object that can be used to iterate over the elements of a collection.
- class **LinkedList**
*A complete implementation of the **List** (p. 1658) interface using a doubly linked list data structure.*
- class **List**
An ordered collection (also known as a sequence).
- class **ListIterator**
An iterator for lists that allows the programmer to traverse the list in either direction, modify the list during iteration, and obtain the iterator's current position in the list.
- class **Map**
***Map** (p. 1768) template that wraps around a `std::map` to provide a more user-friendly interface and to provide common functions that do not exist in `std::map`.*
- class **NoSuchElementException**
- class **PriorityQueue**
An unbounded priority queue based on a binary heap algorithm.
- class **Properties**
Java-like properties class for mapping string names to string values.
- class **Queue**
A kind of collection provides advanced operations than other basic collections, such as insertion, extraction, and inspection.
- class **Random**
***Random** (p. 2229) Value Generator which is used to generate a stream of pseudorandom numbers.*
- class **Set**
A collection that contains no duplicate elements.

- class **StlList**
***List** (p. 1658) class that wraps the STL list object to provide a simpler interface and additional methods not provided by the STL type.*
- class **StlMap**
***Map** (p. 1768) template that wraps around a `std::map` to provide a more user-friendly interface and to provide common functions that do not exist in `std::map`.*
- class **StlQueue**
*The **Queue** (p. 2222) class accepts messages with an `psuh(m)` command where `m` is the message to be queued.*
- class **StlSet**
***Set** (p. 2397) template that wraps around a `std::set` to provide a more user-friendly interface and to provide common functions that do not exist in `std::set`.*
- class **StringTokenizer**
Class that allows for parsing of string based on Tokens.
- class **Timer**
A facility for threads to schedule tasks for future execution in a background thread.
- class **TimerTask**
*A Base class for a task object that can be scheduled for one-time or repeated execution by a **Timer** (p. 2739).*
- class **UUID**
*A class that represents an immutable universally unique identifier (**UUID** (p. 2877)).*

5.48 decaf::util::comparators Namespace Reference

Data Structures

- class **Less**
*Simple **Less** (p. 1612) **Comparator** (p. 888) that compares to elements to determine if the first is less than the second.*

5.49 decaf::util::concurrent Namespace Reference

Namespaces

- namespace **atomic**
- namespace **locks**

Data Structures

- class **ConditionHandle**
- class **MutexHandle**
- class **AbstractExecutorService**

*Provides a default implementation for the methods of the **ExecutorService** (p. 1302) interface.*

- class **BlockingQueue**

*A **decaf::util::Queue** (p. 2222) that additionally supports operations that wait for the queue to become non-empty when retrieving an element, and wait for space to become available in the queue when storing an element.*

- class **BrokenBarrierException**

- class **Callable**

A task that returns a result and may throw an exception.

- class **CancellationException**

- class **ConcurrentMap**

*Interface for a **Map** (p. 1768) type that provides additional atomic putIfAbsent, remove, and replace methods alongside the already available **Map** (p. 1768) interface.*

- class **ConcurrentStlMap**

***Map** (p. 1768) template that wraps around a `std::map` to provide a more user-friendly interface and to provide common functions that do not exist in `std::map`.*

- class **CopyOnWriteArrayList**

- class **CopyOnWriteArraySet**

*Since the **CopyOnWriteArraySet** (p. 1050) and the **CopyOnWriteArrayList** (p. 1030) share much of the same operational semantics this class uses the **CopyOnWriteArrayList** (p. 1030) for all its underlying operations.*

- class **CountDownLatch**

- class **Delayed**

A mix-in style interface for marking objects that should be acted upon after a given delay.

- class **ExecutionException**

- class **Executor**

*An object that executes submitted **decaf.lang Runnable** (p. 2312) tasks.*

- class **Executors**

*Implements a set of utilities for use with **Executors** (p. 1299), **ExecutorService** (p. 1302), **ThreadFactory** (p. 2714), and **Callable** (p. 746) types, as well as providing factory methods for instance of these types configured for the most common use cases.*

- class **ExecutorService**

*An **Executor** (p. 1297) that provides methods to manage termination and methods that can produce a **Future** (p. 1390) for tracking progress of one or more asynchronous tasks.*

- class **Future**

*A **Future** (p. 1390) represents the result of an asynchronous computation.*

- class **LinkedBlockingQueue**

*A **BlockingQueue** (p. 538) derivative that allows for a bound to be placed on the number of elements that can be enqueued at any one time.*

- class **Lock**

A wrapper class around a given synchronization mechanism that provides automatic release upon destruction.

- class **Mutex**

Mutex (p. 1960) object that offers recursive support on all platforms as well as providing the ability to use the standard wait / notify pattern used in languages like Java.

- class **RejectedExecutionException**
- class **RejectedExecutionHandler**

A handler for tasks that cannot be executed by a **ThreadPoolExecutor** (p. 2715).

- class **Semaphore**

A counting semaphore.

- class **Synchronizable**

The interface for all synchronizable objects (that is, objects that can be locked and unlocked).

- class **SynchronousQueue**

A **blocking queue** (p. 538) in which each insert operation must wait for a corresponding remove operation by another thread, and vice versa.

- class **ThreadFactory**

public interface **ThreadFactory** (p. 2714)

- class **ThreadPoolExecutor**

Defines a Thread Pool object that implements the functionality of pooling threads to perform user tasks.

- class **TimeoutException**
- class **TimeUnit**

A **TimeUnit** (p. 2756) represents time durations at a given unit of granularity and provides utility methods to convert across units, and to perform timing and delay operations in these units.

5.50 decaf::util::concurrent::atomic Namespace Reference

Data Structures

- class **AtomicBoolean**

A boolean value that may be updated atomically.

- class **AtomicInteger**

An int value that may be updated atomically.

- class **AtomicRefCounter**

- class **AtomicReference**

An Pointer reference that may be updated atomically.

5.51 decaf::util::concurrent::locks Namespace Reference

Data Structures

- class **AbstractOwnableSynchronizer**

Base class for locks that provide the notion of Ownership, the types of locks that are implemented using this base class would be owned by one specific Thread at any given time.

- class **Condition**
***Condition** (p. 921) factors out the **Mutex** (p. 1960) monitor methods (wait, notify and notifyAll) into distinct objects to give the effect of having multiple wait-sets per object, by combining them with the use of arbitrary **Lock** (p. 1684) implementations.*
- class **Lock**
***Lock** (p. 1684) implementations provide more extensive locking operations than can be obtained using synchronized statements.*
- class **LockSupport**
Basic thread blocking primitives for creating locks and other synchronization classes.
- class **ReadWriteLock**
*A **ReadWriteLock** (p. 2246) maintains a pair of associated locks, one for read-only operations and one for writing.*
- class **ReentrantLock**
*A reentrant mutual exclusion **Lock** (p. 1684) with extended capabilities.*

5.52 decaf::util::logging Namespace Reference

Data Structures

- class **ConsoleHandler**
*This **Handler** (p. 1401) publishes log records to System.err.*
- class **ErrorManager**
***ErrorManager** (p. 1277) objects can be attached to Handlers to process any error that occur on a **Handler** (p. 1401) during Logging.*
- class **Filter**
*A **Filter** (p. 1333) can be used to provide fine grain control over what is logged, beyond the control provided by log levels.*
- class **Formatter**
*A **Formatter** (p. 1387) provides support for formatting LogRecords.*
- class **Handler**
*A **Handler** (p. 1401) object takes log messages from a **Logger** (p. 1693) and exports them.*
- class **Level**
*The **Level** (p. 1616) class defines a set of standard logging levels that can be used to control logging output.*
- class **Logger**
*A **Logger** (p. 1693) object is used to log messages for a specific system or application component.*
- class **LoggerHierarchy**
- class **LogManager**
*There is a single global **LogManager** (p. 1712) object that is used to maintain a set of shared state about Loggers and log services.*
- class **LogRecord**
***LogRecord** (p. 1719) objects are used to pass logging requests between the logging framework and individual log Handlers.*

- class **LogWriter**
- class **MarkBlockLogger**
 - Defines a class that can be used to mark the entry and exit from scoped blocks.*
- class **PropertiesChangeListener**
 - Defines the interface that classes can use to listen for change events on **Properties** (p. 2200).*
- class **SimpleFormatter**
 - Print a brief summary of the **LogRecord** (p. 1719) in a human readable format.*
- class **SimpleLogger**
- class **StreamHandler**
 - Stream based logging **Handler** (p. 1401).*
- class **XMLFormatter**
 - Format a **LogRecord** (p. 1719) into a standard XML format.*

Enumerations

- enum **Levels** { **Off**, **Null**, **Markblock**, **Debug**, **Info**, **Warn**, **Error**, **Fatal**, **Throwing** }
 - Defines an enumeration for logging levels.*

5.52.1 Enumeration Type Documentation

5.52.1.1 enum decaf::util::logging::Levels

Defines an enumeration for logging levels.

Enumerator:

Off
Null
Markblock
Debug
Info
Warn
Error
Fatal
Throwing

5.53 decaf::util::zip Namespace Reference

Data Structures

- class **Adler32**

Class that can be used to compute an Adler-32 **Checksum** (p. 810) for a data stream.

- class **CheckedInputStream**

An implementation of a *FilterInputStream* that will maintain a **Checksum** (p. 810) of the bytes read, the **Checksum** (p. 810) can then be used to verify the integrity of the input stream.

- class **CheckedOutputStream**

An implementation of a *FilterOutputStream* that will maintain a **Checksum** (p. 810) of the bytes written, the **Checksum** (p. 810) can then be used to verify the integrity of the output stream.

- class **Checksum**

An interface used to represent **Checksum** (p. 810) values in the Zip package.

- class **CRC32**

Class that can be used to compute a CRC-32 checksum for a data stream.

- class **DataFormatException**

- class **Deflater**

This class compresses data using the DEFLATE algorithm (see specification).

- class **DeflaterOutputStream**

Provides a *FilterOutputStream* instance that compresses the data before writing it to the wrapped *OutputStream*.

- class **Inflater**

This class uncompresses data that was compressed using the DEFLATE algorithm (see specification).

- class **InflaterInputStream**

A *FilterInputStream* that decompresses data read from the wrapped *InputStream* instance.

- class **ZipException**

5.54 std Namespace Reference

Data Structures

- struct **less< decaf::lang::ArrayPointer< T > >**

An override of the *less* function object so that the *Pointer* objects can be stored in STL Maps, etc.

- struct **less< decaf::lang::Pointer< T > >**

An override of the *less* function object so that the *Pointer* objects can be stored in STL Maps, etc.

Chapter 6

Data Structure Documentation

6.1 decaf::util::concurrent::ThreadPoolExecutor::AbortPolicy - Class Reference

Handler policy for tasks that are rejected upon a call to **ThreadPoolExecutor::execute** (p. 2724) this class always throws a **RejectedExecutionException** (p. 2263).

```
#include <src/main/decaf/util/concurrent/ThreadPoolExecutor.h>
```

Inheritance diagram for decaf::util::concurrent::ThreadPoolExecutor::AbortPolicy:

Public Member Functions

- **AbortPolicy** ()
- virtual **~AbortPolicy** ()
- virtual void **rejectedExecution** (decaf::lang::Runnable *task, **ThreadPoolExecutor** *executer **DECAF_UNUSED**)

6.1.1 Detailed Description

Handler policy for tasks that are rejected upon a call to **ThreadPoolExecutor::execute** (p. 2724) this class always throws a **RejectedExecutionException** (p. 2263).

Since

1.0

6.1.2 Constructor & Destructor Documentation

6.1.2.1 **decaf::util::concurrent::ThreadPoolExecutor::AbortPolicy::AbortPolicy ()** [inline]

6.1.2.2 **virtual decaf::util::concurrent::ThreadPoolExecutor::AbortPolicy::~~AbortPolicy ()** [inline, virtual]

6.1.3 Member Function Documentation

6.1.3.1 **virtual void decaf::util::concurrent::ThreadPoolExecutor::AbortPolicy::rejectedExecution (decaf::lang::Runnable * *task*, ThreadPoolExecutor **executer* *DECAF_UNUSED*)** [inline, virtual]

The documentation for this class was generated from the following file:

- src/main/decaf/util/concurrent/**ThreadPoolExecutor.h**

6.2 decaf::util::AbstractCollection< E > Class Template - Reference

This class provides a skeletal implementation of the **Collection** (p. 851) interface, to minimize the effort required to implement this interface.

```
#include <src/main/decaf/util/AbstractCollection.h>
```

Inheritance diagram for decaf::util::AbstractCollection< E >:

Public Member Functions

- **AbstractCollection ()**
- virtual **~AbstractCollection ()**
- **AbstractCollection< E > & operator= (const AbstractCollection< E > &collection)**

Assignment Operator, copy element from the source collection to this collection after clearing any element stored in this collection.

- virtual void **clear ()**

Removes all of the elements from this collection (optional operation).

- virtual bool **contains** (const E &value) const

Returns true if this collection contains the specified element.

*More formally, returns true if and only if this collection contains at least one element *e* such that (value == NULL ? e == NULL : value == e).*

Parameters

value	<i>The value to check for presence in the collection.</i>
-------	---

Returns

true if there is at least one of the elements in the collection

Exceptions

NullPointerException	<i>if the Collection (p. 851) contains pointers and the Collection (p. 851) does not allow for NULL elements (optional check).</i>
----------------------	--

- virtual bool **containsAll** (const **Collection**< E > &collection) const

Returns true if this collection contains all of the elements in the specified collection.

Parameters

collection	<i>The Collection (p. 851) to compare to this one.</i>
------------	---

Exceptions

NullPointerException	<i>if the Collection (p. 851) contains pointers and the Collection (p. 851) does not allow for NULL elements (optional check).</i>
----------------------	--

- virtual bool **equals** (const **Collection**< E > &collection) const

*Answers true if this **Collection** (p. 851) and the one given are the same size and if each element contained in the **Collection** (p. 851) given is equal to an element contained in this collection.*

- virtual void **copy** (const **Collection**< E > &collection)

*Renders this **Collection** (p. 851) as a Copy of the given **Collection** (p. 851).*

- virtual bool **isEmpty** () const

Returns true if this collection contains no elements.

- virtual bool **add** (const E &value **DECAF_UNUSED**)

- virtual bool **addAll** (const **Collection**< E > &collection)

Adds all of the elements in the specified collection to this collection.

The behavior of this operation is undefined if the specified collection is modified while the operation is in progress. (This implies that the behavior of this call is undefined if the specified collection is this collection, and this collection is nonempty.)

Parameters

collection	<i>The Collection (p. 851) whose elements are added to this one.</i>
------------	---

Returns

true if this collection changed as a result of the call

Exceptions

UnsupportedOperationException	<i>if this is an unmodifiable collection.</i>
NullPointerException	<i>if the Collection (p. 851) is a container of pointers and does not allow NULL values.</i>
IllegalArgumentException	<i>if some property of an element prevents it from being added to this collection</i>
IllegalStateException	<i>if an element cannot be added at this time due to insertion restrictions.</i>

- virtual bool **remove** (const E &value)

Removes a single instance of the specified element from the collection.

More formally, removes an element e such that (value == NULL ? e == NULL : value == e), if this collection contains one or more such elements. Returns true if this collection

contained the specified element (or equivalently, if this collection changed as a result of the call).

Parameters

value	The reference to the element to remove from this Collection (p. 851).
-------	--

Returns

true if the collection was changed, false otherwise.

Exceptions

UnsupportedOperationException	if this is an unmodifiable collection.
NullPointerException	if the Collection (p. 851) is a container of pointers and does not allow NULL values.

- virtual bool **removeAll** (const **Collection**< E > &collection)

Removes all this collection's elements that are also contained in the specified collection (optional operation).

After this call returns, this collection will contain no elements in common with the specified collection.

Parameters

collection	The Collection (p. 851) whose elements are to be removed from this one.
------------	--

Returns

true if the collection changed as a result of this call.

Exceptions

UnsupportedOperationException	if this is an unmodifiable collection.
NullPointerException	if the Collection (p. 851) is a container of pointers and does not allow NULL values.

- virtual bool **retainAll** (const **Collection**< E > &collection)

Retains only the elements in this collection that are contained in the specified collection (optional operation).

In other words, removes from this collection all of its elements that are not contained in the specified collection.

Parameters

collection	The Collection (p. 851) whose elements are to be retained.
------------	---

Returns

true if the collection changed as a result of this call.

Exceptions

UnsupportedOperationException	if this is an unmodifiable collection.
NullPointerException	if the Collection (p. 851) is a container of pointers and does not allow NULL values.

- virtual std::vector< E > **toArray** () const

Answers an STL vector containing copies of all elements contained in this **Collection** (p. 851).

- virtual void **lock** ()
Locks the object.
- virtual bool **tryLock** ()
Attempts to Lock the object, if the lock is already held by another thread than this method returns false.
- virtual void **unlock** ()
Unlocks the object.
- virtual void **wait** ()
Waits on a signal from this object, which is generated by a call to Notify.
- virtual void **wait** (long long millisecs)
Waits on a signal from this object, which is generated by a call to Notify.
- virtual void **wait** (long long millisecs, int nanos)
Waits on a signal from this object, which is generated by a call to Notify.
- virtual void **notify** ()
Signals a waiter on this object that it can now wake up and continue.
- virtual void **notifyAll** ()
Signals the waiters on this object that it can now wake up and continue.

Protected Attributes

- **util::concurrent::Mutex mutex**

6.2.1 Detailed Description

`template<typename E>class decaf::util::AbstractCollection< E >`

This class provides a skeletal implementation of the **Collection** (p.851) interface, to minimize the effort required to implement this interface.

To implement an unmodifiable collection, the programmer needs only to extend this class and provide implementations for the iterator and size methods. (The iterator returned by the iterator method must implement hasNext and next.)

To implement a modifiable collection, the programmer must additionally override this class's add method (which otherwise throws an UnsupportedOperationException), and the iterator returned by the iterator method must additionally implement its remove method.

The programmer should generally provide a void (no argument) and **Collection** (p. 851) constructor, as per the recommendation in the **Collection** (p.851) interface specification.

The documentation for each non-abstract method in this class describes its implementation in detail. Each of these methods may be overridden if the collection being implemented admits a more efficient implementation.

Since

1.0

6.2.2 Constructor & Destructor Documentation

6.2.2.1 `template<typename E> decaf::util::AbstractCollection< E
>::AbstractCollection () [inline]`

6.2.2.2 `template<typename E> virtual decaf::util::AbstractCollection< E
>::~~AbstractCollection () [inline, virtual]`

6.2.3 Member Function Documentation

6.2.3.1 `template<typename E> virtual bool decaf::util::AbstractCollection< E >::add (
const E &value DECAF_UNUSED) [inline, virtual]`

This implementation always throws an `UnsupportedOperationException`.

Referenced by `decaf::util::AbstractCollection< cms::Connection * >::addAll()`, `decaf::util::AbstractCollection< cms::Connection * >::copy()`, and `decaf::util::AbstractCollection< cms::Connection * >::operator=()`.

6.2.3.2 `template<typename E> virtual bool decaf::util::AbstractCollection< E
>::addAll (const Collection< E > &collection) [inline, virtual]`

Adds all of the elements in the specified collection to this collection.

The behavior of this operation is undefined if the specified collection is modified while the operation is in progress. (This implies that the behavior of this call is undefined if the specified collection is this collection, and this collection is nonempty.)

Parameters

<i>collection</i>	The Collection (p. 851) whose elements are added to this one.
-------------------	--

Returns

true if this collection changed as a result of the call

Exceptions

<i>Unsupported- OperationExceptio</i>	if this is an unmodifiable collection.
<i>NullPointerException</i>	if the Collection (p. 851) is a container of pointers and does not allow NULL values.
<i>IllegalArgument- Exception</i>	if some property of an element prevents it from being added to this collection
<i>IllegalStateException</i>	if an element cannot be added at this time due to insertion restrictions.

This implementation iterates over the specified collection, and adds each object returned by the iterator to this collection, in turn.

Note that this implementation will throw an `UnsupportedOperationException` unless `add` is overridden (assuming the specified collection is non-empty).

Implements **decaf::util::Collection< E >** (p. 855).

Reimplemented in **decaf::util::StlList< E >** (p. 2543), **decaf::util::ArrayList< E >** (p. 450), **decaf::util::LinkedList< E >** (p. 1641), **decaf::util::LinkedList< Pointer< Transport > >** (p. 1641), **decaf::util::LinkedList< cms::MessageConsumer * >** (p. 1641), **decaf::util::LinkedList< CompositeTask * >** (p. 1641), **decaf::util::LinkedList< URI >** (p. 1641), **decaf::util::LinkedList< Pointer< MessageDispatch > >** (p. 1641), **decaf::util::LinkedList< Pointer< DestinationInfo > >** (p. 1641), **decaf::util::LinkedList< PrimitiveValueNode >** (p. 1641), **decaf::util::LinkedList< Pointer< Command > >** (p. 1641), **decaf::util::LinkedList< Pointer< BackupTransport > >** (p. 1641), **decaf::util::LinkedList< cms::MessageProducer * >** (p. 1641), **decaf::util::LinkedList< cms::Destination * >** (p. 1641), **decaf::util::LinkedList< cms::Session * >** (p. 1641), **decaf::util::LinkedList< cms::Connection * >** (p. 1641), **decaf::util::concurrent::CopyOnWriteArraySet< E >** (p. 1055), and **decaf::util::AbstractQueue< E >** (p. 140).

6.2.3.3 `template<typename E> virtual void decaf::util::AbstractCollection< E >::clear () [inline, virtual]`

Removes all of the elements from this collection (optional operation).

The collection will be empty after this method returns.

This implementation iterates over this collection, removing each element using the **Iterator.remove** (p. 1560) operation. Most implementations will probably choose to override this method for efficiency.

Note that this implementation will throw an `UnsupportedOperationException` if the iterator returned by this collection's iterator method does not implement the `remove` method and this collection is non-empty.

Exceptions

<i>Unsupported- OperationException</i>	if the clear operation is not supported by this collection
--	--

Implements **decaf::util::Collection< E >** (p. 856).

Reimplemented in **decaf::util::AbstractList< E >** (p. 127), **decaf::util::AbstractList< Pointer< Transport > >** (p. 127), **decaf::util::AbstractList< cms::MessageConsumer * >** (p. 127), **decaf::util::AbstractList< CompositeTask * >** (p. 127), **decaf::util::AbstractList< URI >** (p. 127), **decaf::util::AbstractList< Pointer< MessageDispatch > >** (p. 127), **decaf::util::AbstractList< Pointer< DestinationInfo > >** (p. 127), **decaf::util::AbstractList< PrimitiveValueNode >** (p. 127), **decaf::util::AbstractList< Pointer< Command > >** (p. 127), **decaf::util::AbstractList< Pointer< BackupTransport > >** (p. 127), **decaf::util::AbstractList< cms::MessageProducer * >** (p. 127), **decaf::util::AbstractList< cms::Destination * >** (p. 127), **decaf::util::AbstractList< cms::Session * >** (p. 127), **decaf::util::AbstractList< cms::Connection * >** (p. 127), **decaf::util::StlList< E >** (p. 2545),

`decaf::util::PriorityQueue< E >` (p. 2166), `decaf::util::concurrent::SynchronousQueue< E >` (p. 2658), `decaf::util::concurrent::LinkedBlockingQueue< E >` (p. 1624), `decaf::util::LinkedList< E >` (p. 1644), `decaf::util::LinkedList< Pointer< Transport > >` (p. 1644), `decaf::util::LinkedList< cms::MessageConsumer * >` (p. 1644), `decaf::util::LinkedList< CompositeTask * >` (p. 1644), `decaf::util::LinkedList< URI >` (p. 1644), `decaf::util::LinkedList< Pointer< MessageDispatch > >` (p. 1644), `decaf::util::LinkedList< Pointer< DestinationInfo > >` (p. 1644), `decaf::util::LinkedList< PrimitiveValueNode >` (p. 1644), `decaf::util::LinkedList< Pointer< Command > >` (p. 1644), `decaf::util::LinkedList< Pointer< BackupTransport > >` (p. 1644), `decaf::util::LinkedList< cms::MessageProducer * >` (p. 1644), `decaf::util::LinkedList< cms::Destination * >` (p. 1644), `decaf::util::LinkedList< cms::Session * >` (p. 1644), `decaf::util::LinkedList< cms::Connection * >` (p. 1644), `decaf::util::StlSet< E >` (p. 2578), `decaf::util::StlSet< Pointer< Synchronization > >` (p. 2578), `decaf::util::StlSet< Resource * >` (p. 2578), `decaf::util::ArrayList< E >` (p. 452), `decaf::util::AbstractQueue< E >` (p. 141), and `decaf::util::concurrent::CopyOnWriteArraySet< E >` (p. 1056).

Referenced by `decaf::util::AbstractCollection< cms::Connection * >::copy()`, and `decaf::util::AbstractCollection< cms::Connection * >::operator=()`.

6.2.3.4 `template<typename E> virtual bool decaf::util::AbstractCollection< E >::contains (const E & value) const` [`inline`, `virtual`]

Returns true if this collection contains the specified element.

More formally, returns true if and only if this collection contains at least one element `e` such that `(value == NULL ? e == NULL : value == e)`.

Parameters

<i>value</i>	The value to check for presence in the collection.
--------------	--

Returns

true if there is at least one of the elements in the collection

Exceptions

<i>NullPointerException</i>	if the Collection (p. 851) contains pointers and the Collection (p. 851) does not allow for NULL elements (optional check).
-----------------------------	---

This implementation iterates over the elements in the collection, checking each element in turn for equality with the specified element.

Implements `decaf::util::Collection< E >` (p. 857).

Reimplemented in `decaf::util::StlList< E >` (p. 2545), `decaf::util::ArrayList< E >` (p. 452), `decaf::util::LinkedList< E >` (p. 1644), `decaf::util::LinkedList< Pointer< Transport > >` (p. 1644), `decaf::util::LinkedList< cms::MessageConsumer * >` (p. 1644), `decaf::util::LinkedList< CompositeTask * >` (p. 1644), `decaf::util::LinkedList< URI >` (p. 1644), `decaf::util::LinkedList< Pointer< MessageDispatch`

> > (p. 1644), **decaf::util::LinkedList< Pointer< DestinationInfo > >** (p. 1644), **decaf::util::LinkedList< PrimitiveValueNode >** (p. 1644), **decaf::util::LinkedList< Pointer< Command > >** (p. 1644), **decaf::util::LinkedList< Pointer< BackupTransport > >** (p. 1644), **decaf::util::LinkedList< cms::MessageProducer * >** (p. 1644), **decaf::util::LinkedList< cms::Destination * >** (p. 1644), **decaf::util::LinkedList< cms::Session * >** (p. 1644), **decaf::util::LinkedList< cms::Connection * >** (p. 1644), **decaf::util::StlSet< E >** (p. 2578), **decaf::util::StlSet< Pointer< Synchronization > >** (p. 2578), **decaf::util::StlSet< Resource * >** (p. 2578), and **decaf::util::concurrent::CopyOnWriteArraySet< E >** (p. 1056).

Referenced by **decaf::util::AbstractCollection< cms::Connection * >::containsAll()**.

```
6.2.3.5  template<typename E> virtual bool decaf::util::AbstractCollection< E
        >::containsAll ( const Collection< E > & collection ) const  [inline,
        virtual]
```

Returns true if this collection contains all of the elements in the specified collection.

Parameters

<i>collection</i>	The Collection (p. 851) to compare to this one.
-------------------	--

Exceptions

<i>NullPointerException</i>	if the Collection (p. 851) contains pointers and the Collection (p. 851) does not allow for NULL elements (optional check).
-----------------------------	---

This implementation iterates over the specified collection, checking each element returned by the iterator in turn to see if it's contained in this collection. If all elements are so contained true is returned, otherwise false.

Implements **decaf::util::Collection< E >** (p. 858).

Reimplemented in **decaf::util::concurrent::SynchronousQueue< E >** (p. 2658), and **decaf::util::concurrent::CopyOnWriteArraySet< E >** (p. 1057).

Referenced by **decaf::util::AbstractCollection< cms::Connection * >::equals()**.

```
6.2.3.6  template<typename E> virtual void decaf::util::AbstractCollection< E >::copy
        ( const Collection< E > & collection )  [inline, virtual]
```

Renders this **Collection** (p. 851) as a Copy of the given **Collection** (p. 851).

The default implementation iterates over the contents of the given collection adding each to this collection after first calling this **Collection** (p. 851)'s clear method.

Parameters

<i>collection</i>	The collection to mirror.
-------------------	---------------------------

Exceptions

<i>UnsupportedOperationException</i>	if this is an unmodifiable collection.
<i>IllegalStateException</i>	if the elements cannot be added at this time due to insertion restrictions.

Implements **decaf::util::Collection**< **E** > (p. 859).

Reimplemented in **decaf::util::StlList**< **E** > (p. 2546), **decaf::util::LinkedList**< **E** > (p. 1645), **decaf::util::LinkedList**< **Pointer**< **Transport** > > (p. 1645), **decaf::util::LinkedList**< **cms::MessageConsumer** * > (p. 1645), **decaf::util::LinkedList**< **CompositeTask** * > (p. 1645), **decaf::util::LinkedList**< **URI** > (p. 1645), **decaf::util::LinkedList**< **Pointer**< **MessageDispatch** > > (p. 1645), **decaf::util::LinkedList**< **Pointer**< **DestinationInfo** > > (p. 1645), **decaf::util::LinkedList**< **PrimitiveValueNode** > (p. 1645), **decaf::util::LinkedList**< **Pointer**< **Command** > > (p. 1645), **decaf::util::LinkedList**< **Pointer**< **BackupTransport** > > (p. 1645), **decaf::util::LinkedList**< **cms::MessageProducer** * > (p. 1645), **decaf::util::LinkedList**< **cms::Destination** * > (p. 1645), **decaf::util::LinkedList**< **cms::Session** * > (p. 1645), **decaf::util::LinkedList**< **cms::Connection** * > (p. 1645), **decaf::util::StlSet**< **E** > (p. 2579), **decaf::util::StlSet**< **Pointer**< **Synchronization** > > (p. 2579), **decaf::util::StlSet**< **Resource** * > (p. 2579), and **decaf::util::concurrent::CopyOnWriteArraySet**< **E** > (p. 1057).

```
6.2.3.7  template<typename E> virtual bool decaf::util::AbstractCollection< E
>::equals ( const Collection< E > & collection ) const  [inline,
virtual]
```

Answers true if this **Collection** (p. 851) and the one given are the same size and if each element contained in the **Collection** (p. 851) given is equal to an element contained in this collection.

Parameters

<i>collection</i>	- The Collection (p. 851) to be compared to this one.
-------------------	--

Returns

true if this **Collection** (p. 851) is equal to the one given.

Implements **decaf::util::Collection**< **E** > (p. 860).

Reimplemented in **decaf::util::StlList**< **E** > (p. 2546), **decaf::util::concurrent::SynchronousQueue**< **E** > (p. 2660), **decaf::util::StlSet**< **E** > (p. 2580), **decaf::util::StlSet**< **Pointer**< **Synchronization** > > (p. 2580), **decaf::util::StlSet**< **Resource** * > (p. 2580), and **decaf::util::concurrent::CopyOnWriteArraySet**< **E** > (p. 1058).

```
6.2.3.8  template<typename E> virtual bool decaf::util::AbstractCollection< E
>::isEmpty ( ) const  [inline, virtual]
```

Returns true if this collection contains no elements.

This implementation returns **size()** (p. 864) == 0.

Returns

true if the size method return 0.

Implements **decaf::util::Collection< E >** (p. 860).

Reimplemented in **decaf::util::StlList< E >** (p. 2548), **decaf::util::concurrent::SynchronousQueue< E >** (p. 2660), **decaf::util::LinkedList< E >** (p. 1648), **decaf::util::LinkedList< Pointer< Transport > >** (p. 1648), **decaf::util::LinkedList< cms::MessageConsumer * >** (p. 1648), **decaf::util::LinkedList< CompositeTask * >** (p. 1648), **decaf::util::LinkedList< URI >** (p. 1648), **decaf::util::LinkedList< Pointer< MessageDispatch > >** (p. 1648), **decaf::util::LinkedList< Pointer< DestinationInfo > >** (p. 1648), **decaf::util::LinkedList< PrimitiveValueNode >** (p. 1648), **decaf::util::LinkedList< Pointer< Command > >** (p. 1648), **decaf::util::LinkedList< Pointer< BackupTransport > >** (p. 1648), **decaf::util::LinkedList< cms::MessageProducer * >** (p. 1648), **decaf::util::LinkedList< cms::Destination * >** (p. 1648), **decaf::util::LinkedList< cms::Session * >** (p. 1648), **decaf::util::LinkedList< cms::Connection * >** (p. 1648), **decaf::util::StlSet< E >** (p. 2580), **decaf::util::StlSet< Pointer< Synchronization > >** (p. 2580), **decaf::util::StlSet< Resource * >** (p. 2580), **decaf::util::ArrayList< E >** (p. 454), and **decaf::util::concurrent::CopyOnWriteArraySet< E >** (p. 1058).

Referenced by **decaf::util::AbstractQueue< E >::clear()**, **decaf::util::PriorityQueue< E >::peek()**, **decaf::util::PriorityQueue< E >::poll()**, and **decaf::util::PriorityQueue< E >::remove()**.

6.2.3.9 `template<typename E> virtual void decaf::util::AbstractCollection< E >::lock () [inline, virtual]`

Locks the object.

Exceptions

<i>RuntimeException</i>	if an error occurs while locking the object.
-------------------------	--

Implements **decaf::util::concurrent::Synchronizable** (p. 2640).

Referenced by **decaf::util::concurrent::LinkedBlockingQueue< E >::clear()**, **decaf::util::concurrent::LinkedBlockingQueue< E >::remove()**, and **decaf::util::concurrent::LinkedBlockingQueue< E >::toArray()**.

6.2.3.10 `template<typename E> virtual void decaf::util::AbstractCollection< E >::notify () [inline, virtual]`

Signals a waiter on this object that it can now wake up and continue.

Must have this object locked before calling.

Exceptions

<i>IllegalMonitorStateException</i>	- if the current thread is not the owner of the the Synchronizable Object.
<i>RuntimeException</i>	if an error occurs while notifying one of the waiting threads.

Implements **decaf::util::concurrent::Synchronizable** (p. 2641).

6.2.3.11 `template<typename E> virtual void decaf::util::AbstractCollection< E >::notifyAll() [inline, virtual]`

Signals the waiters on this object that it can now wake up and continue.

Must have this object locked before calling.

Exceptions

<i>IllegalMonitorStateException</i>	- if the current thread is not the owner of the the Synchronizable Object.
<i>RuntimeException</i>	if an error occurs while notifying the waiting threads.

Implements **decaf::util::concurrent::Synchronizable** (p. 2642).

6.2.3.12 `template<typename E> AbstractCollection<E>& decaf::util::AbstractCollection< E >::operator=(const AbstractCollection< E > & collection) [inline]`

Assignment Operator, copy element from the source collection to this collection after clearing any element stored in this collection.

Parameters

<i>collection</i>	- the collection to copy
-------------------	--------------------------

Returns

a reference to this collection

6.2.3.13 `template<typename E> virtual bool decaf::util::AbstractCollection< E >::remove(const E & value) [inline, virtual]`

Removes a single instance of the specified element from the collection.

More formally, removes an element *e* such that (*value* == NULL ? *e* == NULL : *value* == *e*), if this collection contains one or more such elements. Returns true if this collection contained the specified element (or equivalently, if this collection changed as a result of the call).

Parameters

<i>value</i>	The reference to the element to remove from this Collection (p. 851).
--------------	--

Returns

true if the collection was changed, false otherwise.

Exceptions

<i>UnsupportedOperationException</i>	if this is an unmodifiable collection.
<i>NullPointerException</i>	if the Collection (p. 851) is a container of pointers and does not allow NULL values.

This implementation iterates over the collection looking for the specified element. If it finds the element, it removes the element from the collection using the iterator's remove method.

Note that this implementation throws an `UnsupportedOperationException` if the iterator returned by this collection's iterator method does not implement the remove method and this collection contains the specified object.

Implements **decaf::util::Collection< E >** (p. 861).

Reimplemented in **decaf::util::StlList< E >** (p. 2550), **decaf::util::concurrent::LinkedBlockingQueue< E >** (p. 1631), **decaf::util::PriorityQueue< E >** (p. 2170), **decaf::util::ArrayList< E >** (p. 455), **decaf::util::StlSet< E >** (p. 2580), **decaf::util::StlSet< Pointer< Synchronization > >** (p. 2580), **decaf::util::StlSet< Resource * >** (p. 2580), **decaf::util::LinkedList< E >** (p. 1654), **decaf::util::LinkedList< Pointer< Transport > >** (p. 1654), **decaf::util::LinkedList< cms::MessageConsumer * >** (p. 1654), **decaf::util::LinkedList< CompositeTask * >** (p. 1654), **decaf::util::LinkedList< URI >** (p. 1654), **decaf::util::LinkedList< Pointer< MessageDispatch > >** (p. 1654), **decaf::util::LinkedList< Pointer< DestinationInfo > >** (p. 1654), **decaf::util::LinkedList< PrimitiveValueNode >** (p. 1654), **decaf::util::LinkedList< Pointer< Command > >** (p. 1654), **decaf::util::LinkedList< Pointer< BackupTransport > >** (p. 1654), **decaf::util::LinkedList< cms::MessageProducer * >** (p. 1654), **decaf::util::LinkedList< cms::Destination * >** (p. 1654), **decaf::util::LinkedList< cms::Session * >** (p. 1654), **decaf::util::LinkedList< cms::Connection * >** (p. 1654), and **decaf::util::concurrent::Copy-OnWriteArraySet< E >** (p. 1059).

```
6.2.3.14 template<typename E> virtual bool decaf::util::AbstractCollection<
        E >::removeAll( const Collection< E > & collection ) [inline,
        virtual]
```

Removes all this collection's elements that are also contained in the specified collection (optional operation).

After this call returns, this collection will contain no elements in common with the specified collection.

Parameters

<i>collection</i>	The Collection (p. 851) whose elements are to be removed from this one.
-------------------	--

Returns

true if the collection changed as a result of this call.

Exceptions

<i>UnsupportedOperationException</i>	if this is an unmodifiable collection.
<i>NullPointerException</i>	if the Collection (p. 851) is a container of pointers and does not allow NULL values.

This implementation iterates over this collection, checking each element returned by the iterator in turn to see if it's contained in the specified collection. If it's so contained, it's removed from this collection with the iterator's remove method.

Note that this implementation will throw an `UnsupportedOperationException` if the iterator returned by the iterator method does not implement the remove method and this collection contains one or more elements in common with the specified collection.

Implements **decaf::util::Collection**< **E** > (p. 862).

Reimplemented in **decaf::util::concurrent::CopyOnWriteArraySet**< **E** > (p. 1059), **decaf::util::AbstractSet**< **E** > (p. 153), **decaf::util::AbstractSet**< **Pointer**< **Synchronization** > > (p. 153), and **decaf::util::AbstractSet**< **Resource** * > (p. 153).

6.2.3.15 `template<typename E> virtual bool decaf::util::AbstractCollection< E >::retainAll(const Collection< E > & collection) [inline, virtual]`

Retains only the elements in this collection that are contained in the specified collection (optional operation).

In other words, removes from this collection all of its elements that are not contained in the specified collection.

Parameters

<i>collection</i>	The Collection (p. 851) whose elements are to be retained.
-------------------	---

Returns

true if the collection changed as a result of this call.

Exceptions

<i>UnsupportedOperationException</i>	if this is an unmodifiable collection.
--------------------------------------	--

<i>NullPointerException</i>	if the Collection (p. 851) is a container of pointers and does not allow NULL values.
-----------------------------	--

This implementation iterates over this collection, checking each element returned by the iterator in turn to see if it's contained in the specified collection. If it's not so contained, it's removed from this collection with the iterator's remove method.

Note that this implementation will throw an `UnsupportedOperationException` if the iterator returned by the iterator method does not implement the remove method and this collection contains one or more elements not present in the specified collection.

Implements **decaf::util::Collection< E >** (p. 863).

Reimplemented in **decaf::util::concurrent::CopyOnWriteArraySet< E >** (p. 1061).

6.2.3.16 `template<typename E> virtual std::vector<E> decaf::util::-`
AbstractCollection< E >::toArray () const `[inline,`
`virtual]`

Answers an STL vector containing copies of all elements contained in this **Collection** (p. 851).

All the elements in the array will not be referenced by the collection. The elements in the returned array will be sorted to the same order as those returned by the iterator of this collection itself if the collection guarantees the order.

Returns

an vector of copies of all the elements from this **Collection** (p. 851)

Implements **decaf::util::Collection< E >** (p. 865).

Reimplemented in **decaf::util::concurrent::LinkedBlockingQueue< E >** (p. 1632), **decaf::util::ArrayList< E >** (p. 457), **decaf::util::LinkedList< E >** (p. 1658), **decaf::util::LinkedList< Pointer< Transport > >** (p. 1658), **decaf::util::LinkedList< cms::MessageConsumer * >** (p. 1658), **decaf::util::LinkedList< CompositeTask * >** (p. 1658), **decaf::util::LinkedList< URI >** (p. 1658), **decaf::util::LinkedList< Pointer< MessageDispatch > >** (p. 1658), **decaf::util::LinkedList< Pointer< DestinationInfo > >** (p. 1658), **decaf::util::LinkedList< PrimitiveValueNode >** (p. 1658), **decaf::util::LinkedList< Pointer< Command > >** (p. 1658), **decaf::util::LinkedList< Pointer< BackupTransport > >** (p. 1658), **decaf::util::LinkedList< cms::MessageProducer * >** (p. 1658), **decaf::util::LinkedList< cms::Destination * >** (p. 1658), **decaf::util::LinkedList< cms::Session * >** (p. 1658), **decaf::util::LinkedList< cms::Connection * >** (p. 1658), **decaf::util::concurrent::SynchronousQueue< E >** (p. 2665), and **decaf::util::concurrent::CopyOnWriteArraySet< E >** (p. 1062).

6.2.3.17 `template<typename E> virtual bool decaf::util::AbstractCollection< E >::tryLock() [inline, virtual]`

Attempts to Lock the object, if the lock is already held by another thread than this method returns false.

Returns

true if the lock was acquired, false if it is already held by another thread.

Exceptions

<i>RuntimeException</i>	if an error occurs while locking the object.
-------------------------	--

Implements **decaf::util::concurrent::Synchronizable** (p. 2643).

6.2.3.18 `template<typename E> virtual void decaf::util::AbstractCollection< E >::unlock() [inline, virtual]`

Unlocks the object.

Exceptions

<i>RuntimeException</i>	if an error occurs while unlocking the object.
-------------------------	--

Implements **decaf::util::concurrent::Synchronizable** (p. 2645).

6.2.3.19 `template<typename E> virtual void decaf::util::AbstractCollection< E >::wait() [inline, virtual]`

Waits on a signal from this object, which is generated by a call to Notify.

Must have this object locked before calling.

Exceptions

<i>RuntimeException</i>	if an error occurs while waiting on the object.
<i>InterruptedException</i>	if the wait is interrupted before it completes.
<i>IllegalMonitorStateException</i>	- if the current thread is not the owner of the the Synchronizable Object.

Implements **decaf::util::concurrent::Synchronizable** (p. 2646).

6.2.3.20 `template<typename E> virtual void decaf::util::AbstractCollection< E >::wait(long long millisecs) [inline, virtual]`

Waits on a signal from this object, which is generated by a call to Notify.

Must have this object locked before calling. This wait will timeout after the specified time interval.

Parameters

<i>milliseconds</i>	the time in milliseconds to wait, or <code>WAIT_INFINITE</code>
---------------------	---

Exceptions

<i>RuntimeException</i>	if an error occurs while waiting on the object.
<i>InterruptedException</i>	if the wait is interrupted before it completes.
<i>IllegalMonitorStateException</i>	- if the current thread is not the owner of the the Synchronizable Object.

Implements `decaf::util::concurrent::Synchronizable` (p. 2647).

6.2.3.21 `template<typename E> virtual void decaf::util::AbstractCollection< E >::wait
(long long milliseconds, int nanos) [inline, virtual]`

Waits on a signal from this object, which is generated by a call to `Notify`.

Must have this object locked before calling. This wait will timeout after the specified time interval. This method is similar to the one argument wait function except that it add a finer grained control over the amount of time that it waits by adding in the additional nanosecond argument.

NOTE: The ability to wait accurately at a nanosecond scale depends on the platform and OS that the Decaf API is running on, some systems do not provide an accurate enough clock to provide this level of granularity.

Parameters

<i>milliseconds</i>	the time in milliseconds to wait, or <code>WAIT_INFINITE</code>
<i>nanos</i>	additional time in nanoseconds with a range of 0-999999

Exceptions

<i>IllegalArgument-Exception</i>	if an error occurs or the nanos argument is not in the range of [0-999999]
<i>RuntimeException</i>	if an error occurs while waiting on the object.
<i>InterruptedException</i>	if the wait is interrupted before it completes.
<i>IllegalMonitorStateException</i>	- if the current thread is not the owner of the the Synchronizable Object.

Implements `decaf::util::concurrent::Synchronizable` (p. 2648).

6.2.4 Field Documentation

6.2.4.1 `template<typename E> util::concurrent::Mutex decaf::util::AbstractCollection< E >::mutex` [mutable, protected]

Referenced by `decaf::util::AbstractCollection< cms::Connection * >::lock()`, `decaf::util::AbstractCollection< cms::Connection * >::notify()`, `decaf::util::AbstractCollection< cms::Connection * >::notifyAll()`, `decaf::util::AbstractCollection< cms::Connection * >::tryLock()`, `decaf::util::AbstractCollection< cms::Connection * >::unlock()`, and `decaf::util::AbstractCollection< cms::Connection * >::wait()`.

The documentation for this class was generated from the following file:

- `src/main/decaf/util/AbstractCollection.h`

6.3 decaf::util::concurrent::AbstractExecutorService Class - Reference

Provides a default implementation for the methods of the **ExecutorService** (p. 1302) interface.

```
#include <src/main/decaf/util/concurrent/AbstractExecutorService.h>
```

Inheritance diagram for `decaf::util::concurrent::AbstractExecutorService`:

Public Member Functions

- **AbstractExecutorService** ()
- virtual **~AbstractExecutorService** ()

6.3.1 Detailed Description

Provides a default implementation for the methods of the **ExecutorService** (p. 1302) interface.

Use this class as a starting point for implementations of custom executor service implementations.

Since

1.0

6.3.2 Constructor & Destructor Documentation

6.3.2.1 decaf::util::concurrent::AbstractExecutorService::AbstractExecutorService ()

6.3.2.2 virtual decaf::util::concurrent::AbstractExecutorService::~~AbstractExecutorService () [virtual]

The documentation for this class was generated from the following file:

- src/main/decaf/util/concurrent/**AbstractExecutorService.h**

6.4 decaf::util::AbstractList< E > Class Template Reference

This class provides a skeletal implementation of the **List** (p. 1658) interface to minimize the effort required to implement this interface backed by a "random access" data store (such as an array).

```
#include <src/main/decaf/util/AbstractList.h>
```

Inheritance diagram for decaf::util::AbstractList< E >:

Data Structures

- class **ConstSimpleListIterator**
- class **SimpleListIterator**

Public Member Functions

- **AbstractList** ()
- virtual **~AbstractList** ()
- virtual **Iterator**< E > * **iterator** ()
- virtual **Iterator**< E > * **iterator** () const
- virtual **ListIterator**< E > * **listIterator** ()
- virtual **ListIterator**< E > * **listIterator** () const
- virtual **ListIterator**< E > * **listIterator** (int index)
- virtual **ListIterator**< E > * **listIterator** (int index) const
- virtual void **clear** ()
Removes all of the elements from this collection (optional operation).
- virtual bool **add** (const E &value)
Returns true if this collection changed as a result of the call.
- virtual void **add** (int index **DECAF_UNUSED**, const E &element **DECAF_UNUSED**)
- virtual bool **addAll** (int index, const **Collection**< E > &source)
Inserts all of the elements in the specified collection into this list at the specified position (optional operation).

- virtual E **removeAt** (int index **DECAF_UNUSED**)
Removes the element at the specified position in this list.
- virtual E **set** (int index **DECAF_UNUSED**, const E &element **DECAF_UNUSED**)
- virtual int **indexOf** (const E &value) const
Returns the index of the first occurrence of the specified element in this list, or -1 if this list does not contain the element.
- virtual int **lastIndexOf** (const E &value) const
Returns the index of the last occurrence of the specified element in this list, or -1 if this list does not contain the element.

Protected Member Functions

- void **removeRange** (int start, int end)

Protected Attributes

- int **modCount**

6.4.1 Detailed Description

```
template<typename E>class decaf::util::AbstractList< E >
```

This class provides a skeletal implementation of the **List** (p. 1658) interface to minimize the effort required to implement this interface backed by a "random access" data store (such as an array).

For sequential access data (such as a linked list), **AbstractSequentialList** (p. 143) should be used in preference to this class.

To implement an unmodifiable list, the programmer needs only to extend this class and provide implementations for the **get(int)** and **size()** (p. 864) methods.

To implement a modifiable list, the programmer must additionally override the **set(int, E)** method (which otherwise throws an `UnsupportedOperationException`). If the list is variable-size the programmer must additionally override the **add(int, E)** and **remove(int)** methods.

The programmer should generally provide a void (no argument) and collection constructor, as per the recommendation in the **Collection** (p. 851) interface specification.

Unlike the other abstract collection implementations, the programmer does not have to provide an iterator implementation; the iterator and list iterator are implemented by this class, on top of the "random access" methods: **get(int)**, **set(int, E)**, **add(int, E)** and **remove(int)**.

The documentation for each non-abstract method in this class describes its implementation in detail. Each of these methods may be overridden if the collection being implemented admits a more efficient implementation.

Since

1.0

6.4.2 Constructor & Destructor Documentation

6.4.2.1 `template<typename E> decaf::util::AbstractList< E >::AbstractList ()`
`[inline]`

6.4.2.2 `template<typename E> virtual decaf::util::AbstractList< E >::~~AbstractList ()`
`[inline, virtual]`

6.4.3 Member Function Documentation

6.4.3.1 `template<typename E> virtual bool decaf::util::AbstractList< E >::add (const E & value)`
`[inline, virtual]`

Returns true if this collection changed as a result of the call.

(Returns false if this collection does not permit duplicates and already contains the specified element.)

Collections that support this operation may place limitations on what elements may be added to this collection. In particular, some collections will refuse to add null elements, and others will impose restrictions on the type of elements that may be added. **Collection** (p. 851) classes should clearly specify in their documentation any restrictions on what elements may be added.

If a collection refuses to add a particular element for any reason other than that it already contains the element, it must throw an exception (rather than returning false). This preserves the invariant that a collection always contains the specified element after this call returns.

For non-pointer values, i.e. class instances or string's the object will be copied into the collection, thus the object must support being copied, must not hide the copy constructor and assignment operator.

Parameters

<i>value</i>	The reference to the element to add to this Collection (p. 851).
--------------	---

Returns

true if the element was added to this **Collection** (p. 851).

Exceptions

<i>UnsupportedOperationException</i>	if this is an unmodifiable collection.
<i>NullPointerException</i>	if the Collection (p. 851) is a container of pointers and does not allow NULL values.

<i>IllegalArgumentException</i>	if some property of the element prevents it from being added to this collection
<i>IllegalStateException</i>	if the element cannot be added at this time due to insertion restrictions.

Implements **decaf::util::Collection< E >** (p. 853).

Reimplemented in **decaf::util::StlList< E >** (p. 2542), **decaf::util::ArrayList< E >** (p. 448), **decaf::util::LinkedList< E >** (p. 1639), **decaf::util::LinkedList< Pointer< Transport > >** (p. 1639), **decaf::util::LinkedList< cms::MessageConsumer * >** (p. 1639), **decaf::util::LinkedList< CompositeTask * >** (p. 1639), **decaf::util::LinkedList< URI >** (p. 1639), **decaf::util::LinkedList< Pointer< MessageDispatch > >** (p. 1639), **decaf::util::LinkedList< Pointer< DestinationInfo > >** (p. 1639), **decaf::util::LinkedList< PrimitiveValueNode >** (p. 1639), **decaf::util::LinkedList< Pointer< Command > >** (p. 1639), **decaf::util::LinkedList< Pointer< BackupTransport > >** (p. 1639), **decaf::util::LinkedList< cms::MessageProducer * >** (p. 1639), **decaf::util::LinkedList< cms::Destination * >** (p. 1639), **decaf::util::LinkedList< cms::Session * >** (p. 1639), and **decaf::util::LinkedList< cms::Connection * >** (p. 1639).

Referenced by **decaf::util::AbstractList< cms::Connection * >::add()**, and **decaf::util::AbstractList< cms::Connection * >::addAll()**.

6.4.3.2 `template<typename E> virtual void decaf::util::AbstractList< E >::add
(int index DECAF_UNUSED, const E &element DECAF_UNUSED) [inline,
virtual]`

6.4.3.3 `template<typename E> virtual bool decaf::util::AbstractList< E >::addAll(int
index, const Collection< E > &source) [inline, virtual]`

Inserts all of the elements in the specified collection into this list at the specified position (optional operation).

Shifts the element currently at that position (if any) and any subsequent elements to the right (increases their indices). The new elements will appear in this list in the order that they are returned by the specified collection's iterator. The behavior of this operation is undefined if the specified collection is modified while the operation is in progress. (Note that this will occur if the specified collection is this list, and it's nonempty.)

Parameters

<i>index</i>	The index at which to insert the first element from the specified collection
<i>source</i>	The Collection (p. 851) containing elements to be added to this list

Returns

true if this list changed as a result of the call

Exceptions

<i>IndexOutOfBoundsException</i>	if the index given is less than zero or greater than the List (p. 1658) size.
<i>UnsupportedOperationException</i>	if this is an unmodifiable collection.
<i>NullPointerException</i>	if the Collection (p. 851) is a container of pointers and does not allow NULL values.
<i>IllegalArgumentException</i>	if some property of the element prevents it from being added to this collection
<i>IllegalStateException</i>	if the element cannot be added at this time due to insertion restrictions.

Implements **decaf::util::List< E >** (p. 1661).

Reimplemented in **decaf::util::StlList< E >** (p. 2544), **decaf::util::ArrayList< E >** (p. 451), **decaf::util::LinkedList< E >** (p. 1642), **decaf::util::LinkedList< Pointer< Transport > >** (p. 1642), **decaf::util::LinkedList< cms::MessageConsumer * >** (p. 1642), **decaf::util::LinkedList< CompositeTask * >** (p. 1642), **decaf::util::LinkedList< URI >** (p. 1642), **decaf::util::LinkedList< Pointer< MessageDispatch > >** (p. 1642), **decaf::util::LinkedList< Pointer< DestinationInfo > >** (p. 1642), **decaf::util::LinkedList< PrimitiveValueNode >** (p. 1642), **decaf::util::LinkedList< Pointer< Command > >** (p. 1642), **decaf::util::LinkedList< Pointer< BackupTransport > >** (p. 1642), **decaf::util::LinkedList< cms::MessageProducer * >** (p. 1642), **decaf::util::LinkedList< cms::Destination * >** (p. 1642), **decaf::util::LinkedList< cms::Session * >** (p. 1642), **decaf::util::LinkedList< cms::Connection * >** (p. 1642), **decaf::util::AbstractSequentialList< E >** (p. 147), **decaf::util::AbstractSequentialList< Pointer< Transport > >** (p. 147), **decaf::util::AbstractSequentialList< cms::MessageConsumer * >** (p. 147), **decaf::util::AbstractSequentialList< CompositeTask * >** (p. 147), **decaf::util::AbstractSequentialList< URI >** (p. 147), **decaf::util::AbstractSequentialList< Pointer< MessageDispatch > >** (p. 147), **decaf::util::AbstractSequentialList< Pointer< DestinationInfo > >** (p. 147), **decaf::util::AbstractSequentialList< PrimitiveValueNode >** (p. 147), **decaf::util::AbstractSequentialList< Pointer< Command > >** (p. 147), **decaf::util::AbstractSequentialList< Pointer< BackupTransport > >** (p. 147), **decaf::util::AbstractSequentialList< cms::MessageProducer * >** (p. 147), **decaf::util::AbstractSequentialList< cms::Destination * >** (p. 147), **decaf::util::AbstractSequentialList< cms::Session * >** (p. 147), and **decaf::util::AbstractSequentialList< cms::Connection * >** (p. 147).

6.4.3.4 `template<typename E> virtual void decaf::util::AbstractList< E >::clear ()`
`[inline, virtual]`

Removes all of the elements from this collection (optional operation).

The collection will be empty after this method returns.

This implementation iterates over this collection, removing each element using the **Iterator.remove** (p. 1560) operation. Most implementations will probably choose to override this method for efficiency.

Note that this implementation will throw an `UnsupportedOperationException` if the iterator returned by this collection's iterator method does not implement the remove method and this collection is non-empty.

Exceptions

<i>UnsupportedOperationException</i>	if the clear operation is not supported by this collection
--------------------------------------	--

Reimplemented from `decaf::util::AbstractCollection< E >` (p. 111).

Reimplemented in `decaf::util::StlList< E >` (p. 2545), `decaf::util::LinkedList< E >` (p. 1644), `decaf::util::LinkedList< Pointer< Transport > >` (p. 1644), `decaf::util::LinkedList< cms::MessageConsumer * >` (p. 1644), `decaf::util::LinkedList< CompositeTask * >` (p. 1644), `decaf::util::LinkedList< URI >` (p. 1644), `decaf::util::LinkedList< Pointer< MessageDispatch > >` (p. 1644), `decaf::util::LinkedList< Pointer< DestinationInfo > >` (p. 1644), `decaf::util::LinkedList< PrimitiveValueNode >` (p. 1644), `decaf::util::LinkedList< Pointer< Command > >` (p. 1644), `decaf::util::LinkedList< Pointer< BackupTransport > >` (p. 1644), `decaf::util::LinkedList< cms::MessageProducer * >` (p. 1644), `decaf::util::LinkedList< cms::Destination * >` (p. 1644), `decaf::util::LinkedList< cms::Session * >` (p. 1644), `decaf::util::LinkedList< cms::Connection * >` (p. 1644), and `decaf::util::ArrayList< E >` (p. 452).

6.4.3.5 `template<typename E> virtual int decaf::util::AbstractList< E >::indexOf (const E & value) const` [inline, virtual]

Returns the index of the first occurrence of the specified element in this list, or -1 if this list does not contain the element.

More formally, returns the lowest index *i* such that `get(i) == value`, or -1 if there is no such index.

Parameters

<i>value</i>	The element to search for in this List (p. 1658).
--------------	--

Returns

the index of the first occurrence of the specified element in this list,

Exceptions

<i>NullPointerException</i>	if the Collection (p. 851) is a container of pointers and does not allow NULL values.
-----------------------------	--

Implements `decaf::util::List< E >` (p. 1663).

Reimplemented in `decaf::util::StlList< E >` (p. 2547), `decaf::util::ArrayList< E >` (p. 453), `decaf::util::LinkedList< E >` (p. 1647), `decaf::util::LinkedList< Pointer<`

Transport > > (p. 1647), **decaf::util::LinkedList**< **cms::MessageConsumer** * > (p. 1647), **decaf::util::LinkedList**< **CompositeTask** * > (p. 1647), **decaf::util::LinkedList**< **URI** > (p. 1647), **decaf::util::LinkedList**< **Pointer**< **MessageDispatch** > > (p. 1647), **decaf::util::LinkedList**< **Pointer**< **DestinationInfo** > > (p. 1647), **decaf::util::LinkedList**< **PrimitiveValueNode** > (p. 1647), **decaf::util::LinkedList**< **Pointer**< **Command** > > (p. 1647), **decaf::util::LinkedList**< **Pointer**< **BackupTransport** > > (p. 1647), **decaf::util::LinkedList**< **cms::MessageProducer** * > (p. 1647), **decaf::util::LinkedList**< **cms::Destination** * > (p. 1647), **decaf::util::LinkedList**< **cms::Session** * > (p. 1647), and **decaf::util::LinkedList**< **cms::Connection** * > (p. 1647).

6.4.3.6 `template<typename E> virtual Iterator<E>* decaf::util::AbstractList< E >::iterator() [inline, virtual]`

Returns

an iterator over a set of elements of type T.

Implements **decaf::lang::Iterable**< **E** > (p. 1557).

Reimplemented in **decaf::util::StlList**< **E** > (p. 2548), **decaf::util::AbstractSequentialList**< **E** > (p. 148), **decaf::util::AbstractSequentialList**< **Pointer**< **Transport** > > (p. 148), **decaf::util::AbstractSequentialList**< **cms::MessageConsumer** * > (p. 148), **decaf::util::AbstractSequentialList**< **CompositeTask** * > (p. 148), **decaf::util::AbstractSequentialList**< **URI** > (p. 148), **decaf::util::AbstractSequentialList**< **Pointer**< **MessageDispatch** > > (p. 148), **decaf::util::AbstractSequentialList**< **Pointer**< **DestinationInfo** > > (p. 148), **decaf::util::AbstractSequentialList**< **PrimitiveValueNode** > (p. 148), **decaf::util::AbstractSequentialList**< **Pointer**< **Command** > > (p. 148), **decaf::util::AbstractSequentialList**< **Pointer**< **BackupTransport** > > (p. 148), **decaf::util::AbstractSequentialList**< **cms::MessageProducer** * > (p. 148), **decaf::util::AbstractSequentialList**< **cms::Destination** * > (p. 148), **decaf::util::AbstractSequentialList**< **cms::Session** * > (p. 148), and **decaf::util::AbstractSequentialList**< **cms::Connection** * > (p. 148).

Referenced by **decaf::util::ArrayList**< **E** >::ArrayList().

6.4.3.7 `template<typename E> virtual Iterator<E>* decaf::util::AbstractList< E >::iterator() const [inline, virtual]`

Implements **decaf::lang::Iterable**< **E** > (p. 1558).

Reimplemented in **decaf::util::StlList**< **E** > (p. 2548), **decaf::util::AbstractSequentialList**< **E** > (p. 149), **decaf::util::AbstractSequentialList**< **Pointer**< **Transport** > > (p. 149), **decaf::util::AbstractSequentialList**< **cms::MessageConsumer** * > (p. 149), **decaf::util::AbstractSequentialList**< **CompositeTask** * > (p. 149), **decaf::util::AbstractSequentialList**< **URI** > (p. 149), **decaf::util::AbstractSequentialList**< **Pointer**< **MessageDispatch** > > (p. 149), **decaf::util::AbstractSequentialList**< **Pointer**< **DestinationInfo** > > (p. 149), **decaf::util::AbstractSequentialList**< **PrimitiveValueNode** > (p. 149), **decaf::util::AbstractSequentialList**< **Pointer**< **Command** > > (p. 149), **decaf::util::AbstractSequentialList**<

Pointer< **BackupTransport** > > (p. 149), **decaf::util::AbstractSequentialList**< **cms::MessageProducer** * > (p. 149), **decaf::util::AbstractSequentialList**< **cms::Destination** * > (p. 149), **decaf::util::AbstractSequentialList**< **cms::Session** * > (p. 149), and **decaf::util::AbstractSequentialList**< **cms::Connection** * > (p. 149).

6.4.3.8 `template<typename E> virtual int decaf::util::AbstractList< E >::lastIndexOf (const E & value) const [inline, virtual]`

Returns the index of the last occurrence of the specified element in this list, or -1 if this list does not contain the element.

More formally, returns the highest index *i* such that `get(i) == value` or -1 if there is no such index.

Parameters

<i>value</i>	The element to search for in this List (p. 1658).
--------------	--

Returns

the index of the last occurrence of the specified element in this list.

Exceptions

<i>NullPointerException</i>	if the Collection (p. 851) is a container of pointers and does not allow NULL values.
-----------------------------	--

Implements **decaf::util::List**< **E** > (p. 1664).

Reimplemented in **decaf::util::StlList**< **E** > (p. 2548), **decaf::util::ArrayList**< **E** > (p. 454), **decaf::util::LinkedList**< **E** > (p. 1648), **decaf::util::LinkedList**< **Pointer**< **Transport** > > (p. 1648), **decaf::util::LinkedList**< **cms::MessageConsumer** * > (p. 1648), **decaf::util::LinkedList**< **CompositeTask** * > (p. 1648), **decaf::util::LinkedList**< **URI** > (p. 1648), **decaf::util::LinkedList**< **Pointer**< **MessageDispatch** > > (p. 1648), **decaf::util::LinkedList**< **Pointer**< **DestinationInfo** > > (p. 1648), **decaf::util::LinkedList**< **PrimitiveValueNode** > (p. 1648), **decaf::util::LinkedList**< **Pointer**< **Command** > > (p. 1648), **decaf::util::LinkedList**< **Pointer**< **BackupTransport** > > (p. 1648), **decaf::util::LinkedList**< **cms::MessageProducer** * > (p. 1648), **decaf::util::LinkedList**< **cms::Destination** * > (p. 1648), **decaf::util::LinkedList**< **cms::Session** * > (p. 1648), and **decaf::util::LinkedList**< **cms::Connection** * > (p. 1648).

6.4.3.9 `template<typename E> virtual ListIterator<E>* decaf::util::AbstractList< E >::listIterator () [inline, virtual]`

Returns

a list iterator over the elements in this list (in proper sequence).

Implements **decaf::util::List< E >** (p. 1665).

Reimplemented in **decaf::util::StlList< E >** (p. 2549), **decaf::util::AbstractSequentialList< E >** (p. 149), **decaf::util::AbstractSequentialList< Pointer< Transport > >** (p. 149), **decaf::util::AbstractSequentialList< cms::MessageConsumer * >** (p. 149), **decaf::util::AbstractSequentialList< CompositeTask * >** (p. 149), **decaf::util::AbstractSequentialList< URI >** (p. 149), **decaf::util::AbstractSequentialList< Pointer< MessageDispatch > >** (p. 149), **decaf::util::AbstractSequentialList< Pointer< DestinationInfo > >** (p. 149), **decaf::util::AbstractSequentialList< PrimitiveValueNode >** (p. 149), **decaf::util::AbstractSequentialList< Pointer< Command > >** (p. 149), **decaf::util::AbstractSequentialList< Pointer< BackupTransport > >** (p. 149), **decaf::util::AbstractSequentialList< cms::MessageProducer * >** (p. 149), **decaf::util::AbstractSequentialList< cms::Destination * >** (p. 149), **decaf::util::AbstractSequentialList< cms::Session * >** (p. 149), and **decaf::util::AbstractSequentialList< cms::Connection * >** (p. 149).

Referenced by **decaf::util::AbstractList< cms::Connection * >::indexOf()**, **decaf::util::AbstractList< cms::Connection * >::lastIndexOf()**, and **decaf::util::AbstractList< cms::Connection * >::removeRange()**.

6.4.3.10 `template<typename E> virtual ListIterator<E>* decaf::util::AbstractList< E >::listIterator() const [inline, virtual]`

Implements **decaf::util::List< E >** (p. 1666).

Reimplemented in **decaf::util::StlList< E >** (p. 2549), **decaf::util::AbstractSequentialList< E >** (p. 149), **decaf::util::AbstractSequentialList< Pointer< Transport > >** (p. 149), **decaf::util::AbstractSequentialList< cms::MessageConsumer * >** (p. 149), **decaf::util::AbstractSequentialList< CompositeTask * >** (p. 149), **decaf::util::AbstractSequentialList< URI >** (p. 149), **decaf::util::AbstractSequentialList< Pointer< MessageDispatch > >** (p. 149), **decaf::util::AbstractSequentialList< Pointer< DestinationInfo > >** (p. 149), **decaf::util::AbstractSequentialList< PrimitiveValueNode >** (p. 149), **decaf::util::AbstractSequentialList< Pointer< Command > >** (p. 149), **decaf::util::AbstractSequentialList< Pointer< BackupTransport > >** (p. 149), **decaf::util::AbstractSequentialList< cms::MessageProducer * >** (p. 149), **decaf::util::AbstractSequentialList< cms::Destination * >** (p. 149), **decaf::util::AbstractSequentialList< cms::Session * >** (p. 149), and **decaf::util::AbstractSequentialList< cms::Connection * >** (p. 149).

6.4.3.11 `template<typename E> virtual ListIterator<E>* decaf::util::AbstractList< E >::listIterator(int index) [inline, virtual]`

Parameters

<i>index</i>	index of first element to be returned from the list iterator (by a call to the next method).
--------------	--

Returns

a list iterator of the elements in this list (in proper sequence), starting at the specified position in this list. The specified index indicates the first element that would be returned by an initial call to next. An initial call to previous would return the element with the specified index minus one.

Exceptions

<i>IndexOutOfBoundsException</i>	if the index is out of range (index < 0 index > size()) (p. 864)
----------------------------------	---

Implements **decaf::util::List< E >** (p. 1666).

Reimplemented in **decaf::util::LinkedList< E >** (p. 1649), **decaf::util::LinkedList< Pointer< Transport > >** (p. 1649), **decaf::util::LinkedList< cms::MessageConsumer * >** (p. 1649), **decaf::util::LinkedList< CompositeTask * >** (p. 1649), **decaf::util::LinkedList< URI >** (p. 1649), **decaf::util::LinkedList< Pointer< MessageDispatch > >** (p. 1649), **decaf::util::LinkedList< Pointer< DestinationInfo > >** (p. 1649), **decaf::util::LinkedList< PrimitiveValueNode >** (p. 1649), **decaf::util::LinkedList< Pointer< Command > >** (p. 1649), **decaf::util::LinkedList< Pointer< BackupTransport > >** (p. 1649), **decaf::util::LinkedList< cms::MessageProducer * >** (p. 1649), **decaf::util::LinkedList< cms::Destination * >** (p. 1649), **decaf::util::LinkedList< cms::Session * >** (p. 1649), **decaf::util::LinkedList< cms::Connection * >** (p. 1649), **decaf::util::StlList< E >** (p. 2549), **decaf::util::AbstractSequentialList< E >** (p. 149), **decaf::util::AbstractSequentialList< Pointer< Transport > >** (p. 149), **decaf::util::AbstractSequentialList< cms::MessageConsumer * >** (p. 149), **decaf::util::AbstractSequentialList< CompositeTask * >** (p. 149), **decaf::util::AbstractSequentialList< URI >** (p. 149), **decaf::util::AbstractSequentialList< Pointer< MessageDispatch > >** (p. 149), **decaf::util::AbstractSequentialList< Pointer< DestinationInfo > >** (p. 149), **decaf::util::AbstractSequentialList< PrimitiveValueNode >** (p. 149), **decaf::util::AbstractSequentialList< Pointer< Command > >** (p. 149), **decaf::util::AbstractSequentialList< Pointer< BackupTransport > >** (p. 149), **decaf::util::AbstractSequentialList< cms::MessageProducer * >** (p. 149), **decaf::util::AbstractSequentialList< cms::Destination * >** (p. 149), **decaf::util::AbstractSequentialList< cms::Session * >** (p. 149), and **decaf::util::AbstractSequentialList< cms::Connection * >** (p. 149).

```
6.4.3.12 template<typename E> virtual ListIterator<E>* decaf::util::AbstractList< E
>::listIterator ( int index ) const [inline, virtual]
```

Implements **decaf::util::List< E >** (p. 1668).

Reimplemented in **decaf::util::LinkedList< E >** (p. 1649), **decaf::util::LinkedList< Pointer< Transport > >** (p. 1649), **decaf::util::LinkedList< cms::MessageConsumer * >** (p. 1649), **decaf::util::LinkedList< CompositeTask * >** (p. 1649), **decaf::util::LinkedList< URI >** (p. 1649), **decaf::util::LinkedList< Pointer< MessageDispatch > >** (p. 1649), **decaf::util::LinkedList< Pointer< DestinationInfo > >** (p. 1649), **decaf::util::LinkedList< PrimitiveValueNode >** (p. 1649),

decaf::util::LinkedList< Pointer< Command > > (p. 1649), decaf::util::LinkedList< Pointer< BackupTransport > > (p. 1649), decaf::util::LinkedList< cms::MessageProducer * > (p. 1649), decaf::util::LinkedList< cms::Destination * > (p. 1649), decaf::util::LinkedList< cms::Session * > (p. 1649), decaf::util::LinkedList< cms::Connection * > (p. 1649), decaf::util::StlList< E > (p. 2550), decaf::util::AbstractSequentialList< E > (p. 150), decaf::util::AbstractSequentialList< Pointer< Transport > > (p. 150), decaf::util::AbstractSequentialList< cms::MessageConsumer * > (p. 150), decaf::util::AbstractSequentialList< CompositeTask * > (p. 150), decaf::util::AbstractSequentialList< URI > (p. 150), decaf::util::AbstractSequentialList< Pointer< MessageDispatch > > (p. 150), decaf::util::AbstractSequentialList< Pointer< DestinationInfo > > (p. 150), decaf::util::AbstractSequentialList< PrimitiveValueNode > (p. 150), decaf::util::AbstractSequentialList< Pointer< Command > > (p. 150), decaf::util::AbstractSequentialList< Pointer< BackupTransport > > (p. 150), decaf::util::AbstractSequentialList< cms::MessageProducer * > (p. 150), decaf::util::AbstractSequentialList< cms::Destination * > (p. 150), decaf::util::AbstractSequentialList< cms::Session * > (p. 150), and decaf::util::AbstractSequentialList< cms::Connection * > (p. 150).

6.4.3.13 `template<typename E> virtual E decaf::util::AbstractList< E >::removeAt (int index index) [inline, virtual]`

Removes the element at the specified position in this list.

Shifts any subsequent elements to the left (subtracts one from their indices). Returns the element that was removed from the list.

Parameters

<i>index</i>	- the index of the element to be removed.
--------------	---

Returns

the element previously at the specified position.

Exceptions

<i>IndexOutOfBoundsException</i>	if the index given is less than zero or greater than the List (p. 1658) size.
<i>UnsupportedOperationException</i>	if this is an unmodifiable collection.

Implements **decaf::util::List< E >** (p. 1668).

Reimplemented in **decaf::util::StlList< E >** (p. 2550), **decaf::util::ArrayList< E >** (p. 456), **decaf::util::AbstractSequentialList< E >** (p. 150), **decaf::util::AbstractSequentialList< Pointer< Transport > >** (p. 150), **decaf::util::AbstractSequentialList< cms::MessageConsumer * >** (p. 150), **decaf::util::AbstractSequentialList< CompositeTask * >** (p. 150), **decaf::util::AbstractSequentialList< URI >** (p. 150), **decaf::util::AbstractSequentialList< Pointer< MessageDispatch > >**

> (p. 150), `decaf::util::AbstractSequentialList< Pointer< DestinationInfo > >` (p. 150), `decaf::util::AbstractSequentialList< PrimitiveValueNode >` (p. 150), `decaf::util::AbstractSequentialList< Pointer< Command > >` (p. 150), `decaf::util::AbstractSequentialList< Pointer< BackupTransport > >` (p. 150), `decaf::util::AbstractSequentialList< cms::MessageProducer * >` (p. 150), `decaf::util::AbstractSequentialList< cms::Destination * >` (p. 150), `decaf::util::AbstractSequentialList< cms::Session * >` (p. 150), and `decaf::util::AbstractSequentialList< cms::Connection * >` (p. 150).

6.4.3.14 `template<typename E> void decaf::util::AbstractList< E >::removeRange (int start, int end)` [`inline`, `protected`]

Referenced by `decaf::util::AbstractList< cms::Connection * >::clear()`.

6.4.3.15 `template<typename E> virtual E decaf::util::AbstractList< E >::set (int index DECAF_UNUSED, const E &element DECAF_UNUSED)` [`inline`, `virtual`]

6.4.4 Field Documentation

6.4.4.1 `template<typename E> int decaf::util::AbstractList< E >::modCount` [`protected`]

The documentation for this class was generated from the following file:

- `src/main/decaf/util/AbstractList.h`

6.5 `decaf::util::AbstractMap< K, V, COMPARATOR >` Class - Template Reference

This class provides a skeletal implementation of the **Map** (p. 1768) interface, to minimize the effort required to implement this interface.

```
#include <src/main/decaf/util/AbstractMap.h>
```

Inheritance diagram for `decaf::util::AbstractMap< K, V, COMPARATOR >`:

Public Member Functions

- `virtual ~AbstractMap ()`

6.5.1 Detailed Description

```
template<typename K, typename V, typename COMPARTOR>class decaf::util::AbstractMap<
K, V, COMPARTOR >
```

This class provides a skeletal implementation of the **Map** (p. 1768) interface, to minimize the effort required to implement this interface.

To implement an unmodifiable map, the programmer needs only to extend this class and provide an implementation for the `entrySet` method, which returns a set-view of the map's mappings. Typically, the returned set will, in turn, be implemented atop **AbstractSet** (p. 152). This set should not support the `add` or `remove` methods, and its iterator should not support the `remove` method.

To implement a modifiable map, the programmer must additionally override this class's `put` method (which otherwise throws an `UnsupportedOperationException`), and the iterator returned by `entrySet().iterator()` must additionally implement its `remove` method.

The programmer should generally provide a void (no argument) and map constructor, as per the recommendation in the **Map** (p. 1768) interface specification.

The documentation for each non-abstract method in this class describes its implementation in detail. Each of these methods may be overridden if the map being implemented admits a more efficient implementation.

Since

1.0

6.5.2 Constructor & Destructor Documentation

```
6.5.2.1 template<typename K , typename V , typename COMPARTOR > virtual
decaf::util::AbstractMap< K, V, COMPARTOR >::~~AbstractMap ( )
[inline, virtual]
```

The documentation for this class was generated from the following file:

- `src/main/decaf/util/AbstractMap.h`

6.6 decaf::util::concurrent::locks::AbstractOwnableSynchronizer - Class Reference

Base class for locks that provide the notion of Ownership, the types of locks that are implemented using this base class would be owned by one specific Thread at any given time.

```
#include <src/main/decaf/util/concurrent/locks/Abstract-
OwnableSynchronizer.h>
```

Public Member Functions

- virtual **~AbstractOwnableSynchronizer** ()

Protected Member Functions

- **AbstractOwnableSynchronizer** ()
- **decaf::lang::Thread * getExclusiveOwnerThread** () const
Gets the Thread that was last set using the setExclusiveOwnerThread method, or NULL if no Thread has been made the exclusive owner.
- void **setExclusiveOwnerThread** (decaf::lang::Thread *thread)
Sets the Thread that has exclusive ownership of this Synchronizer, can be NULL to indicate that no Thread now owns this Synchronizer.

6.6.1 Detailed Description

Base class for locks that provide the notion of Ownership, the types of locks that are implemented using this base class would be owned by one specific Thread at any given time.

Since

1.0

6.6.2 Constructor & Destructor Documentation

6.6.2.1 **virtual decaf::util::concurrent::locks::AbstractOwnableSynchronizer::~AbstractOwnableSynchronizer** ()
[virtual]

6.6.2.2 **decaf::util::concurrent::locks::AbstractOwnableSynchronizer::AbstractOwnableSynchronizer** ()
[protected]

6.6.3 Member Function Documentation

6.6.3.1 **decaf::lang::Thread* decaf::util::concurrent::locks::AbstractOwnableSynchronizer::getExclusiveOwnerThread** () const
[protected]

Gets the Thread that was last set using the setExclusiveOwnerThread method, or NULL if no Thread has been made the exclusive owner.

Returns

pointer to the owner Thread or NULL if not set.

```
6.6.3.2 void decaf::util::concurrent::locks::AbstractOwnableSynchronizer-
::setExclusiveOwnerThread ( decaf::lang::Thread * thread )
[protected]
```

Sets the Thread that has exclusive ownership of this Synchronizer, can be NULL to indicate that no Thread now owns this Synchronizer.

Parameters

<i>thread</i>	The Thread that now has ownership, or NULL if ownership is released.
---------------	--

The documentation for this class was generated from the following file:

- src/main/decaf/util/concurrent/locks/**AbstractOwnableSynchronizer.h**

6.7 decaf::util::AbstractQueue< E > Class Template Reference

This class provides skeletal implementations of some **Queue** (p. 2222) operations.

```
#include <src/main/decaf/util/AbstractQueue.h>
```

Inheritance diagram for decaf::util::AbstractQueue< E >:

Public Member Functions

- **AbstractQueue** ()
- virtual ~**AbstractQueue** ()
- virtual bool **add** (const E &value)

Returns true if this collection changed as a result of the call.

(Returns false if this collection does not permit duplicates and already contains the specified element.)

Collections that support this operation may place limitations on what elements may be added to this collection. In particular, some collections will refuse to add null elements, and others will impose restrictions on the type of elements that may be added.

Collection (p. 851) classes should clearly specify in their documentation any restrictions on what elements may be added.

If a collection refuses to add a particular element for any reason other than that it already contains the element, it must throw an exception (rather than returning false). This preserves the invariant that a collection always contains the specified element after this call returns.

For non-pointer values, i.e. class instances or string's the object will be copied into the collection, thus the object must support being copied, must not hide the copy constructor and assignment operator.

Parameters

value	The reference to the element to add to this Collection (p. 851).
-------	---

Returns

*true if the element was added to this **Collection** (p. 851).*

Exceptions

UnsupportedOperationException	<i>if this is an unmodifiable collection.</i>
NullPointerException	<i>if the Collection (p. 851) is a container of pointers and does not allow NULL values.</i>
IllegalArgumentException	<i>if some property of the element prevents it from being added to this collection</i>
IllegalStateException	<i>if the element cannot be added at this time due to insertion restrictions.</i>

- virtual bool **addAll** (const **Collection**< E > &collection)

Adds all of the elements in the specified collection to this collection.

The behavior of this operation is undefined if the specified collection is modified while the operation is in progress. (This implies that the behavior of this call is undefined if the specified collection is this collection, and this collection is nonempty.)

Parameters

collection	<i>The Collection (p. 851) whose elements are added to this one.</i>
------------	---

Returns

true if this collection changed as a result of the call

Exceptions

UnsupportedOperationException	<i>if this is an unmodifiable collection.</i>
NullPointerException	<i>if the Collection (p. 851) is a container of pointers and does not allow NULL values.</i>
IllegalArgumentException	<i>if some property of an element prevents it from being added to this collection</i>
IllegalStateException	<i>if an element cannot be added at this time due to insertion restrictions.</i>

This implementation iterates over the specified collection, and adds each object returned by the iterator to this collection, in turn.

Note that this implementation will throw an UnsupportedOperationException unless add is overridden (assuming the specified collection is non-empty).

- virtual E **remove** ()

Gets and removes the element in the head of the queue.

*Throws a **NoSuchElementException** (p. 1984) if there is no element in the queue.*

Returns

the element in the head of the queue.

Exceptions

NoSuchElementException (p. 1984)	<i>if there is no element in the queue.</i>
--	---

- virtual E **element** () const

Gets but not removes the element in the head of the queue.

*Throws a **NoSuchElementException** (p. 1984) if there is no element in the queue.*

Returns

the element in the head of the queue.

Exceptions

NoSuchElementException (p. 1984)	<i>if there is no element in the queue.</i>
--	---

- virtual void **clear** ()

Removes all of the elements from this collection (optional operation).

The collection will be empty after this method returns.

*This implementation iterates over this collection, removing each element using the **Iterator.remove** (p. 1560) operation. Most implementations will probably choose to override this method for efficiency.*

*Note that this implementation will throw an **UnsupportedOperationException** if the iterator returned by this collection's iterator method does not implement the remove method and this collection is non-empty.*

Exceptions

UnsupportedOperationException	<i>if the clear operation is not supported by this collection</i>
--------------------------------------	---

6.7.1 Detailed Description

```
template<typename E>class decaf::util::AbstractQueue< E >
```

This class provides skeletal implementations of some **Queue** (p. 2222) operations.

Methods add, remove, and element are based on offer, poll, and peek, respectively.

A **Queue** (p. 2222) implementation that extends this class must minimally define a method **Queue** (p. 2222). **offer(E)** which does not permit insertion of null elements, along with methods **Queue** (p. 2222). **peek()** (p. 2225), **Queue.poll()** (p. 2225), **Collection.size()** (p. 864), and a **Collection.iterator()** (p. 1557) supporting **Iterator.remove()** (p. 1560). Typically, additional methods will be overridden as well. If these requirements cannot be met, consider instead subclassing **AbstractCollection** (p. 106).

Since

1.0

6.7.2 Constructor & Destructor Documentation

```
6.7.2.1 template<typename E > decaf::util::AbstractQueue< E >::AbstractQueue ( )  
[inline]
```

```
6.7.2.2 template<typename E > virtual decaf::util::AbstractQueue< E  
>::~~AbstractQueue ( ) [inline, virtual]
```

6.7.3 Member Function Documentation

6.7.3.1 `template<typename E> virtual bool decaf::util::AbstractQueue< E >::add (const E & value) [inline, virtual]`

Returns true if this collection changed as a result of the call.

(Returns false if this collection does not permit duplicates and already contains the specified element.)

Collections that support this operation may place limitations on what elements may be added to this collection. In particular, some collections will refuse to add null elements, and others will impose restrictions on the type of elements that may be added. **Collection** (p. 851) classes should clearly specify in their documentation any restrictions on what elements may be added.

If a collection refuses to add a particular element for any reason other than that it already contains the element, it must throw an exception (rather than returning false). This preserves the invariant that a collection always contains the specified element after this call returns.

For non-pointer values, i.e. class instances or string's the object will be copied into the collection, thus the object must support being copied, must not hide the copy constructor and assignment operator.

Parameters

<i>value</i>	The reference to the element to add to this Collection (p. 851).
--------------	---

Returns

true if the element was added to this **Collection** (p. 851).

Exceptions

<i>UnsupportedOperationException</i>	if this is an unmodifiable collection.
<i>NullPointerException</i>	if the Collection (p. 851) is a container of pointers and does not allow NULL values.
<i>IllegalArgumentException</i>	if some property of the element prevents it from being added to this collection
<i>IllegalStateException</i>	if the element cannot be added at this time due to insertion restrictions.

This implementation returns true if offer succeeds, else throws an `IllegalStateException`.

Implements `decaf::util::Collection< E >` (p. 853).

Reimplemented in `decaf::util::PriorityQueue< E >` (p. 2165).

References `decaf::util::Queue< E >::offer()`.

6.7.3.2 `template<typename E > virtual bool decaf::util::AbstractQueue< E >::addAll (const Collection< E > & collection) [inline, virtual]`

Adds all of the elements in the specified collection to this collection.

The behavior of this operation is undefined if the specified collection is modified while the operation is in progress. (This implies that the behavior of this call is undefined if the specified collection is this collection, and this collection is nonempty.)

Parameters

<i>collection</i>	The Collection (p. 851) whose elements are added to this one.
-------------------	--

Returns

true if this collection changed as a result of the call

Exceptions

<i>UnsupportedOperationException</i>	if this is an unmodifiable collection.
<i>NullPointerException</i>	if the Collection (p. 851) is a container of pointers and does not allow NULL values.
<i>IllegalArgumentException</i>	if some property of an element prevents it from being added to this collection
<i>IllegalStateException</i>	if an element cannot be added at this time due to insertion restrictions.

This implementation iterates over the specified collection, and adds each object returned by the iterator to this collection, in turn.

Note that this implementation will throw an UnsupportedOperationException unless add is overridden (assuming the specified collection is non-empty).

This implementation checks to see if the **Queue** (p. 2222) is being added to itself and throws an IllegalArgumentException if so, otherwise it delegates the add to the **AbstractCollection** (p. 106)'s addAll implementation.

Reimplemented from **decaf::util::AbstractCollection< E >** (p. 110).

Referenced by decaf::util::concurrent::LinkedBlockingQueue< E >::operator=().

6.7.3.3 `template<typename E > virtual void decaf::util::AbstractQueue< E >::clear () [inline, virtual]`

Removes all of the elements from this collection (optional operation).

The collection will be empty after this method returns.

This implementation iterates over this collection, removing each element using the **Iterator.remove** (p. 1560) operation. Most implementations will probably choose to override this method for efficiency.

Note that this implementation will throw an `UnsupportedOperationException` if the iterator returned by this collection's iterator method does not implement the remove method and this collection is non-empty.

Exceptions

<i>Unsupported- OperationException</i>	if the clear operation is not supported by this collection
--	--

This implementation repeatedly invokes poll until it returns false.

Reimplemented from `decaf::util::AbstractCollection< E >` (p. 111).

Reimplemented in `decaf::util::PriorityQueue< E >` (p. 2166), `decaf::util::concurrent::SynchronousQueue< E >` (p. 2658), and `decaf::util::concurrent::LinkedBlockingQueue< E >` (p. 1624).

References `decaf::util::AbstractCollection< E >::isEmpty()`, and `decaf::util::Queue< E >::poll()`.

6.7.3.4 `template<typename E> virtual E decaf::util::AbstractQueue< E >::element () const [inline, virtual]`

Gets but not removes the element in the head of the queue.

Throws a **NoSuchElementException** (p. 1984) if there is no element in the queue.

Returns

the element in the head of the queue.

Exceptions

NoSuchElementException (p. 1984)	if there is no element in the queue.
--	--------------------------------------

This implementation returns the result of peek unless the queue is empty otherwise it throws a **NoSuchElementException** (p. 1984).

Implements `decaf::util::Queue< E >` (p. 2223).

References `decaf::util::Queue< E >::peek()`.

Referenced by `decaf::util::concurrent::SynchronousQueue< E >::drainTo()`.

6.7.3.5 `template<typename E> virtual E decaf::util::AbstractQueue< E >::remove () [inline, virtual]`

Gets and removes the element in the head of the queue.

Throws a **NoSuchElementException** (p. 1984) if there is no element in the queue.

Returns

the element in the head of the queue.

Exceptions

NoSuchElementException (p. 1984)	if there is no element in the queue.
--	--------------------------------------

This implementation returns the result of poll unless the queue is empty.

Implements **decaf::util::Queue< E >** (p. 2226).

Reimplemented in **decaf::util::PriorityQueue< E >** (p. 2169).

References **decaf::util::Queue< E >::poll()**.

The documentation for this class was generated from the following file:

- src/main/decaf/util/**AbstractQueue.h**

6.8 decaf::util::AbstractSequentialList< E > Class Template - Reference

This class provides a skeletal implementation of the **List** (p. 1658) interface to minimize the effort required to implement this interface backed by a "sequential access" data store (such as a linked list).

```
#include <src/main/decaf/util/AbstractSequentialList.h>
```

Inheritance diagram for **decaf::util::AbstractSequentialList< E >**:

Public Member Functions

- virtual **~AbstractSequentialList** ()
- virtual **Iterator< E > * iterator** ()
- virtual **Iterator< E > * iterator** () const
- virtual **ListIterator< E > * listIterator** ()
- virtual **ListIterator< E > * listIterator** () const
- virtual **ListIterator< E > * listIterator** (int index **DECAF_UNUSED**)
- virtual **ListIterator< E > * listIterator** (int index **DECAF_UNUSED**) const
- virtual **E get** (int index) const

Gets the element contained at position passed.

Parameters

index	The position to get.
-------	----------------------

Returns

value at index specified.

Exceptions

IndexOutOfBoundsException	<i>if the index given is less than zero or greater than the List (p. 1658) size.</i>
---------------------------	---

- virtual **E set** (int index, const E &element)

Replaces the element at the specified position in this list with the specified element.

Parameters

index	<i>The index of the element to replace.</i>
element	<i>The element to be stored at the specified position.</i>

Returns

the element previously at the specified position.

Exceptions

IndexOutOfBoundsException	<i>if the index given is less than zero or greater than the List (p. 1658) size.</i>
UnsupportedOperationException	<i>if this is an unmodifiable collection.</i>
NullPointerException	<i>if the Collection (p. 851) is a container of pointers and does not allow NULL values.</i>
IllegalArgumentException	<i>if some property of the element prevents it from being added to this collection</i>
IllegalStateException	<i>if the element cannot be added at this time due to insertion restrictions.</i>

- virtual void **add** (int index, const E &element)

Inserts the specified element at the specified position in this list.

Shifts the element currently at that position (if any) and any subsequent elements to the right (adds one to their indices).

Parameters

index	<i>The index at which the specified element is to be inserted.</i>
element	<i>The element to be inserted in this List (p. 1658).</i>

Exceptions

IndexOutOfBoundsException	<i>if the index is greater than size of the List (p. 1658).</i>
UnsupportedOperationException	<i>if this is an unmodifiable collection.</i>
NullPointerException	<i>if the Collection (p. 851) is a container of pointers and does not allow NULL values.</i>
IllegalArgumentException	<i>if some property of the element prevents it from being added to this collection</i>
IllegalStateException	<i>if the element cannot be added at this time due to insertion restrictions.</i>

- virtual bool **addAll** (int index, const **Collection**< E > &source)

Inserts all of the elements in the specified collection into this list at the specified position (optional operation).

Shifts the element currently at that position (if any) and any subsequent elements to the right (increases their indices). The new elements will appear in this list in the order that they are returned by the specified collection's iterator. The behavior of this operation

is undefined if the specified collection is modified while the operation is in progress.
(Note that this will occur if the specified collection is this list, and it's nonempty.)

Parameters

index	The index at which to insert the first element from the specified collection
source	The Collection (p. 851) containing elements to be added to this list

Returns

true if this list changed as a result of the call

Exceptions

IndexOutOfBoundsException	if the index given is less than zero or greater than the List (p. 1658) size.
UnsupportedOperationException	if this is an unmodifiable collection.
NullPointerException	if the Collection (p. 851) is a container of pointers and does not allow NULL values.
IllegalArgumentException	if some property of the element prevents it from being added to this collection
IllegalStateException	if the element cannot be added at this time due to insertion restrictions.

- virtual E **removeAt** (int index)

Removes the element at the specified position in this list.
Shifts any subsequent elements to the left (subtracts one from their indices). Returns the element that was removed from the list.

Parameters

index	- the index of the element to be removed.
-------	---

Returns

the element previously at the specified position.

Exceptions

IndexOutOfBoundsException	if the index given is less than zero or greater than the List (p. 1658) size.
UnsupportedOperationException	if this is an unmodifiable collection.

6.8.1 Detailed Description

```
template<typename E>class decaf::util::AbstractSequentialList< E >
```

This class provides a skeletal implementation of the **List** (p. 1658) interface to minimize the effort required to implement this interface backed by a "sequential access" data store (such as a linked list).

For random access data (such as an array), **AbstractList** (p. 123) should be used in preference to this class.

This class is the opposite of the **AbstractList** (p. 123) class in the sense that it implements the "random access" methods (get(int index), set(int index, E element), add(int index, E element) and remove(int index)) on top of the list's list iterator, instead of the other way around.

To implement a list the programmer needs only to extend this class and provide implementations for the `listIterator` and `size` methods. For an unmodifiable list, the programmer need only implement the list iterator's `hasNext`, `next`, `hasPrevious`, `previous` and `index` methods.

For a modifiable list the programmer should additionally implement the list iterator's `set` method. For a variable-size list the programmer should additionally implement the list iterator's `remove` and `add` methods.

The programmer should generally provide a void (no argument) and collection constructor, as per the recommendation in the **Collection** (p. 851) interface specification.

Since

1.0

6.8.2 Constructor & Destructor Documentation

6.8.2.1 `template<typename E> virtual decaf::util::AbstractSequentialList< E
>::~AbstractSequentialList() [inline, virtual]`

6.8.3 Member Function Documentation

6.8.3.1 `template<typename E> virtual void decaf::util::AbstractSequentialList< E
>::add(int index , const E & element) [inline, virtual]`

Inserts the specified element at the specified position in this list.

Shifts the element currently at that position (if any) and any subsequent elements to the right (adds one to their indices).

Parameters

<i>index</i>	The index at which the specified element is to be inserted.
<i>element</i>	The element to be inserted in this List (p. 1658).

Exceptions

<i>IndexOutOfBoundsException</i>	if the index is greater than size of the List (p. 1658).
<i>UnsupportedOperationException</i>	if this is an unmodifiable collection.
<i>NullPointerException</i>	if the Collection (p. 851) is a container of pointers and does not allow NULL values.
<i>IllegalArgumentException</i>	if some property of the element prevents it from being added to this collection
<i>IllegalStateException</i>	if the element cannot be added at this time due to insertion restrictions.

This implementation first gets a list iterator pointing to the indexed element (with list-

Iterator(index)). Then, it inserts the specified element with **ListIterator.add** (p. 1672).

Implements **decaf::util::List< E >** (p. 1660).

Reimplemented in **decaf::util::LinkedList< E >** (p. 1640), **decaf::util::LinkedList< Pointer< Transport > >** (p. 1640), **decaf::util::LinkedList< cms::MessageConsumer * >** (p. 1640), **decaf::util::LinkedList< CompositeTask * >** (p. 1640), **decaf::util::LinkedList< URI >** (p. 1640), **decaf::util::LinkedList< Pointer< MessageDispatch > >** (p. 1640), **decaf::util::LinkedList< Pointer< DestinationInfo > >** (p. 1640), **decaf::util::LinkedList< PrimitiveValueNode >** (p. 1640), **decaf::util::LinkedList< Pointer< Command > >** (p. 1640), **decaf::util::LinkedList< BackupTransport > >** (p. 1640), **decaf::util::LinkedList< cms::MessageProducer * >** (p. 1640), **decaf::util::LinkedList< cms::Destination * >** (p. 1640), **decaf::util::LinkedList< cms::Session * >** (p. 1640), and **decaf::util::LinkedList< cms::Connection * >** (p. 1640).

```
6.8.3.2 template<typename E> virtual bool decaf::util::AbstractSequentialList<
    E>::addAll ( int index, const Collection< E > & source ) [inline,
    virtual]
```

Inserts all of the elements in the specified collection into this list at the specified position (optional operation).

Shifts the element currently at that position (if any) and any subsequent elements to the right (increases their indices). The new elements will appear in this list in the order that they are returned by the specified collection's iterator. The behavior of this operation is undefined if the specified collection is modified while the operation is in progress. (Note that this will occur if the specified collection is this list, and it's nonempty.)

Parameters

<i>index</i>	The index at which to insert the first element from the specified collection
<i>source</i>	The Collection (p. 851) containing elements to be added to this list

Returns

true if this list changed as a result of the call

Exceptions

<i>IndexOutOfBoundsException</i>	if the index given is less than zero or greater than the List (p. 1658) size.
<i>UnsupportedOperationException</i>	if this is an unmodifiable collection.
<i>NullPointerException</i>	if the Collection (p. 851) is a container of pointers and does not allow NULL values.
<i>IllegalArgumentException</i>	if some property of the element prevents it from being added to this collection
<i>IllegalStateException</i>	if the element cannot be added at this time due to insertion restrictions.

This implementation gets an iterator over the specified collection and a list iterator over this list pointing to the indexed element (with `listIterator(index)`). Then, it iterates over the specified collection, inserting the elements obtained from the iterator into this list, one at a time, using `ListIterator.add` (p. 1672) (to skip over the added element).

Reimplemented from `decaf::util::AbstractList< E >` (p. 126).

Reimplemented in `decaf::util::LinkedList< E >` (p. 1642), `decaf::util::LinkedList< Pointer< Transport > >` (p. 1642), `decaf::util::LinkedList< cms::MessageConsumer * >` (p. 1642), `decaf::util::LinkedList< CompositeTask * >` (p. 1642), `decaf::util::LinkedList< URI >` (p. 1642), `decaf::util::LinkedList< Pointer< MessageDispatch > >` (p. 1642), `decaf::util::LinkedList< Pointer< DestinationInfo > >` (p. 1642), `decaf::util::LinkedList< PrimitiveValueNode >` (p. 1642), `decaf::util::LinkedList< Pointer< Command > >` (p. 1642), `decaf::util::LinkedList< Pointer< BackupTransport > >` (p. 1642), `decaf::util::LinkedList< cms::MessageProducer * >` (p. 1642), `decaf::util::LinkedList< cms::Destination * >` (p. 1642), `decaf::util::LinkedList< cms::Session * >` (p. 1642), and `decaf::util::LinkedList< cms::Connection * >` (p. 1642).

6.8.3.3 `template<typename E> virtual E decaf::util::AbstractSequentialList< E >::get (int index) const [inline, virtual]`

Gets the element contained at position passed.

Parameters

<i>index</i>	The position to get.
--------------	----------------------

Returns

value at index specified.

Exceptions

<i>IndexOutOfBoundsException</i>	if the index given is less than zero or greater than the List (p. 1658) size.
----------------------------------	--

This implementation first gets a list iterator pointing to the indexed element (with `listIterator(index)`). Then, it gets the element using `ListIterator.next` (p. 1560) and returns it.

Implements `decaf::util::List< E >` (p. 1662).

Reimplemented in `decaf::util::LinkedList< E >` (p. 1646), `decaf::util::LinkedList< Pointer< Transport > >` (p. 1646), `decaf::util::LinkedList< cms::MessageConsumer * >` (p. 1646), `decaf::util::LinkedList< CompositeTask * >` (p. 1646), `decaf::util::LinkedList< URI >` (p. 1646), `decaf::util::LinkedList< Pointer< MessageDispatch > >` (p. 1646), `decaf::util::LinkedList< Pointer< DestinationInfo > >` (p. 1646), `decaf::util::LinkedList< PrimitiveValueNode >` (p. 1646), `decaf::util::LinkedList< Pointer< Command > >` (p. 1646), `decaf::util::LinkedList`

List< **Pointer**< **BackupTransport** > > (p. 1646), **decaf::util::LinkedList**< **cms::MessageProducer** * > (p. 1646), **decaf::util::LinkedList**< **cms::Destination** * > (p. 1646), **decaf::util::LinkedList**< **cms::Session** * > (p. 1646), and **decaf::util::LinkedList**< **cms::Connection** * > (p. 1646).

```
6.8.3.4 template<typename E> virtual Iterator<E>* decaf::util::-
    AbstractSequentialList< E >::iterator ( ) [inline,
        virtual]
```

Returns

an iterator over a set of elements of type T.

Reimplemented from **decaf::util::AbstractList**< **E** > (p. 128).

Referenced by **decaf::util::LinkedList**< **cms::Connection** * >::removeFirstOccurrence().

```
6.8.3.5 template<typename E> virtual Iterator<E>* decaf::util::-
    AbstractSequentialList< E >::iterator ( ) const [inline,
        virtual]
```

Reimplemented from **decaf::util::AbstractList**< **E** > (p. 129).

```
6.8.3.6 template<typename E> virtual ListIterator<E>* decaf::util::-
    AbstractSequentialList< E >::listIterator ( ) [inline,
        virtual]
```

Returns

a list iterator over the elements in this list (in proper sequence).

Reimplemented from **decaf::util::AbstractList**< **E** > (p. 130).

Referenced by **decaf::util::AbstractSequentialList**< **cms::Connection** * >::add(), **decaf::util::AbstractSequentialList**< **cms::Connection** * >::addAll(), **decaf::util::AbstractSequentialList**< **cms::Connection** * >::get(), **decaf::util::AbstractSequentialList**< **cms::Connection** * >::iterator(), **decaf::util::AbstractSequentialList**< **cms::Connection** * >::listIterator(), **decaf::util::AbstractSequentialList**< **cms::Connection** * >::removeAt(), and **decaf::util::AbstractSequentialList**< **cms::Connection** * >::set().

```
6.8.3.7 template<typename E> virtual ListIterator<E>* decaf::util::-
    AbstractSequentialList< E >::listIterator ( ) const [inline,
        virtual]
```

Reimplemented from **decaf::util::AbstractList**< **E** > (p. 131).

```
6.8.3.8 template<typename E> virtual ListIterator<E>* decaf::util::Abstract-
SequentialList< E >::listIterator ( int index index ) [inline,
virtual]
```

Parameters

<i>index</i>	index of first element to be returned from the list iterator (by a call to the next method).
--------------	--

Returns

a list iterator of the elements in this list (in proper sequence), starting at the specified position in this list. The specified index indicates the first element that would be returned by an initial call to next. An initial call to previous would return the element with the specified index minus one.

Exceptions

<i>IndexOutOfBoundsException</i>	if the index is out of range ($\text{index} < 0 \parallel \text{index} > \text{size}()$ (p. 864))
----------------------------------	--

Reimplemented from **decaf::util::AbstractList< E >** (p. 131).

Reimplemented in **decaf::util::LinkedList< E >** (p. 1649), **decaf::util::LinkedList< Pointer< Transport > >** (p. 1649), **decaf::util::LinkedList< cms::MessageConsumer * >** (p. 1649), **decaf::util::LinkedList< CompositeTask * >** (p. 1649), **decaf::util::LinkedList< URI >** (p. 1649), **decaf::util::LinkedList< Pointer< MessageDispatch > >** (p. 1649), **decaf::util::LinkedList< Pointer< DestinationInfo > >** (p. 1649), **decaf::util::LinkedList< PrimitiveValueNode >** (p. 1649), **decaf::util::LinkedList< Pointer< Command > >** (p. 1649), **decaf::util::LinkedList< Pointer< BackupTransport > >** (p. 1649), **decaf::util::LinkedList< cms::MessageProducer * >** (p. 1649), **decaf::util::LinkedList< cms::Destination * >** (p. 1649), **decaf::util::LinkedList< cms::Session * >** (p. 1649), and **decaf::util::LinkedList< cms::Connection * >** (p. 1649).

```
6.8.3.9 template<typename E> virtual ListIterator<E>* decaf::util::Abstract-
SequentialList< E >::listIterator ( int index DECAF_UNUSED ) const
[inline, virtual]
```

Reimplemented from **decaf::util::AbstractList< E >** (p. 132).

Reimplemented in **decaf::util::LinkedList< E >** (p. 1649), **decaf::util::LinkedList< Pointer< Transport > >** (p. 1649), **decaf::util::LinkedList< cms::MessageConsumer * >** (p. 1649), **decaf::util::LinkedList< CompositeTask * >** (p. 1649), **decaf::util::LinkedList< URI >** (p. 1649), **decaf::util::LinkedList< Pointer< MessageDispatch > >** (p. 1649), **decaf::util::LinkedList< Pointer< DestinationInfo > >** (p. 1649), **decaf::util::LinkedList< PrimitiveValueNode >** (p. 1649), **decaf::util::LinkedList< Pointer< Command > >** (p. 1649), **decaf::util::LinkedList< Pointer< BackupTransport > >** (p. 1649), **decaf::util::LinkedList< cms::MessageProducer * >** (p. 1649), **decaf::util::LinkedList< cms::Destination * >**

(p. 1649), **decaf::util::LinkedList**< **cms::Session** * > (p. 1649), and **decaf::util::LinkedList**< **cms::Connection** * > (p. 1649).

6.8.3.10 `template<typename E> virtual E decaf::util::AbstractSequentialList< E
>::removeAt (int index) [inline, virtual]`

Removes the element at the specified position in this list.

Shifts any subsequent elements to the left (subtracts one from their indices). Returns the element that was removed from the list.

Parameters

<i>index</i>	- the index of the element to be removed.
--------------	---

Returns

the element previously at the specified position.

Exceptions

<i>IndexOutOfBoundsException</i>	if the index given is less than zero or greater than the List (p. 1658) size.
<i>UnsupportedOperationException</i>	if this is an unmodifiable collection.

This implementation first gets a list iterator pointing to the indexed element (with `listIterator(index)`). Then, it removes the element with **ListIterator.remove** (p. 1560).

Reimplemented from **decaf::util::AbstractList**< **E** > (p. 133).

6.8.3.11 `template<typename E> virtual E decaf::util::AbstractSequentialList< E
>::set (int index , const E & element) [inline, virtual]`

Replaces the element at the specified position in this list with the specified element.

Parameters

<i>index</i>	The index of the element to replace.
<i>element</i>	The element to be stored at the specified position.

Returns

the element previously at the specified position.

Exceptions

<i>IndexOutOfBoundsException</i>	if the index given is less than zero or greater than the List (p. 1658) size.
----------------------------------	--

<i>UnsupportedOperationException</i>	if this is an unmodifiable collection.
<i>NullPointerException</i>	if the Collection (p. 851) is a container of pointers and does not allow NULL values.
<i>IllegalArgumentException</i>	if some property of the element prevents it from being added to this collection
<i>IllegalStateException</i>	if the element cannot be added at this time due to insertion restrictions.

This implementation first gets a list iterator pointing to the indexed element (with `listIterator(index)`). Then, it gets the current element using **ListIterator.next** (p. 1560) and replaces it with **ListIterator.set** (p. 1674).

Implements **decaf::util::List< E >** (p. 1669).

Reimplemented in **decaf::util::LinkedList< E >** (p. 1657), **decaf::util::LinkedList< Pointer< Transport > >** (p. 1657), **decaf::util::LinkedList< cms::MessageConsumer * >** (p. 1657), **decaf::util::LinkedList< CompositeTask * >** (p. 1657), **decaf::util::LinkedList< URI >** (p. 1657), **decaf::util::LinkedList< Pointer< MessageDispatch > >** (p. 1657), **decaf::util::LinkedList< Pointer< DestinationInfo > >** (p. 1657), **decaf::util::LinkedList< PrimitiveValueNode >** (p. 1657), **decaf::util::LinkedList< Pointer< Command > >** (p. 1657), **decaf::util::LinkedList< Pointer< BackupTransport > >** (p. 1657), **decaf::util::LinkedList< cms::MessageProducer * >** (p. 1657), **decaf::util::LinkedList< cms::Destination * >** (p. 1657), **decaf::util::LinkedList< cms::Session * >** (p. 1657), and **decaf::util::LinkedList< cms::Connection * >** (p. 1657).

The documentation for this class was generated from the following file:

- `src/main/decaf/util/AbstractSequentialList.h`

6.9 decaf::util::AbstractSet< E > Class Template Reference

This class provides a skeletal implementation of the **Set** (p. 2397) interface to minimize the effort required to implement this interface.

```
#include <src/main/decaf/util/AbstractSet.h>
```

Inheritance diagram for **decaf::util::AbstractSet< E >**:

Public Member Functions

- virtual **~AbstractSet** ()
- virtual bool **removeAll** (const **Collection**< E > &collection)
Removes all this collection's elements that are also contained in the specified collection (optional operation).

After this call returns, this collection will contain no elements in common with the specified collection.

Parameters

collection	The Collection (p. 851) whose elements are to be removed from this one.
------------	--

Returns

true if the collection changed as a result of this call.

Exceptions

UnsupportedOperationException	if this is an unmodifiable collection.
NullPointerException	if the Collection (p. 851) is a container of pointers and does not allow NULL values.

This implementation iterates over this collection, checking each element returned by the iterator in turn to see if it's contained in the specified collection. If it's so contained, it's removed from this collection with the iterator's remove method.

Note that this implementation will throw an UnsupportedOperationException if the iterator returned by the iterator method does not implement the remove method and this collection contains one or more elements in common with the specified collection.

6.9.1 Detailed Description

template<typename E>class decaf::util::AbstractSet< E >

This class provides a skeletal implementation of the **Set** (p. 2397) interface to minimize the effort required to implement this interface.

The process of implementing a set by extending this class is identical to that of implementing a **Collection** (p. 851) by extending **AbstractCollection** (p. 106), except that all of the methods and constructors in subclasses of this class must obey the additional constraints imposed by the **Set** (p. 2397) interface (for instance, the add method must not permit addition of multiple instances of an object to a set).

Since

1.0

6.9.2 Constructor & Destructor Documentation

6.9.2.1 template<typename E> virtual decaf::util::AbstractSet< E >::~~AbstractSet () [inline, virtual]

6.9.3 Member Function Documentation

6.9.3.1 template<typename E> virtual bool decaf::util::AbstractSet< E >::removeAll (const Collection< E > & collection) [inline, virtual]

Removes all this collection's elements that are also contained in the specified collection (optional operation).

After this call returns, this collection will contain no elements in common with the specified collection.

Parameters

<i>collection</i>	The Collection (p. 851) whose elements are to be removed from this one.
-------------------	--

Returns

true if the collection changed as a result of this call.

Exceptions

<i>UnsupportedOperationException</i>	if this is an unmodifiable collection.
<i>NullPointerException</i>	if the Collection (p. 851) is a container of pointers and does not allow NULL values.

This implementation iterates over this collection, checking each element returned by the iterator in turn to see if it's contained in the specified collection. If it's so contained, it's removed from this collection with the iterator's remove method.

Note that this implementation will throw an `UnsupportedOperationException` if the iterator returned by the iterator method does not implement the remove method and this collection contains one or more elements in common with the specified collection.

This implementation determines which is the smaller of this set and the specified collection, by invoking the size method on each. If this set has fewer elements, then the implementation iterates over this set, checking each element returned by the iterator in turn to see if it is contained in the specified collection. If it is so contained, it is removed from this set with the iterator's remove method. If the specified collection has fewer elements, then the implementation iterates over the specified collection, removing from this set each element returned by the iterator, using this set's remove method.

Note that this implementation will throw an `UnsupportedOperationException` if the iterator returned by the iterator method does not implement the remove method.

Reimplemented from **decaf::util::AbstractCollection**< **E** > (p. 117).

Reimplemented in **decaf::util::concurrent::CopyOnWriteArraySet**< **E** > (p. 1059).

The documentation for this class was generated from the following file:

- src/main/decaf/util/**AbstractSet.h**

6.10 activemq::transport::AbstractTransportFactory Class - Reference

Abstract implementation of the **TransportFactory** (p. 2798) interface, providing the base functionality that's common to most of the **TransportFactory** (p. 2798) instances.


```
#include <src/main/activemq/transport/AbstractTransport-
Factory.h>
```

Inheritance diagram for activemq::transport::AbstractTransportFactory:

Public Member Functions

- virtual `~AbstractTransportFactory()`

Protected Member Functions

- virtual `Pointer< wireformat::WireFormat > createWireFormat (const decaf::util::Properties &properties)`

*Creates the WireFormat that is configured for this **Transport** (p. 2790) and returns it.*

6.10.1 Detailed Description

Abstract implementation of the **TransportFactory** (p. 2798) interface, providing the base functionality that's common to most of the **TransportFactory** (p. 2798) instances.

Since

3.0

6.10.2 Constructor & Destructor Documentation

6.10.2.1 `virtual activemq::transport::AbstractTransportFactory::~AbstractTransportFactory() [inline, virtual]`

6.10.3 Member Function Documentation

6.10.3.1 `virtual Pointer< wireformat::WireFormat > activemq::transport::AbstractTransportFactory::createWireFormat (const decaf::util::Properties & properties) [protected, virtual]`

Creates the WireFormat that is configured for this **Transport** (p. 2790) and returns it.

The default WireFormat is Openwire.

Parameters

<i>properties</i>	The properties that were configured on the URI.
-------------------	---

Returns

a pointer to a WireFormat instance that the caller then owns.

Exceptions

<i>NoSuchElementException</i>	if the configured WireFormat is not found.
-------------------------------	--

The documentation for this class was generated from the following file:

- src/main/activemq/transport/**AbstractTransportFactory.h**

6.11 activemq::core::ActiveMQAckHandler Class Reference

Interface class that is used to give CMS Messages an interface to Ack themselves with.

```
#include <src/main/activemq/core/ActiveMQAckHandler.h>
```

Public Member Functions

- virtual **~ActiveMQAckHandler** ()
- virtual void **acknowledgeMessage** (const **commands::Message** *message)=0

Method called to acknowledge the message once it has been received by a Message-Consumer.

6.11.1 Detailed Description

Interface class that is used to give CMS Messages an interface to Ack themselves with.

Since

2.0

6.11.2 Constructor & Destructor Documentation

6.11.2.1 virtual **activemq::core::ActiveMQAckHandler::~ActiveMQAckHandler** ()
[inline, virtual]

6.11.3 Member Function Documentation

6.11.3.1 virtual void **activemq::core::ActiveMQAckHandler::acknowledgeMessage** (const **commands::Message** * *message*) [pure virtual]

Method called to acknowledge the message once it has been received by a Message-Consumer.

Parameters

<i>message</i>	The Message to Acknowledge.
----------------	-----------------------------

Exceptions

<i>CMSEException</i>	if an error occurs while acknowledging the given Message.
----------------------	---

The documentation for this class was generated from the following file:

- src/main/activemq/core/**ActiveMQAckHandler.h**

6.12 activemq::commands::ActiveMQBlobMessage Class Reference

```
#include <src/main/activemq/commands/ActiveMQBlobMessage.h>
```

Inheritance diagram for activemq::commands::ActiveMQBlobMessage:

Public Member Functions

- **ActiveMQBlobMessage** ()
- virtual **~ActiveMQBlobMessage** () throw ()
- virtual unsigned char **getDataStructureType** () const
*Get the **DataStructure** (p. 1133) Type as defined in CommandTypes.h.*
- virtual **ActiveMQBlobMessage** * **cloneDataStructure** () const
Clone this object and return a new instance that the caller now owns, this will be an exact copy of this one.
- virtual void **copyDataStructure** (const **DataStructure** *src)
Copy the contents of the passed object into this objects members, overwriting any existing data.
- virtual std::string **toString** () const
*Returns a string containing the information for this **DataStructure** (p. 1133) such as its type and value of its elements.*
- virtual bool **equals** (const **DataStructure** *value) const
*Compares the **DataStructure** (p. 1133) passed in to this one, and returns if they are equivalent.*
- virtual **cms::Message** * **clone** () const
Clone this message exactly, returns a new instance that the caller is required to delete.
- std::string **getRemoteBlobUrl** () const
Get the Remote URL of the Blob.
- void **setRemoteBlobUrl** (const std::string &remoteURL)
Set the Remote URL of the Blob.

- `std::string getMimeType () const`
Get the Mime Type of the Blob.
- `void setMimeType (const std::string &mimeType)`
Set the Mime Type of the Blob.
- `std::string getName () const`
Gets the Name of the Blob.
- `void setName (const std::string &name)`
Sets the Name of the Blob.
- `bool isDeletedByBroker () const`
Gets if this Blob is deleted by the Broker.
- `void setDeletedByBroker (bool value)`
Sets the Deleted By Broker flag.

Static Public Attributes

- static const unsigned char **ID_ACTIVEMQBLOBMESSAGE** = 29
- static const std::string **BINARY_MIME_TYPE**

6.12.1 Constructor & Destructor Documentation

6.12.1.1 `activemq::commands::ActiveMQBlobMessage::ActiveMQBlobMessage ()`

6.12.1.2 `virtual activemq::commands::ActiveMQBlobMessage-
::~~ActiveMQBlobMessage () throw () [inline,
virtual]`

6.12.2 Member Function Documentation

6.12.2.1 `virtual cms::Message* activemq::commands::Active-
MQBlobMessage::clone () const [inline,
virtual]`

Clone this message exactly, returns a new instance that the caller is required to delete.

Returns

new copy of this message

Implements `cms::Message` (p. 1844).

6.12.2.2 `virtual ActiveMQBlobMessage* activemq::commands::-
ActiveMQBlobMessage::cloneDataStructure () const
[virtual]`

Clone this object and return a new instance that the caller now owns, this will be an exact copy of this one.

Returns

new copy of this object.

Reimplemented from **activemq::commands::Message** (p. 1826).

6.12.2.3 `virtual void activemq::commands::ActiveMQBlobMessage::copyDataStructure (const DataStructure * src)`
[virtual]

Copy the contents of the passed object into this objects members, overwriting any existing data.

Returns

src - Source Object

Reimplemented from **activemq::commands::Message** (p. 1827).

6.12.2.4 `virtual bool activemq::commands::ActiveMQBlobMessage::equals (const DataStructure * value) const` [virtual]

Compares the **DataStructure** (p. 1133) passed in to this one, and returns if they are equivalent.

Equivalent here means that they are of the same type, and that each element of the objects are the same.

Returns

true if **DataStructure** (p. 1133)'s are Equal.

Reimplemented from **activemq::commands::ActiveMQMessageTemplate< cms::Message >** (p. 297).

6.12.2.5 `virtual unsigned char activemq::commands::ActiveMQBlobMessage::getDataStructureType () const`
[virtual]

Get the **DataStructure** (p. 1133) Type as defined in CommandTypes.h.

Returns

The type of the data structure

Reimplemented from **activemq::commands::Message** (p. 1829).

6.12.2.6 `std::string activemq::commands::ActiveMQBlobMessage::getMimeType () const [inline]`

Get the Mime Type of the Blob.

Returns

string holding the MIME Type.

6.12.2.7 `std::string activemq::commands::ActiveMQBlobMessage::getName () const [inline]`

Gets the Name of the Blob.

Returns

string name of the Blob.

6.12.2.8 `std::string activemq::commands::ActiveMQBlobMessage::getRemoteBlobUrl () const [inline]`

Get the Remote URL of the Blob.

Returns

string from of the Remote Blob URL.

6.12.2.9 `bool activemq::commands::ActiveMQBlobMessage::isDeletedByBroker () const [inline]`

Gets if this Blob is deleted by the Broker.

Returns

true if the Blob is deleted by the Broker.

6.12.2.10 `void activemq::commands::ActiveMQBlobMessage::setDeletedByBroker (bool value) [inline]`

Sets the Deleted By Broker flag.

Parameters

<i>value</i>	- set the Delete by broker flag to value.
--------------	---

6.12.2.11 `void activemq::commands::ActiveMQBlobMessage::setMimeType (const std::string & mimeType) [inline]`

Set the Mime Type of the Blob.

Parameters

<i>mimeType</i>	- String holding the MIME Type.
-----------------	---------------------------------

6.12.2.12 `void activemq::commands::ActiveMQBlobMessage::setName (const std::string & name) [inline]`

Sets the Name of the Blob.

Parameters

<i>name</i>	- Name of the Blob.
-------------	---------------------

6.12.2.13 `void activemq::commands::ActiveMQBlobMessage::setRemoteBlobUrl (const std::string & remoteURL) [inline]`

Set the Remote URL of the Blob.

Parameters

<i>remoteURL</i>	- String form of the Remote URL.
------------------	----------------------------------

6.12.2.14 `virtual std::string activemq::commands::ActiveMQBlobMessage::toString () const [virtual]`

Returns a string containing the information for this **DataStructure** (p. 1133) such as its type and value of its elements.

Returns

formatted string useful for debugging.

Reimplemented from **activemq::commands::Message** (p. 1836).

6.12.3 Field Documentation

6.12.3.1 `const std::string activemq::commands::ActiveMQBlobMessage::BINARY_MIME_TYPE [static]`

```
6.12.3.2  const unsigned char activemq::commands::ActiveMQ-
          BlobMessage::ID_ACTIVEMQBLOBMESSAGE = 29
          [static]
```

The documentation for this class was generated from the following file:

- src/main/activemq/commands/**ActiveMQBlobMessage.h**

6.13 activemq::wireformat::openwire::marshal::generated::ActiveMQBlobMessageMarshaller Class Reference

Marshaling code for Open Wire Format for **ActiveMQBlobMessageMarshaller** (p. 161).

```
#include <src/main/activemq/wireformat/openwire/marshal/generated/-
ActiveMQBlobMessageMarshaller.h>
```

Inheritance diagram for activemq::wireformat::openwire::marshal::generated::ActiveMQBlobMessageMarshaller:

Public Member Functions

- **ActiveMQBlobMessageMarshaller** ()
- virtual **~ActiveMQBlobMessageMarshaller** ()
- virtual **commands::DataStructure * createObject** () const
Creates a new instance of the class that this class is a marshaling director for.
- virtual unsigned char **getDataStructureType** () const
Gets the DataStructureType that this class marshals/unmarshals.
- virtual void **tightUnmarshal** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataInputStream** *dataIn, **utils::BooleanStream** *bs)
Tight Un-marhsal to the given stream.
- virtual int **tightMarshal1** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **utils::BooleanStream** *bs)
Tight Marhsal to the given stream.
- virtual void **tightMarshal2** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataOutputStream** *dataOut, **utils::BooleanStream** *bs)
Tight Marhsal to the given stream.
- virtual void **looseUnmarshal** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataInputStream** *dataIn)
Loose Un-marhsal to the given stream.
- virtual void **looseMarshal** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataOutputStream** *dataOut)
Tight Marhsal to the given stream.

6.13 activemq::wireformat::openwire::marshal::generated::ActiveMQBlobMessageMarshaller Class

Reference

163

6.13.1 Detailed Description

Marshaling code for Open Wire Format for **ActiveMQBlobMessageMarshaller** (p. 161).

NOTE!: This file is auto generated - do not modify! if you need to make a change, please see the Java Classes in the activemq-openwire-generator module

6.13.2 Constructor & Destructor Documentation

6.13.2.1 **activemq::wireformat::openwire::marshal::generated::ActiveMQBlobMessageMarshaller::ActiveMQBlobMessageMarshaller ()**
[inline]

6.13.2.2 **virtual activemq::wireformat::openwire::marshal::generated::ActiveMQBlobMessageMarshaller::~~ActiveMQBlobMessageMarshaller ()**
[inline, virtual]

6.13.3 Member Function Documentation

6.13.3.1 **virtual commands::DataStructure* activemq::wireformat::openwire::marshal::generated::ActiveMQBlobMessageMarshaller::createObject () const** [virtual]

Creates a new instance of the class that this class is a marshaling director for.

Returns

newly allocated Command

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1120).

6.13.3.2 **virtual unsigned char activemq::wireformat::openwire::marshal::generated::ActiveMQBlobMessageMarshaller::getDataStructureType () const**
[virtual]

Gets the DataStructureType that this class marshals/unmarshals.

Returns

byte Id of this classes DataStructureType

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1122).

6.13.3.3 virtual void `activemq::wireformat::openwire::marshal::generated::ActiveMQBlobMessageMarshaller::looseMarshal (OpenWireFormat * format, commands::DataStructure * command, decaf::io::DataOutputStream * ds)` [virtual]

Tight Marshal to the given stream.

Parameters

<i>format</i>	- The OpenWireFormat properties
<i>command</i>	- the object to Marshal
<i>ds</i>	- DataOutputStream to marshal to

Exceptions

<i>IOException</i>	if an error occurs.
--------------------	---------------------

Reimplemented from `activemq::wireformat::openwire::marshal::generated::MessageMarshaller` (p. 1919).

6.13.3.4 virtual void `activemq::wireformat::openwire::marshal::generated::ActiveMQBlobMessageMarshaller::looseUnmarshal (OpenWireFormat * format, commands::DataStructure * command, decaf::io::DataInputStream * dis)` [virtual]

Loose Un-marshal to the given stream.

Parameters

<i>format</i>	- The OpenWireFormat properties
<i>command</i>	- the object to Un-Marshal
<i>dis</i>	- the DataInputStream to Un-Marshal from

Exceptions

<i>IOException</i>	if an error occurs.
--------------------	---------------------

Reimplemented from `activemq::wireformat::openwire::marshal::generated::MessageMarshaller` (p. 1919).

6.13.3.5 virtual int `activemq::wireformat::openwire::marshal::generated::ActiveMQBlobMessageMarshaller::tightMarshal1 (OpenWireFormat * format, commands::DataStructure * command, utils::BooleanStream * bs)` [virtual]

Tight Marshal to the given stream.

6.13 activemq::wireformat::openwire::marshal::generated::ActiveMQBlobMessageMarshaller Class

Reference

165

Parameters

<i>format</i>	The OpenwireFormat properties
<i>command</i>	The object to Marshal
<i>bs</i>	The boolean stream to marshal to.

Exceptions

<i>IOException</i>	if an error occurs.
--------------------	---------------------

Reimplemented from **activemq::wireformat::openwire::marshal::generated::MessageMarshaller** (p. 1920).

6.13.3.6 virtual void **activemq::wireformat::openwire::marshal::generated::ActiveMQBlobMessageMarshaller::tightMarshal2** (**OpenWireFormat** * *format*, **commands::DataStructure** * *command*, **decaf::io::DataOutputStream** * *ds*, **utils::BooleanStream** * *bs*) [virtual]

Tight Marhsal to the given stream.

Parameters

<i>format</i>	- The OpenwireFormat properties
<i>command</i>	- the object to Marshal
<i>ds</i>	- the DataOutputStream to Marshal to
<i>bs</i>	- boolean stream to marshal to.

Exceptions

<i>IOException</i>	if an error occurs.
--------------------	---------------------

Reimplemented from **activemq::wireformat::openwire::marshal::generated::MessageMarshaller** (p. 1921).

6.13.3.7 virtual void **activemq::wireformat::openwire::marshal::generated::ActiveMQBlobMessageMarshaller::tightUnmarshal** (**OpenWireFormat** * *format*, **commands::DataStructure** * *command*, **decaf::io::DataInputStream** * *dis*, **utils::BooleanStream** * *bs*) [virtual]

Tight Un-marhsal to the given stream.

Parameters

<i>format</i>	- The OpenwireFormat properties
<i>command</i>	- the object to Un-Marshal
<i>dis</i>	- the DataInputStream to Un-Marshal from
<i>bs</i>	- boolean stream to unmarshal from.

Exceptions

<i>IOException</i> if an error occurs.
--

Reimplemented from **activemq::wireformat::openwire::marshal::generated::MessageMarshaller** (p. 1921).

The documentation for this class was generated from the following file:

- src/main/activemq/wireformat/openwire/marshal/generated/**ActiveMQBlobMessageMarshaller.h**

6.14 activemq::commands::ActiveMQBytesMessage Class - Reference

```
#include <src/main/activemq/commands/ActiveMQBytesMessage.h>
```

Inheritance diagram for **activemq::commands::ActiveMQBytesMessage**:

Public Member Functions

- **ActiveMQBytesMessage** ()
- virtual **~ActiveMQBytesMessage** () throw ()
- virtual unsigned char **getDataStructureType** () const
*Get the **DataStructure** (p. 1133) Type as defined in CommandTypes.h.*
- virtual **ActiveMQBytesMessage * cloneDataStructure** () const
Clone this object and return a new instance that the caller now owns, this will be an exact copy of this one.
- virtual void **copyDataStructure** (const **DataStructure** *src)
Copy the contents of the passed object into this objects members, overwriting any existing data.
- virtual std::string **toString** () const
*Returns a string containing the information for this **DataStructure** (p. 1133) such as its type and value of its elements.*
- virtual bool **equals** (const **DataStructure** *value) const
*Compares the **DataStructure** (p. 1133) passed in to this one, and returns if they are equivalent.*
- virtual **cms::BytesMessage * clone** () const
Clones this message.
- virtual void **clearBody** ()
Clears out the body of the message.
- virtual void **onSend** ()
*Allows derived **Message** (p. 1821) classes to perform tasks before a message is sent.*

- virtual void **setBodyBytes** (const unsigned char *buffer, int numBytes)
sets the bytes given to the message body.
- virtual unsigned char * **getBodyBytes** () const
Gets the bytes that are contained in this message and returns them in a newly allocated array that becomes the property of the caller.
- virtual int **getBodyLength** () const
Returns the number of bytes contained in the body of this message.
- virtual void **reset** ()
Puts the message body in read-only mode and repositions the stream of bytes to the beginning.
- virtual bool **readBoolean** () const
Reads a Boolean from the Bytes message stream.
- virtual void **writeBoolean** (bool value)
Writes a boolean to the bytes message stream as a 1-byte value.
- virtual unsigned char **readByte** () const
Reads a Byte from the Bytes message stream.
- virtual void **writeByte** (unsigned char value)
Writes a byte to the bytes message stream as a 1-byte value.
- virtual int **readBytes** (std::vector< unsigned char > &value) const
Reads a byte array from the bytes message stream.
- virtual void **writeBytes** (const std::vector< unsigned char > &value)
Writes a byte array to the bytes message stream using the vector size as the number of bytes to write.
- virtual int **readBytes** (unsigned char *buffer, int length) const
Reads a portion of the bytes message stream.
- virtual void **writeBytes** (const unsigned char *value, int offset, int length)
Writes a portion of a byte array to the bytes message stream.
- virtual char **readChar** () const
Reads a Char from the Bytes message stream.
- virtual void **writeChar** (char value)
Writes a char to the bytes message stream as a 1-byte value.
- virtual float **readFloat** () const
Reads a 32 bit float from the Bytes message stream.
- virtual void **writeFloat** (float value)
Writes a float to the bytes message stream as a 4 byte value.
- virtual double **readDouble** () const
Reads a 64 bit double from the Bytes message stream.
- virtual void **writeDouble** (double value)
Writes a double to the bytes message stream as a 8 byte value.
- virtual short **readShort** () const
Reads a 16 bit signed short from the Bytes message stream.
- virtual void **writeShort** (short value)
Writes a signed short to the bytes message stream as a 2 byte value.
- virtual unsigned short **readUnsignedShort** () const

Reads a 16 bit unsigned short from the Bytes message stream.

- virtual void **writeUnsignedShort** (unsigned short value)

Writes a unsigned short to the bytes message stream as a 2 byte value.

- virtual int **readInt** () const

Reads a 32 bit signed integer from the Bytes message stream.

- virtual void **writeInt** (int value)

Writes a signed int to the bytes message stream as a 4 byte value.

- virtual long long **readLong** () const

Reads a 64 bit long from the Bytes message stream.

- virtual void **writeLong** (long long value)

Writes a long long to the bytes message stream as a 8 byte value.

- virtual std::string **readString** () const

Reads an ASCII String from the Bytes message stream.

- virtual void **writeString** (const std::string &value)

Writes an ASCII String to the Bytes message stream.

- virtual std::string **readUTF** () const

Reads an UTF String from the BytesMessage stream.

- virtual void **writeUTF** (const std::string &value)

Writes an UTF String to the BytesMessage stream.

Static Public Attributes

- static const unsigned char **ID_ACTIVEMQBYTESMESSAGE** = 24

6.14.1 Constructor & Destructor Documentation

6.14.1.1 **activemq::commands::ActiveMQBytesMessage::ActiveMQBytesMessage ()**

6.14.1.2 **virtual activemq::commands::ActiveMQBytesMessage::~~ActiveMQBytesMessage () throw ()** *[virtual]*

6.14.2 Member Function Documentation

6.14.2.1 **virtual void activemq::commands::ActiveMQBytesMessage::clearBody ()**
[virtual]

Clears out the body of the message.

This does not clear the headers or properties.

Exceptions

<i>CMSException</i>	- if an internal error occurs.
---------------------	--------------------------------

Reimplemented from **activemq::commands::ActiveMQMessageTemplate< cms::-**

BytesMessage > (p. 296).

6.14.2.2 `virtual cms::BytesMessage* activemq::commands::ActiveMQBytesMessage::clone () const [inline, virtual]`

Clones this message.

Returns

a deep copy of this message.

Exceptions

<i>CMSException</i>	- if an internal error occurs while cloning the Message (p. 1821).
---------------------	---

Implements **cms::BytesMessage** (p. 720).

6.14.2.3 `virtual ActiveMQBytesMessage* activemq::commands::ActiveMQBytesMessage::cloneDataStructure () const [virtual]`

Clone this object and return a new instance that the caller now owns, this will be an exact copy of this one.

Returns

new copy of this object.

Reimplemented from **activemq::commands::Message** (p. 1826).

6.14.2.4 `virtual void activemq::commands::ActiveMQBytesMessage::copyDataStructure (const DataStructure * src) [virtual]`

Copy the contents of the passed object into this objects members, overwriting any existing data.

Returns

src - Source Object

Reimplemented from **activemq::commands::Message** (p. 1827).

6.14.2.5 `virtual bool activemq::commands::ActiveMQBytesMessage::equals (const DataStructure * value) const [virtual]`

Compares the **DataStructure** (p. 1133) passed in to this one, and returns if they are equivalent.

Equivalent here means that they are of the same type, and that each element of the objects are the same.

Returns

true if **DataStructure** (p. 1133)'s are Equal.

Reimplemented from **activemq::commands::ActiveMQMessageTemplate**< **cms::BytesMessage** > (p. 297).

6.14.2.6 `virtual unsigned char* activemq::commands::ActiveMQBytesMessage::getBodyBytes () const [virtual]`

Gets the bytes that are contained in this message and returns them in a newly allocated array that becomes the property of the caller.

This is a copy of the data contained in this message, changing the value contained in this array has no effect on the data contained in this message.

Returns

pointer to a byte buffer that the call owns upon completion of this method.

Exceptions

<i>CMSEException</i>	- If an internal error occurs.
<i>MessageNotReadableException</i>	- If the message is in Write Only Mode.

Implements **cms::BytesMessage** (p. 721).

6.14.2.7 `virtual int activemq::commands::ActiveMQBytesMessage::getBodyLength () const [virtual]`

Returns the number of bytes contained in the body of this message.

Returns

number of bytes.

Exceptions

<i>CMSException</i>	- If an internal error occurs.
<i>MessageNotReadableException</i>	- If the message is in Write Only Mode.

Implements **cms::BytesMessage** (p. 721).

6.14.2.8 virtual unsigned char **activemq::commands::ActiveMQBytesMessage::getDataStructureType** () const
[virtual]

Get the **DataStructure** (p. 1133) Type as defined in CommandTypes.h.

Returns

The type of the data structure

Reimplemented from **activemq::commands::Message** (p. 1829).

6.14.2.9 virtual void **activemq::commands::ActiveMQBytesMessage::onSend** ()
[virtual]

Allows derived **Message** (p. 1821) classes to perform tasks before a message is sent.

Reimplemented from **activemq::commands::ActiveMQMessageTemplate< cms::BytesMessage >** (p. 299).

6.14.2.10 virtual bool **activemq::commands::ActiveMQBytesMessage::readBoolean** () const [virtual]

Reads a Boolean from the Bytes message stream.

Returns

boolean value from stream

Exceptions

<i>CMSException</i>	- if the CMS provider fails to read the message due to some internal error.
<i>MessageEOFException</i>	- if unexpected end of bytes stream has been reached.
<i>MessageNotReadableException</i>	- if the message is in write-only mode.

Implements **cms::BytesMessage** (p. 722).

6.14.2.11 `virtual unsigned char activemq::commands::ActiveMQBytesMessage::readByte () const` `[virtual]`

Reads a Byte from the Bytes message stream.

Returns

unsigned char value from stream

Exceptions

<i>CMSEException</i>	- if the CMS provider fails to read the message due to some internal error.
<i>MessageEOFException</i>	- if unexpected end of bytes stream has been reached.
<i>MessageNotReadableException</i>	- if the message is in write-only mode.

Implements `cms::BytesMessage` (p. 722).

6.14.2.12 `virtual int activemq::commands::ActiveMQBytesMessage::readBytes (std::vector< unsigned char > & value) const` `[virtual]`

Reads a byte array from the bytes message stream.

If the length of vector value is less than the number of bytes remaining to be read from the stream, the vector should be filled. A subsequent call reads the next increment, and so on.

If the number of bytes remaining in the stream is less than the length of vector value, the bytes should be read into the vector. The return value of the total number of bytes read will be less than the length of the vector, indicating that there are no more bytes left to be read from the stream. The next read of the stream returns -1.

Parameters

<i>value</i>	buffer to place data in
--------------	-------------------------

Returns

the total number of bytes read into the buffer, or -1 if there is no more data because the end of the stream has been reached

Exceptions

<i>CMSEException</i>	- if the CMS provider fails to read the message due to some internal error.
<i>MessageEOFException</i>	- if unexpected end of bytes stream has been reached.
<i>MessageNotReadableException</i>	- if the message is in write-only mode.

Implements **cms::BytesMessage** (p. 723).

6.14.2.13 `virtual int activemq::commands::ActiveMQBytesMessage::readBytes (unsigned char * buffer, int length) const` `[virtual]`

Reads a portion of the bytes message stream.

If the length of array value is less than the number of bytes remaining to be read from the stream, the array should be filled. A subsequent call reads the next increment, and so on.

If the number of bytes remaining in the stream is less than the length of array value, the bytes should be read into the array. The return value of the total number of bytes read will be less than the length of the array, indicating that there are no more bytes left to be read from the stream. The next read of the stream returns -1.

If length is negative, or length is greater than the length of the array value, then an `IndexOutOfBoundsException` is thrown. No bytes will be read from the stream for this exception case.

Parameters

<i>buffer</i>	the buffer into which the data is read
<i>length</i>	the number of bytes to read; must be less than or equal to value.length

Returns

the total number of bytes read into the buffer, or -1 if there is no more data because the end of the stream has been reached

Exceptions

<i>CMSException</i>	- if the CMS provider fails to read the message due to some internal error.
<i>MessageEOF-Exception</i>	- if unexpected end of bytes stream has been reached.
<i>MessageNot-ReadableException</i>	- if the message is in write-only mode.

Implements **cms::BytesMessage** (p. 723).

6.14.2.14 `virtual char activemq::commands::ActiveMQBytesMessage::readChar () const` `[virtual]`

Reads a Char from the Bytes message stream.

Returns

char value from stream

Exceptions

<i>CMSException</i>	- if the CMS provider fails to read the message due to some internal error.
<i>MessageEOF-Exception</i>	- if unexpected end of bytes stream has been reached.
<i>MessageNot-ReadableException</i>	- if the message is in write-only mode.

Implements **cms::BytesMessage** (p. 724).

6.14.2.15 virtual double **activemq::commands::ActiveMQBytesMessage::readDouble () const** [virtual]

Reads a 64 bit double from the Bytes message stream.

Returns

double value from stream

Exceptions

<i>CMSException</i>	- if the CMS provider fails to read the message due to some internal error.
<i>MessageEOF-Exception</i>	- if unexpected end of bytes stream has been reached.
<i>MessageNot-ReadableException</i>	- if the message is in write-only mode.

Implements **cms::BytesMessage** (p. 725).

6.14.2.16 virtual float **activemq::commands::ActiveMQBytesMessage::readFloat () const** [virtual]

Reads a 32 bit float from the Bytes message stream.

Returns

double value from stream

Exceptions

<i>CMSException</i>	- if the CMS provider fails to read the message due to some internal error.
<i>MessageEOF-Exception</i>	- if unexpected end of bytes stream has been reached.
<i>MessageNot-ReadableException</i>	- if the message is in write-only mode.

Implements **cms::BytesMessage** (p. 725).

6.14.2.17 `virtual int activemq::commands::ActiveMQBytesMessage::readInt ()
const [virtual]`

Reads a 32 bit signed integer from the Bytes message stream.

Returns

int value from stream

Exceptions

<i>CMSException</i>	- if the CMS provider fails to read the message due to some internal error.
<i>MessageEOF-Exception</i>	- if unexpected end of bytes stream has been reached.
<i>MessageNot-ReadableException</i>	- if the message is in write-only mode.

Implements **cms::BytesMessage** (p. 725).

6.14.2.18 `virtual long long activemq::commands::ActiveMQBytesMessage::read-
Long () const [virtual]`

Reads a 64 bit long from the Bytes message stream.

Returns

long long value from stream

Exceptions

<i>CMSException</i>	- if the CMS provider fails to read the message due to some internal error.
<i>MessageEOF-Exception</i>	- if unexpected end of bytes stream has been reached.
<i>MessageNot-ReadableException</i>	- if the message is in write-only mode.

Implements **cms::BytesMessage** (p. 726).

6.14.2.19 `virtual short activemq::commands::ActiveMQBytesMessage::readShort () const [virtual]`

Reads a 16 bit signed short from the Bytes message stream.

Returns

short value from stream

Exceptions

<i>CMSException</i>	- if the CMS provider fails to read the message due to some internal error.
<i>MessageEOF-Exception</i>	- if unexpected end of bytes stream has been reached.
<i>MessageNot-ReadableException</i>	- if the message is in write-only mode.

Implements **cms::BytesMessage** (p. 726).

6.14.2.20 `virtual std::string activemq::commands::ActiveMQBytesMessage::readString () const [virtual]`

Reads an ASCII String from the Bytes message stream.

Returns

String from stream

Exceptions

<i>CMSException</i>	- if the CMS provider fails to read the message due to some internal error.
<i>MessageEOF-Exception</i>	- if unexpected end of bytes stream has been reached.
<i>MessageNot-ReadableException</i>	- if the message is in write-only mode.

Implements **cms::BytesMessage** (p. 727).

6.14.2.21 `virtual unsigned short activemq::commands::ActiveMQBytesMessage::readUnsignedShort () const [virtual]`

Reads a 16 bit unsigned short from the Bytes message stream.

Returns

unsigned short value from stream

Exceptions

<i>CMSException</i>	- if the CMS provider fails to read the message due to some internal error.
<i>MessageEOF-Exception</i>	- if unexpected end of bytes stream has been reached.
<i>MessageNot-ReadableException</i>	- if the message is in write-only mode.

Implements **cms::BytesMessage** (p. 727).

6.14.2.22 virtual std::string activemq::commands::ActiveMQBytesMessage::readUTF() const [virtual]

Reads an UTF String from the BytesMessage stream.

Returns

String from stream

Exceptions

<i>CMSException</i>	- if the CMS provider fails to read the message due to some internal error.
<i>MessageEOF-Exception</i>	- if unexpected end of bytes stream has been reached.
<i>MessageNot-ReadableException</i>	- if the message is in write-only mode.

Implements **cms::BytesMessage** (p. 728).

6.14.2.23 virtual void activemq::commands::ActiveMQBytesMessage::reset() [virtual]

Puts the message body in read-only mode and repositions the stream of bytes to the beginning.

Exceptions

<i>CMSException</i>	- If the provider fails to perform the reset operation.
<i>MessageFormatException</i>	- If the Message (p. 1821) has an invalid format.

Implements **cms::BytesMessage** (p. 728).

6.14.2.24 `virtual void activemq::commands::ActiveMQBytesMessage::setBodyBytes (const unsigned char * buffer, int numBytes)`
`[virtual]`

sets the bytes given to the message body.

Parameters

<i>buffer</i>	Byte Buffer to copy
<i>numBytes</i>	Number of bytes in Buffer to copy

Exceptions

<i>CMSException</i>	- If an internal error occurs.
<i>MessageNotWriteableException</i>	- if in Read Only Mode.

Implements **cms::BytesMessage** (p. 729).

6.14.2.25 `virtual std::string activemq::commands::ActiveMQBytesMessage::toString () const` `[virtual]`

Returns a string containing the information for this **DataStructure** (p. 1133) such as its type and value of its elements.

Returns

formatted string useful for debugging.

Reimplemented from **activemq::commands::Message** (p. 1836).

6.14.2.26 `virtual void activemq::commands::ActiveMQBytesMessage::writeBoolean (bool value)` `[virtual]`

Writes a boolean to the bytes message stream as a 1-byte value.

The value true is written as the value (byte)1; the value false is written as the value (byte)0.

Parameters

<i>value</i>	boolean to write to the stream
--------------	--------------------------------

Exceptions

<i>CMSException</i>	- if the CMS provider fails to write the message due to some internal error.
<i>MessageNotWriteableException</i>	- if the message is in read-only mode.

Implements **cms::BytesMessage** (p. 729).

6.14.2.27 `virtual void activemq::commands::ActiveMQBytesMessage::writeByte (unsigned char value) [virtual]`

Writes a byte to the bytes message stream as a 1-byte value.

Parameters

<i>value</i>	byte to write to the stream
--------------	-----------------------------

Exceptions

<i>CMSEException</i>	- if the CMS provider fails to write the message due to some internal error.
<i>MessageNotWritableException</i>	- if the message is in read-only mode.

Implements **cms::BytesMessage** (p. 729).

6.14.2.28 `virtual void activemq::commands::ActiveMQBytesMessage::writeBytes (const std::vector< unsigned char > & value) [virtual]`

Writes a byte array to the bytes message stream using the vector size as the number of bytes to write.

Parameters

<i>value</i>	bytes to write to the stream
--------------	------------------------------

Exceptions

<i>CMSEException</i>	- if the CMS provider fails to write the message due to some internal error.
<i>MessageNotWritableException</i>	- if the message is in read-only mode.

Implements **cms::BytesMessage** (p. 730).

6.14.2.29 `virtual void activemq::commands::ActiveMQBytesMessage::writeBytes (const unsigned char * value, int offset, int length) [virtual]`

Writes a portion of a byte array to the bytes message stream.

size as the number of bytes to write.

Parameters

<i>value</i>	bytes to write to the stream
<i>offset</i>	the initial offset within the byte array
<i>length</i>	the number of bytes to use

Exceptions

<i>CMSException</i>	- if the CMS provider fails to write the message due to some internal error.
<i>MessageNot-WriteableException</i>	- if the message is in read-only mode.

Implements **cms::BytesMessage** (p. 730).

6.14.2.30 `virtual void activemq::commands::ActiveMQBytesMessage::writeChar (char value) [virtual]`

Writes a char to the bytes message stream as a 1-byte value.

Parameters

<i>value</i>	char to write to the stream
--------------	-----------------------------

Exceptions

<i>CMSException</i>	- if the CMS provider fails to write the message due to some internal error.
<i>MessageNot-WriteableException</i>	- if the message is in read-only mode.

Implements **cms::BytesMessage** (p. 731).

6.14.2.31 `virtual void activemq::commands::ActiveMQBytesMessage::writeDouble (double value) [virtual]`

Writes a double to the bytes message stream as a 8 byte value.

Parameters

<i>value</i>	double to write to the stream
--------------	-------------------------------

Exceptions

<i>CMSException</i>	- if the CMS provider fails to write the message due to some internal error.
<i>MessageNot-WriteableException</i>	- if the message is in read-only mode.

Implements **cms::BytesMessage** (p. 731).

6.14.2.32 `virtual void activemq::commands::ActiveMQBytesMessage::writeFloat (float value) [virtual]`

Writes a float to the bytes message stream as a 4 byte value.

Parameters

<i>value</i>	float to write to the stream
--------------	------------------------------

Exceptions

<i>CMSException</i>	- if the CMS provider fails to write the message due to some internal error.
<i>MessageNotWritableException</i>	- if the message is in read-only mode.

Implements **cms::BytesMessage** (p. 732).

6.14.2.33 `virtual void activemq::commands::ActiveMQBytesMessage::writeInt (int value) [virtual]`

Writes a signed int to the bytes message stream as a 4 byte value.

Parameters

<i>value</i>	signed int to write to the stream
--------------	-----------------------------------

Exceptions

<i>CMSException</i>	- if the CMS provider fails to write the message due to some internal error.
<i>MessageNotWritableException</i>	- if the message is in read-only mode.

Implements **cms::BytesMessage** (p. 732).

6.14.2.34 `virtual void activemq::commands::ActiveMQBytesMessage::writeLong (long long value) [virtual]`

Writes a long long to the bytes message stream as a 8 byte value.

Parameters

<i>value</i>	signed long long to write to the stream
--------------	---

Exceptions

<i>CMSException</i>	- if the CMS provider fails to write the message due to some internal error.
<i>MessageNot-WriteableException</i>	- if the message is in read-only mode.

Implements **cms::BytesMessage** (p. 732).

6.14.2.35 `virtual void activemq::commands::ActiveMQBytesMessage::writeShort (short value) [virtual]`

Writes a signed short to the bytes message stream as a 2 byte value.

Parameters

<i>value</i>	signed short to write to the stream
--------------	-------------------------------------

Exceptions

<i>CMSException</i>	- if the CMS provider fails to write the message due to some internal error.
<i>MessageNot-WriteableException</i>	- if the message is in read-only mode.

Implements **cms::BytesMessage** (p. 733).

6.14.2.36 `virtual void activemq::commands::ActiveMQBytesMessage::writeString (const std::string & value) [virtual]`

Writes an ASCII String to the Bytes message stream.

Parameters

<i>value</i>	String to write to the stream
--------------	-------------------------------

Exceptions

<i>CMSException</i>	- if the CMS provider fails to write the message due to some internal error.
<i>MessageNot-WriteableException</i>	- if the message is in read-only mode.

Implements **cms::BytesMessage** (p. 733).

6.14.2.37 `virtual void activemq::commands::ActiveMQBytesMessage::writeUnsignedShort (unsigned short value)`
`[virtual]`

Writes a unsigned short to the bytes message stream as a 2 byte value.

Parameters

<i>value</i>	unsigned short to write to the stream
--------------	---------------------------------------

Exceptions

<i>CMSException</i>	- if the CMS provider fails to write the message due to some internal error.
<i>MessageNotWritableException</i>	- if the message is in read-only mode.

Implements **cms::BytesMessage** (p. 734).

6.14.2.38 `virtual void activemq::commands::ActiveMQBytesMessage::writeUTF (const std::string & value)`
`[virtual]`

Writes an UTF String to the BytesMessage stream.

Parameters

<i>value</i>	String to write to the stream
--------------	-------------------------------

Exceptions

<i>CMSException</i>	- if the CMS provider fails to read the message due to some internal error.
<i>MessageNotWritableException</i>	- if the message is in read-only mode.

Implements **cms::BytesMessage** (p. 734).

6.14.3 Field Documentation

6.14.3.1 `const unsigned char activemq::commands::ActiveMQBytesMessage::ID_ACTIVEMQBYTESMESSAGE = 24`
`[static]`

The documentation for this class was generated from the following file:

- `src/main/activemq/commands/ActiveMQBytesMessage.h`

6.15 activemq::wireformat::openwire::marshal::generated::ActiveMQBytesMessageMarshaller Class Reference

Marshaling code for Open Wire Format for **ActiveMQBytesMessageMarshaller** (p. 183).

```
#include <src/main/activemq/wireformat/openwire/marshal/generated/ActiveMQBytesMessageMarshaller.h>
```

Inheritance diagram for activemq::wireformat::openwire::marshal::generated::ActiveMQBytesMessageMarshaller:

Public Member Functions

- **ActiveMQBytesMessageMarshaller** ()
- virtual **~ActiveMQBytesMessageMarshaller** ()
- virtual **commands::DataStructure * createObject** () const
Creates a new instance of the class that this class is a marshaling director for.
- virtual unsigned char **getDataStructureType** () const
Gets the DataStructureType that this class marshals/unmarshals.
- virtual void **tightUnmarshal** (**OpenWireFormat *wireFormat**, **commands::DataStructure *dataStructure**, **decaf::io::DataInputStream *dataIn**, **utils::BooleanStream *bs**)
Tight Un-marhsal to the given stream.
- virtual int **tightMarshal1** (**OpenWireFormat *wireFormat**, **commands::DataStructure *dataStructure**, **utils::BooleanStream *bs**)
Tight Marhsal to the given stream.
- virtual void **tightMarshal2** (**OpenWireFormat *wireFormat**, **commands::DataStructure *dataStructure**, **decaf::io::DataOutputStream *dataOut**, **utils::BooleanStream *bs**)
Tight Marhsal to the given stream.
- virtual void **looseUnmarshal** (**OpenWireFormat *wireFormat**, **commands::DataStructure *dataStructure**, **decaf::io::DataInputStream *dataIn**)
Loose Un-marhsal to the given stream.
- virtual void **looseMarshal** (**OpenWireFormat *wireFormat**, **commands::DataStructure *dataStructure**, **decaf::io::DataOutputStream *dataOut**)
Tight Marhsal to the given stream.

6.15.1 Detailed Description

Marshaling code for Open Wire Format for **ActiveMQBytesMessageMarshaller** (p. 183).

NOTE!: This file is auto generated - do not modify! if you need to make a change, please see the Java Classes in the activemq-openwire-generator module

6.15 activemq::wireformat::openwire::marshal::generated::ActiveMQBytesMessageMarshaller Class

Reference

185

6.15.2 Constructor & Destructor Documentation

6.15.2.1 **activemq::wireformat::openwire::marshal::generated::ActiveMQBytesMessageMarshaller::ActiveMQBytesMessageMarshaller ()**
[inline]

6.15.2.2 **virtual activemq::wireformat::openwire::marshal::generated::ActiveMQBytesMessageMarshaller::~~ActiveMQBytesMessageMarshaller ()**
[inline, virtual]

6.15.3 Member Function Documentation

6.15.3.1 **virtual commands::DataStructure* activemq::wireformat::openwire::marshal::generated::ActiveMQBytesMessageMarshaller::createObject () const** [virtual]

Creates a new instance of the class that this class is a marshaling director for.

Returns

newly allocated Command

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1120).

6.15.3.2 **virtual unsigned char activemq::wireformat::openwire::marshal::generated::ActiveMQBytesMessageMarshaller::getDataStructureType () const**
[virtual]

Gets the DataStructureType that this class marshals/unmarshals.

Returns

byte Id of this classes DataStructureType

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1122).

6.15.3.3 **virtual void activemq::wireformat::openwire::marshal::generated::ActiveMQBytesMessageMarshaller::looseMarshal (OpenWireFormat * *format*, commands::DataStructure * *command*, decaf::io::DataOutputStream * *ds*)** [virtual]

Tight Marshal to the given stream.

Parameters

<i>format</i>	- The OpenWireFormat properties
<i>command</i>	- the object to Marshal
<i>ds</i>	- DataOutputStream to marshal to

Exceptions

<i>IOException</i>	if an error occurs.
--------------------	---------------------

Reimplemented from **activemq::wireformat::openwire::marshal::generated::MessageMarshaller** (p. 1919).

6.15.3.4 **virtual void activemq::wireformat::openwire::marshal::generated::ActiveMQBytesMessageMarshaller::looseUnmarshal (OpenWireFormat * format, commands::DataStructure * command, decaf::io::DataInputStream * dis)** [virtual]

Loose Un-marhsal to the given stream.

Parameters

<i>format</i>	- The OpenWireFormat properties
<i>command</i>	- the object to Un-Marshal
<i>dis</i>	- the DataInputStream to Un-Marshal from

Exceptions

<i>IOException</i>	if an error occurs.
--------------------	---------------------

Reimplemented from **activemq::wireformat::openwire::marshal::generated::MessageMarshaller** (p. 1919).

6.15.3.5 **virtual int activemq::wireformat::openwire::marshal::generated::ActiveMQBytesMessageMarshaller::tightMarshal1 (OpenWireFormat * format, commands::DataStructure * command, utils::BooleanStream * bs)** [virtual]

Tight Marhsal to the given stream.

Parameters

<i>format</i>	The OpenWireFormat properties
<i>command</i>	The object to Marshal
<i>bs</i>	The boolean stream to marshal to.

Exceptions

<i>IOException</i>	if an error occurs.
--------------------	---------------------

Reimplemented from **activemq::wireformat::openwire::marshal::generated::MessageMarshaller** (p. 1920).

6.15 activemq::wireformat::openwire::marshal::generated::ActiveMQBytes-MessageMarshaller Class

Reference

187

6.15.3.6 virtual void activemq::wireformat::openwire::marshal::generated::ActiveMQBytesMessageMarshaller::tightMarshal2 (OpenWireFormat * *format*, commands::DataStructure * *command*, decaf::io::DataOutputStream * *ds*, utils::BooleanStream * *bs*) [virtual]

Tight Marhsal to the given stream.

Parameters

<i>format</i>	- The OpenWireFormat properties
<i>command</i>	- the object to Marshal
<i>ds</i>	- the DataOutputStream to Marshal to
<i>bs</i>	- boolean stream to marshal to.

Exceptions

<i>IOException</i>	if an error occurs.
--------------------	---------------------

Reimplemented from **activemq::wireformat::openwire::marshal::generated::MessageMarshaller** (p. 1921).

6.15.3.7 virtual void activemq::wireformat::openwire::marshal::generated::ActiveMQBytesMessageMarshaller::tightUnmarshal (OpenWireFormat * *format*, commands::DataStructure * *command*, decaf::io::DataInputStream * *dis*, utils::BooleanStream * *bs*) [virtual]

Tight Un-marhsal to the given stream.

Parameters

<i>format</i>	- The OpenWireFormat properties
<i>command</i>	- the object to Un-Marshal
<i>dis</i>	- the DataInputStream to Un-Marshal from
<i>bs</i>	- boolean stream to unmarshal from.

Exceptions

<i>IOException</i>	if an error occurs.
--------------------	---------------------

Reimplemented from **activemq::wireformat::openwire::marshal::generated::MessageMarshaller** (p. 1921).

The documentation for this class was generated from the following file:

- src/main/activemq/wireformat/openwire/marshal/generated/**ActiveMQBytes-MessageMarshaller.h**

6.16 activemq::core::ActiveMQConnection Class Reference

Concrete connection used for all connectors to the ActiveMQ broker.

```
#include <src/main/activemq/core/ActiveMQConnection.h>
```

Inheritance diagram for activemq::core::ActiveMQConnection:

Public Member Functions

- **ActiveMQConnection** (const **Pointer**< **transport::Transport** > &transport, const **Pointer**< **decaf::util::Properties** > &properties)
Constructor.
- virtual **~ActiveMQConnection** () throw ()
- virtual void **addSession** (**ActiveMQSession** *session)
Adds the session resources for the given session instance.
- virtual void **removeSession** (**ActiveMQSession** *session)
Removes the session resources for the given session instance.
- virtual void **addProducer** (**ActiveMQProducer** *producer)
Adds an active Producer to the Set of known producers.
- virtual void **removeProducer** (const **Pointer**< **commands::ProducerId** > &producerId)
Removes an active Producer to the Set of known producers.
- virtual void **addDispatcher** (const **Pointer**< **commands::ConsumerId** > &consumer, **Dispatcher** *dispatcher)
Adds a dispatcher for a consumer.
- virtual void **removeDispatcher** (const **Pointer**< **commands::ConsumerId** > &consumer)
Removes the dispatcher for a consumer.
- virtual void **sendPullRequest** (const **commands::ConsumerInfo** *consumer, long long timeout)
If supported sends a message pull request to the service provider asking for the delivery of a new message.
- bool **isClosed** () const
Checks if this connection has been closed.
- bool **isStarted** () const
Check if this connection has been started.
- bool **isTransportFailed** () const
Checks if the Connection's Transport has failed.
- virtual void **destroyDestination** (const **commands::ActiveMQDestination** *destination)
Requests that the Broker removes the given Destination.
- virtual void **destroyDestination** (const **cms::Destination** *destination)

Requests that the Broker removes the given Destination.

- virtual const **cms::ConnectionMetaData** * **getMetaData** () const

Gets the metadata for this connection.

Returns

the connection MetaData pointer (caller does not own it).

Exceptions

CMSEException (p. 826)	<i>if the provider fails to get the connection metadata for this connection.</i>
-------------------------------	--

See also

ConnectionMetaData (p. 978)

Since

2.0

- virtual **cms::Session** * **createSession** ()

*Creates an AUTO_ACKNOWLEDGE **Session** (p. 2361).*

Exceptions

CMSEException (p. 826)	
-------------------------------	--

- virtual std::string **getClientID** () const

Get the Client Id for this session, the client Id is provider specific and is either assigned by the connection factory or set using the setClientID method.

Returns

*Client Id String for this **Connection** (p. 933).*

Exceptions

CMSEException (p. 826)	<i>if the provider fails to return the client id or an internal error occurs.</i>
-------------------------------	---

- virtual void **setClientID** (const std::string &clientID)

Sets the client identifier for this connection.

*The preferred way to assign a CMS client's client identifier is for it to be configured in a client-specific **ConnectionFactory** (p. 955) object and transparently assigned to the **Connection** (p. 933) object it creates.*

*If a client sets the client identifier explicitly, it must do so immediately after it creates the connection and before any other action on the connection is taken. After this point, setting the client identifier is a programming error that should throw an **IllegalStateException** (p. 1419).*

Parameters

clientID	<i>The unique client identifier to assign to the Connection (p. 933).</i>
----------	--

Exceptions

CMSEException (p. 826)	<i>if the provider fails to set the client id due to some internal error.</i>
InvalidClientID-Exception	<i>if the id given is somehow invalid or is a duplicate.</i>
IllegalStateException (p. 1419)	<i>if the client tries to set the id after a Connection (p. 933) method has been called.</i>

- virtual **cms::Session** * **createSession** (**cms::Session::AcknowledgeMode** ackMode)

*Creates a new **Session** (p. 2361) to work for this **Connection** (p. 933) using the spec-*

ified acknowledgment mode.

Parameters

ackMode	<i>the Acknowledgment Mode to use.</i>
---------	--

Exceptions

CMSEException (p. 826)

- virtual void **close** ()

Closes this connection as well as any Sessions created from it (and those Sessions' consumers and producers).

Exceptions

CMSEException (p. 826)

- virtual void **start** ()

Starts the service.

Exceptions

CMSEException (p. 826)	<i>if an internal error occurs while starting.</i>
-------------------------------	--

- virtual void **stop** ()

Stops this service.

Exceptions

CMSEException (p. 826)	<i>- if an internal error occurs while stopping the Service.</i>
-------------------------------	--

- virtual **cms::ExceptionListener * getExceptionListener** () const

Gets the registered Exception Listener for this connection.

Returns

pointer to an exception listener or NULL

- virtual void **setExceptionListener** (**cms::ExceptionListener *listener**)

Sets the registered Exception Listener for this connection.

Parameters

listener	<i>pointer to and ExceptionListener (p. 1286)</i>
----------	--

- void **setUsername** (const std::string &username)

Sets the username that should be used when creating a new connection.

- const std::string & **getUsername** () const

Gets the username that this factory will use when creating a new connection instance.

- void **setPassword** (const std::string &password)

Sets the password that should be used when creating a new connection.

- const std::string & **getPassword** () const

Gets the password that this factory will use when creating a new connection instance.

- void **setDefaultClientId** (const std::string &clientId)

Sets the Client Id.

- void **setBrokerURL** (const std::string &brokerURL)

Sets the Broker URL that should be used when creating a new connection instance.

- const std::string & **getBrokerURL** () const

Gets the Broker URL that this factory will use when creating a new connection instance.

- void **setPrefetchPolicy** (**PrefetchPolicy** *policy)

*Sets the **PrefetchPolicy** (p. 2110) instance that this factory should use when it creates new Connection instances.*

- **PrefetchPolicy** * **getPrefetchPolicy** () const

*Gets the pointer to the current **PrefetchPolicy** (p.2110) that is in use by this - ConnectionFactory.*

- void **setRedeliveryPolicy** (**RedeliveryPolicy** *policy)

*Sets the **RedeliveryPolicy** (p. 2250) instance that this factory should use when it creates new Connection instances.*

- **RedeliveryPolicy** * **getRedeliveryPolicy** () const

*Gets the pointer to the current **RedeliveryPolicy** (p. 2250) that is in use by this - ConnectionFactory.*

- bool **isDispatchAsync** () const

- void **setDispatchAsync** (bool value)

Should messages be dispatched synchronously or asynchronously from the producer thread for non-durable topics in the broker? For fast consumers set this to false.

- bool **isAlwaysSyncSend** () const

Gets if the Connection should always send things Synchronously.

- void **setAlwaysSyncSend** (bool value)

Sets if the Connection should always send things Synchronously.

- bool **isUseAsyncSend** () const

Gets if the useAsyncSend option is set.

- void **setUseAsyncSend** (bool value)

Sets the useAsyncSend option.

- bool **isUseCompression** () const

Gets if the Connection is configured for Message body compression.

- void **setUseCompression** (bool value)

Sets whether Message body compression is enabled.

- void **setCompressionLevel** (int value)

Sets the Compression level used when Message body compression is enabled, a value of -1 causes the Compression Library to use the default setting which is a balance of speed and compression.

- int **getCompressionLevel** () const

Gets the currently configured Compression level for Message bodies.

- unsigned int **getSendTimeout** () const

Gets the assigned send timeout for this Connector.

- void **setSendTimeout** (unsigned int timeout)

Sets the send timeout to use when sending Message objects, this will cause all messages to be sent using a Synchronous request is non-zero.

- unsigned int **getCloseTimeout** () const

Gets the assigned close timeout for this Connector.

- void **setCloseTimeout** (unsigned int timeout)

Sets the close timeout to use when sending the disconnect request.

- unsigned int **getProducerWindowSize** () const
Gets the configured producer window size for Producers that are created from this connector.
- void **setProducerWindowSize** (unsigned int windowSize)
Sets the size in Bytes of messages that a producer can send before it is blocked to await a ProducerAck from the broker that frees enough memory to allow another message to be sent.
- bool **isMessagePrioritySupported** () const
- void **setMessagePrioritySupported** (bool value)
Set whether or not this factory should create Connection objects with the Message priority support function enabled.
- long long **getNextTempDestinationId** ()
Get the Next Temporary Destination Id.
- long long **getNextLocalTransactionId** ()
Get the Next Temporary Destination Id.
- void **addTransportListener** (transport::TransportListener *transport-Listener)
Adds a transport listener so that a client can be notified of events in the underlying transport, client's are always notified after the event has been processed by the - Connection class.
- void **removeTransportListener** (transport::TransportListener *transport-Listener)
Removes a registered TransportListener from the Connection's set of Transport listeners, this listener will no longer receive any Transport related events.
- virtual void **onCommand** (const Pointer< commands::Command > &command)
Event handler for the receipt of a non-response command from the transport.
- virtual void **onException** (const decaf::lang::Exception &ex)
Event handler for an exception from a command transport.
- virtual void **transportInterrupted** ()
The transport has suffered an interruption from which it hopes to recover.
- virtual void **transportResumed** ()
The transport has resumed after an interruption.
- const commands::ConnectionInfo & **getConnectionInfo** () const
Gets the ConnectionInfo for this Object, if the Connection is not open than this method throws an exception.
- const commands::ConnectionId & **getConnectionId** () const
Gets the ConnectionId for this Object, if the Connection is not open than this method throws an exception.
- transport::Transport & **getTransport** () const
Gets a reference to this object's Transport instance.
- Pointer< threads::Scheduler > **getScheduler** () const
Gets a reference to the Connection objects built in Scheduler instance.
- std::string **getResourceManagerId** () const
Returns the Id of the Resource Manager that this client will use should it be entered into an XA Transaction.

- void **cleanup** ()
*Clean up this connection object, resetting it back to a state that mirrors what a newly created **ActiveMQConnection** (p. 187) object has.*
- void **oneway** (**Pointer**< **commands::Command** > command)
Sends a message without request that the broker send a response to indicate that it was received.
- **Pointer**< **commands::Response** > **syncRequest** (**Pointer**< **commands::Command** > command, unsigned int timeout=0)
Sends a synchronous request and returns the response from the broker.
- virtual void **fire** (const **exceptions::ActiveMQException** &ex)
Notify the exception listener.
- void **setTransportInterruptionProcessingComplete** ()
Indicates that a Connection resource that is processing the transportInterrupted event has completed.
- **decaf::lang::Exception** * **getFirstFailureError** () const
Gets the pointer to the first exception that caused the Connection to become failed.
- void **onAsyncException** (const **decaf::lang::Exception** &ex)
Event handler for dealing with async exceptions.
- void **checkClosed** () const
Check for Closed State and Throw an exception if true.
- void **checkClosedOrFailed** () const
Check for Closed State and Failed State and Throw an exception if either is true.
- void **ensureConnectionInfoSent** ()
If its not been sent, then send the ConnectionInfo to the Broker.

Protected Member Functions

- virtual **Pointer** < **commands::SessionId** > **getNextSessionId** ()
- void **disconnect** (long long lastDeliveredSequenceId)
- void **waitForTransportInterruptionProcessingToComplete** ()
- void **signalInterruptionProcessingComplete** ()
- const **decaf::util::Properties** & **getProperties** () const

6.16.1 Detailed Description

Concrete connection used for all connectors to the ActiveMQ broker.

Since

2.0

6.16.2 Constructor & Destructor Documentation

6.16.2.1 `activemq::core::ActiveMQConnection::ActiveMQConnection (const Pointer< transport::Transport > & transport, const Pointer< decaf::util::Properties > & properties)`

Constructor.

Parameters

<i>transport</i>	The Transport requested for this connection to the Broker.
<i>properties</i>	The Properties that were defined for this connection

6.16.2.2 `virtual activemq::core::ActiveMQConnection::~~ActiveMQConnection () throw () [virtual]`

6.16.3 Member Function Documentation

6.16.3.1 `virtual void activemq::core::ActiveMQConnection::addDispatcher (const Pointer< commands::ConsumerId > & consumer, Dispatcher * dispatcher) [virtual]`

Adds a dispatcher for a consumer.

Parameters

<i>consumer</i>	- The consumer for which to register a dispatcher.
<i>dispatcher</i>	- The dispatcher to handle incoming messages for the consumer.

Exceptions

<i>CMSEException</i>	if an error occurs while removing performing the operation.
----------------------	---

6.16.3.2 `virtual void activemq::core::ActiveMQConnection::addProducer (ActiveMQProducer * producer) [virtual]`

Adds an active Producer to the Set of known producers.

Parameters

<i>producer</i>	The Producer to add from the the known set.
-----------------	---

Exceptions

<i>CMSEException</i>	if an error occurs while removing performing the operation.
----------------------	---

6.16.3.3 virtual void activemq::core::ActiveMQConnection::addSession (ActiveMQSession * session) [virtual]

Adds the session resources for the given session instance.

Parameters

<i>session</i>	The session to be added to this connection.
----------------	---

Exceptions

<i>CMSEException</i>	if an error occurs while removing performing the operation.
----------------------	---

6.16.3.4 void activemq::core::ActiveMQConnection::addTransportListener (transport::TransportListener * transportListener)

Adds a transport listener so that a client can be notified of events in the underlying transport, client's are always notified after the event has been processed by the Connection class.

Client's should ensure that the registered listener does not block or take a long amount of time to execute in order to not degrade performance of this Connection.

Parameters

<i>transport-Listener</i>	The TransportListener instance to add to this Connection's set of listeners to notify of Transport events.
---------------------------	--

6.16.3.5 void activemq::core::ActiveMQConnection::checkClosed () const

Check for Closed State and Throw an exception if true.

Exceptions

<i>CMSEException</i>	if the Connection is closed.
----------------------	------------------------------

6.16.3.6 void activemq::core::ActiveMQConnection::checkClosedOrFailed () const

Check for Closed State and Failed State and Throw an exception if either is true.

Exceptions

<i>CMSEException</i>	if the Connection is closed or failed.
----------------------	--

6.16.3.7 void activemq::core::ActiveMQConnection::cleanup ()

Clean up this connection object, reseting it back to a state that mirrors what a newly created **ActiveMQConnection** (p. 187) object has.

6.16.3.8 virtual void activemq::core::ActiveMQConnection::close () [virtual]

Closes this connection as well as any Sessions created from it (and those Sessions' consumers and producers).

Exceptions

CMSException (p. 826)

Implements **cms::Connection** (p. 935).

6.16.3.9 virtual cms::Session* activemq::core::ActiveMQConnection::createSession () [virtual]

Creates an AUTO_ACKNOWLEDGE **Session** (p. 2361).

Exceptions

CMSException (p. 826)

Implements **cms::Connection** (p. 935).

6.16.3.10 virtual cms::Session* activemq::core::ActiveMQConnection::createSession (cms::Session::AcknowledgeMode ackMode) [virtual]

Creates a new **Session** (p. 2361) to work for this **Connection** (p. 933) using the specified acknowledgment mode.

Parameters

<i>ackMode</i>	the Acknowledgment Mode to use.
----------------	---------------------------------

Exceptions

CMSException (p. 826)

Implements **cms::Connection** (p. 936).

Reimplemented in **activemq::core::ActiveMQXAConnection** (p. 433).

6.16.3.11 virtual void **activemq::core::ActiveMQConnection::destroyDestination** (
const **commands::ActiveMQDestination** * *destination*) [virtual]

Requests that the Broker removes the given Destination.

Calling this method implies that the client is finished with the Destination and that no other messages will be sent or received for the given Destination. The Broker frees all resources it has associated with this Destination.

Parameters

<i>destination</i>	The Destination the Broker will be requested to remove.
--------------------	---

Exceptions

<i>NullPointerException</i>	If the passed Destination is Null
<i>IllegalStateException</i>	If the connection is closed.
<i>UnsupportedOperationException</i>	If the wire format in use does not support this operation.
<i>ActiveMQException</i>	If any other error occurs during the attempt to destroy the destination.

6.16.3.12 virtual void **activemq::core::ActiveMQConnection::destroyDestination** (
const **cms::Destination** * *destination*) [virtual]

Requests that the Broker removes the given Destination.

Calling this method implies that the client is finished with the Destination and that no other messages will be sent or received for the given Destination. The Broker frees all resources it has associated with this Destination.

Parameters

<i>destination</i>	The CMS Destination the Broker will be requested to remove.
--------------------	---

Exceptions

<i>NullPointerException</i>	If the passed Destination is Null
<i>IllegalStateException</i>	If the connection is closed.
<i>UnsupportedOperationException</i>	If the wire format in use does not support this operation.
<i>ActiveMQException</i>	If any other error occurs during the attempt to destroy the destination.

6.16.3.13 void **activemq::core::ActiveMQConnection::disconnect** (long long
lastDeliveredSequenceId) [protected]

6.16.3.14 `void activemq::core::ActiveMQConnection::ensureConnectionInfoSent ()`

If its not been sent, then send the ConnectionInfo to the Broker.

6.16.3.15 `virtual void activemq::core::ActiveMQConnection::fire (const exceptions::ActiveMQException & ex) [virtual]`

Notify the exception listener.

Parameters

<code>ex</code>	the exception to fire
-----------------	-----------------------

6.16.3.16 `const std::string& activemq::core::ActiveMQConnection::getBrokerURL () const`

Gets the Broker URL that this factory will use when creating a new connection instance.

Returns

brokerURL string

6.16.3.17 `virtual std::string activemq::core::ActiveMQConnection::getClientID () const [virtual]`

Get the Client Id for this session, the client Id is provider specific and is either assigned by the connection factory or set using the setClientID method.

Returns

Client Id String for this **Connection** (p. 933).

Exceptions

<i>CMSException</i> (p. 826)	if the provider fails to return the client id or an internal error occurs.
--	--

Implements **cms::Connection** (p. 936).

6.16.3.18 `unsigned int activemq::core::ActiveMQConnection::getCloseTimeout () const`

Gets the assigned close timeout for this Connector.

Returns

the close timeout configured in the connection uri

6.16.3.19 `int activemq::core::ActiveMQConnection::getCompressionLevel ()`
`const`

Gets the currently configured Compression level for Message bodies.

Returns

the int value of the current compression level.

6.16.3.20 `const commands::ConnectionId& activemq::core-
 ::ActiveMQConnection::getConnectionId ()`
`const`

Gets the ConnectionId for this Object, if the Connection is not open than this method throws an exception.

Exceptions

<i>ActiveMQException</i>	if an error occurs while performing this operation.
--------------------------	---

6.16.3.21 `const commands::ConnectionInfo& activemq::core-
 ::ActiveMQConnection::getConnectionInfo ()`
`const`

Gets the ConnectionInfo for this Object, if the Connection is not open than this method throws an exception.

Exceptions

<i>ActiveMQException</i>	if an error occurs while performing this operation.
--------------------------	---

6.16.3.22 `virtual cms::ExceptionListener* activemq::core:-
 ActiveMQConnection::getExceptionListener () const`
`[virtual]`

Gets the registered Exception Listener for this connection.

Returns

pointer to an exception listener or NULL

Implements **cms::Connection** (p. 936).

6.16.3.23 `decaf::lang::Exception* activemq::core::ActiveMQConnection::getFirstFailureError () const`

Gets the pointer to the first exception that caused the Connection to become failed.

Returns

pointer to and Exception instance or NULL if none is set.

6.16.3.24 `virtual const cms::ConnectionMetaData* activemq::core::ActiveMQConnection::getMetaData () const [inline, virtual]`

Gets the metadata for this connection.

Returns

the connection MetaData pointer (caller does not own it).

Exceptions

CMSException (p. 826)	if the provider fails to get the connection metadata for this connection.
---------------------------------	---

See also

ConnectionMetaData (p. 978)

Since

2.0

Implements **cms::Connection** (p. 937).

6.16.3.25 `long long activemq::core::ActiveMQConnection::getNextLocalTransactionId ()`

Get the Next Temporary Destination Id.

Returns

the next id in the sequence.

6.16.3.26 `virtual Pointer<commands::SessionId> activemq::core::ActiveMQConnection::getNextSessionId () [protected, virtual]`

Returns

the next available Session Id.

6.16.3.27 long long activemq::core::ActiveMQConnection::getNextTempDestinationId ()

Get the Next Temporary Destination Id.

Returns

the next id in the sequence.

6.16.3.28 const std::string& activemq::core::ActiveMQConnection::getPassword () const

Gets the password that this factory will use when creating a new connection instance.

Returns

password string, "" for default credentials

6.16.3.29 PrefetchPolicy* activemq::core::ActiveMQConnection::getPrefetchPolicy () const

Gets the pointer to the current **PrefetchPolicy** (p.2110) that is in use by this - ConnectionFactory.

Returns

a pointer to this objects **PrefetchPolicy** (p. 2110).

6.16.3.30 unsigned int activemq::core::ActiveMQConnection::getProducerWindowSize () const

Gets the configured producer window size for Producers that are created from this connector.

This only applies if there is no send timeout and the producer is able to send asynchronously.

Returns

size in bytes of messages that this producer can produce before it must block and wait for ProducerAck messages to free resources.

6.16.3.31 `const decaf::util::Properties& activemq::core::ActiveMQConnection::getProperties () const` [protected]

6.16.3.32 `RedeliveryPolicy* activemq::core::ActiveMQConnection::getRedeliveryPolicy () const`

Gets the pointer to the current **RedeliveryPolicy** (p.2250) that is in use by this - ConnectionFactory.

Returns

a pointer to this objects **RedeliveryPolicy** (p. 2250).

6.16.3.33 `std::string activemq::core::ActiveMQConnection::getResourceManagerId () const`

Returns the Id of the Resource Manager that this client will use should it be entered into an XA Transaction.

Returns

a string containing the resource manager Id for XA Transactions.

6.16.3.34 `Pointer<threads::Scheduler> activemq::core::ActiveMQConnection::getScheduler () const`

Gets a reference to the Connection objects built in Scheduler instance.

Returns

a reference to a Scheduler instance owned by this Connection.

6.16.3.35 `unsigned int activemq::core::ActiveMQConnection::getSendTimeout () const`

Gets the assigned send timeout for this Connector.

Returns

the send timeout configured in the connection uri

6.16.3.36 `transport::Transport& activemq::core::ActiveMQConnection::getTransport () const`

Gets a reference to this object's Transport instance.

Returns

a reference to the Transport that is in use by this Connection.

6.16.3.37 `const std::string& activemq::core::ActiveMQConnection::getUsername () const`

Gets the username that this factory will use when creating a new connection instance.

Returns

username string, "" for default credentials

6.16.3.38 `bool activemq::core::ActiveMQConnection::isAlwaysSyncSend () const`

Gets if the Connection should always send things Synchronously.

Returns

true if sends should always be Synchronous.

6.16.3.39 `bool activemq::core::ActiveMQConnection::isClosed () const`
`[inline]`

Checks if this connection has been closed.

Returns

true if the connection is closed

6.16.3.40 `bool activemq::core::ActiveMQConnection::isDispatchAsync () const`

Returns

The value of the dispatch asynchronously option sent to the broker.

6.16.3.41 `bool activemq::core::ActiveMQConnection::isMessagePriority-Supported () const`

Returns

true if the Connections that this factory creates should support the message based priority settings.

6.16.3.42 `bool activemq::core::ActiveMQConnection::isStarted () const`
[inline]

Check if this connection has been started.

Returns

true if the start method has been called.

6.16.3.43 `bool activemq::core::ActiveMQConnection::isTransportFailed () const`
[inline]

Checks if the Connection's Transport has failed.

Returns

true if the Connection's Transport has failed.

6.16.3.44 `bool activemq::core::ActiveMQConnection::isUseAsyncSend () const`

Gets if the useAsyncSend option is set.

Returns

true if on false if not.

6.16.3.45 `bool activemq::core::ActiveMQConnection::isUseCompression () const`

Gets if the Connection is configured for Message body compression.

Returns

if the Message body will be Compressed or not.

6.16.3.46 `void activemq::core::ActiveMQConnection::onAsyncException (const`
`decaf::lang::Exception & ex)`

Event handler for dealing with async exceptions.

Parameters

<code>ex</code>	The exception that caused the error condition.
-----------------	--

6.16.3.47 `virtual void activemq::core::ActiveMQConnection::onCommand (const Pointer< commands::Command > & command) [virtual]`

Event handler for the receipt of a non-response command from the transport.

Parameters

<i>command</i>	the received command object.
----------------	------------------------------

Implements **activemq::transport::TransportListener** (p.2811).

6.16.3.48 `void activemq::core::ActiveMQConnection::oneway (Pointer< commands::Command > command)`

Sends a message without request that the broker send a response to indicate that it was received.

Parameters

<i>command</i>	The Command object to send to the Broker.
----------------	---

Exceptions

<i>ActiveMQException</i>	if not currently connected, or if the operation fails for any reason.
--------------------------	---

6.16.3.49 `virtual void activemq::core::ActiveMQConnection::onException (const decaf::lang::Exception & ex) [virtual]`

Event handler for an exception from a command transport.

Parameters

<i>ex</i>	The exception.
-----------	----------------

Implements **activemq::transport::TransportListener** (p.2811).

6.16.3.50 `virtual void activemq::core::ActiveMQConnection::removeDispatcher (const Pointer< commands::ConsumerId > & consumer) [virtual]`

Removes the dispatcher for a consumer.

Parameters

<i>consumer</i>	- The consumer for which to remove the dispatcher.
-----------------	--

Exceptions

<i>CMSEException</i>	if an error occurs while removing performing the operation.
----------------------	---

6.16.3.51 **virtual void activemq::core::ActiveMQConnection::removeProducer (**
const Pointer< commands::ProducerId > & producerId) [virtual]

Removes an active Producer to the Set of known producers.

Parameters

<i>producerId</i>	- The ProducerId to remove from the the known set.
-------------------	--

Exceptions

<i>CMSEException</i>	if an error occurs while removing performing the operation.
----------------------	---

6.16.3.52 **virtual void activemq::core::ActiveMQConnection::removeSession (**
ActiveMQSession * session) [virtual]

Removes the session resources for the given session instance.

Parameters

<i>session</i>	The session to be unregistered from this connection.
----------------	--

Exceptions

<i>CMSEException</i>	if an error occurs while removing performing the operation.
----------------------	---

6.16.3.53 **void activemq::core::ActiveMQConnection::removeTransportListener (**
transport::TransportListener * transportListener)

Removes a registered TransportListener from the Connection's set of Transport listeners, this listener will no longer receive any Transport related events.

The caller is responsible for freeing the listener in all cases.

Parameters

<i>transport-Listener</i>	The pointer to the TransportListener to remove from the set of listeners.
---------------------------	---

6.16.3.54 `virtual void activemq::core::ActiveMQConnection::sendPullRequest
(const commands::ConsumerInfo * consumer, long long timeout)
[virtual]`

If supported sends a message pull request to the service provider asking for the delivery of a new message.

This is used in the case where the service provider has been configured with a zero prefetch or is only capable of delivering messages on a pull basis.

Parameters

<i>consumer</i>	- the ConsumerInfo for the requesting Consumer.
<i>timeout</i>	- the time that the client is willing to wait.

Exceptions

<i>ActiveMQException</i>	if an error occurs while removing performing the operation.
--------------------------	---

6.16.3.55 `void activemq::core::ActiveMQConnection::setAlwaysSyncSend (bool
value)`

Sets if the Connection should always send things Synchronously.

Parameters

<i>value</i>	true if sends should always be Synchronous.
--------------	---

6.16.3.56 `void activemq::core::ActiveMQConnection::setBrokerURL (const
std::string & brokerURL)`

Sets the Broker URL that should be used when creating a new connection instance.

Parameters

<i>brokerURL</i>	string
------------------	--------

6.16.3.57 `virtual void activemq::core::ActiveMQConnection::setClientID (const
std::string & clientID) [virtual]`

Sets the client identifier for this connection.

The preferred way to assign a CMS client's client identifier is for it to be configured in a client-specific **ConnectionFactory** (p. 955) object and transparently assigned to the **Connection** (p. 933) object it creates.

If a client sets the client identifier explicitly, it must do so immediately after it creates

the connection and before any other action on the connection is taken. After this point, setting the client identifier is a programming error that should throw an **IllegalStateException** (p. 1419).

Parameters

<i>clientId</i>	The unique client identifier to assign to the Connection (p. 933).
-----------------	---

Exceptions

CMSException (p. 826)	if the provider fails to set the client id due to some internal error.
<i>InvalidClientID-Exception</i>	if the id given is somehow invalid or is a duplicate.
IllegalStateException (p. 1419)	if the client tries to set the id after a Connection (p. 933) method has been called.

Implements **cms::Connection** (p. 937).

6.16.3.58 **void activemq::core::ActiveMQConnection::setCloseTimeout (unsigned int timeout)**

Sets the close timeout to use when sending the disconnect request.

Parameters

<i>timeout</i>	- The time to wait for a close message.
----------------	---

6.16.3.59 **void activemq::core::ActiveMQConnection::setCompressionLevel (int value)**

Sets the Compression level used when Message body compression is enabled, a value of -1 causes the Compression Library to use the default setting which is a balance of speed and compression.

The range of compression levels is [0..9] where 0 indicates best speed and 9 indicates best compression.

Parameters

<i>value</i>	A signed int value that controls the compression level.
--------------	---

6.16.3.60 **void activemq::core::ActiveMQConnection::setDefaultClientId (const std::string & clientId)**

Sets the Client Id.

Parameters

<i>clientId</i>	- The new clientId value.
-----------------	---------------------------

6.16.3.61 void **activemq::core::ActiveMQConnection::setDispatchAsync** (bool *value*)

Should messages be dispatched synchronously or asynchronously from the producer thread for non-durable topics in the broker? For fast consumers set this to false.

For slow consumers set it to true so that dispatching will not block fast consumers. .

Parameters

<i>value</i>	The value of the dispatch asynchronously option sent to the broker.
--------------	---

6.16.3.62 virtual void **activemq::core::ActiveMQConnection::setExceptionListener** (cms::ExceptionListener * *listener*) [virtual]

Sets the registered Exception Listener for this connection.

Parameters

<i>listener</i>	pointer to and ExceptionListener (p. 1286)
-----------------	---

Implements **cms::Connection** (p. 938).

6.16.3.63 void **activemq::core::ActiveMQConnection::setMessagePriority-Supported** (bool *value*)

Set whether or not this factory should create Connection objects with the Message priority support function enabled.

Parameters

<i>value</i>	Boolean indicating if Message priority should be enabled.
--------------	---

6.16.3.64 void **activemq::core::ActiveMQConnection::setPassword** (const std::string & *password*)

Sets the password that should be used when creating a new connection.

Parameters

<i>password</i>	string
-----------------	--------

6.16.3.65 void activemq::core::ActiveMQConnection::setPrefetchPolicy (PrefetchPolicy * *policy*)

Sets the **PrefetchPolicy** (p. 2110) instance that this factory should use when it creates new Connection instances.

The **PrefetchPolicy** (p. 2110) passed becomes the property of the factory and will be deleted when the factory is destroyed.

Parameters

<i>policy</i>	The new PrefetchPolicy (p. 2110) that the ConnectionFactory should clone for Connections.
---------------	--

6.16.3.66 void activemq::core::ActiveMQConnection::setProducerWindowSize (unsigned int *windowSize*)

Sets the size in Bytes of messages that a producer can send before it is blocked to await a ProducerAck from the broker that frees enough memory to allow another message to be sent.

Parameters

<i>windowSize</i>	- The size in bytes of the Producers memory window.
-------------------	---

6.16.3.67 void activemq::core::ActiveMQConnection::setRedeliveryPolicy (RedeliveryPolicy * *policy*)

Sets the **RedeliveryPolicy** (p. 2250) instance that this factory should use when it creates new Connection instances.

The **RedeliveryPolicy** (p. 2250) passed becomes the property of the factory and will be deleted when the factory is destroyed.

Parameters

<i>policy</i>	The new RedeliveryPolicy (p. 2250) that the ConnectionFactory should clone for Connections.
---------------	--

6.16.3.68 void activemq::core::ActiveMQConnection::setSendTimeout (unsigned int *timeout*)

Sets the send timeout to use when sending Message objects, this will cause all messages to be sent using a Synchronous request is non-zero.

Parameters

<i>timeout</i>	- The time to wait for a response.
----------------	------------------------------------

6.16.3.69 void activemq::core::ActiveMQConnection::setTransportInterruption-ProcessingComplete ()

Indicates that a Connection resource that is processing the transportInterrupted event has completed.

6.16.3.70 void activemq::core::ActiveMQConnection::setUseAsyncSend (bool *value*)

Sets the useAsyncSend option.

Parameters

<i>value</i>	- true to activate, false to disable.
--------------	---------------------------------------

6.16.3.71 void activemq::core::ActiveMQConnection::setUseCompression (bool *value*)

Sets whether Message body compression is enabled.

Parameters

<i>value</i>	Boolean indicating if Message body compression is enabled.
--------------	--

6.16.3.72 void activemq::core::ActiveMQConnection::setUsername (const std::string & *username*)

Sets the username that should be used when creating a new connection.

Parameters

<i>username</i>	string
-----------------	--------

6.16.3.73 void activemq::core::ActiveMQConnection::signalInterruption-ProcessingComplete () [protected]**6.16.3.74 virtual void activemq::core::ActiveMQConnection::start () [virtual]**

Starts the service.

Exceptions

<i>CMSException</i> (p. 826)	if an internal error occurs while starting.
--	---

Implements **cms::Startable** (p. 2534).

6.16.3.75 **virtual void activemq::core::ActiveMQConnection::stop ()** [virtual]

Stops this service.

Exceptions

<i>CMSException</i> (p. 826)	- if an internal error occurs while stopping the Service.
--	---

Implements **cms::Stoppable** (p. 2602).

6.16.3.76 **Pointer<commands::Response> activemq::core::ActiveMQConnection::syncRequest (Pointer< commands::Command > command, unsigned int timeout = 0)**

Sends a synchronous request and returns the response from the broker.

This method converts any error responses it receives into an exception.

Parameters

<i>command</i>	The Command object that is to be sent to the broker.
<i>timeout</i>	The time in milliseconds to wait for a response, default is zero or infinite.

Returns

a Pointer instance to the Response object sent from the Broker.

Exceptions

<i>BrokerException</i>	if the response from the broker is of type ExceptionResponse.
<i>ActiveMQException</i>	if any other error occurs while sending the Command.

6.16.3.77 **virtual void activemq::core::ActiveMQConnection::transportInterrupted ()** [virtual]

The transport has suffered an interruption from which it hopes to recover.

Implements **activemq::transport::TransportListener** (p. 2812).

6.16.3.78 `virtual void activemq::core::ActiveMQConnection::transportResumed ()`
`[virtual]`

The transport has resumed after an interruption.

Implements `activemq::transport::TransportListener` (p. 2812).

6.16.3.79 `void activemq::core::ActiveMQConnection::waitFor-`
`TransportInterruptionProcessingToComplete ()`
`[protected]`

The documentation for this class was generated from the following file:

- `src/main/activemq/core/ActiveMQConnection.h`

6.17 activemq::core::ActiveMQConnectionFactory Class Reference

```
#include <src/main/activemq/core/ActiveMQConnectionFactory.h>
```

Inheritance diagram for `activemq::core::ActiveMQConnectionFactory`:

Public Member Functions

- **ActiveMQConnectionFactory** ()
- **ActiveMQConnectionFactory** (const std::string &uri, const std::string &username="", const std::string &password="")
Constructor.
- **ActiveMQConnectionFactory** (const **decaf::net::URI** &uri, const std::string &username="", const std::string &password="")
Constructor.
- virtual **~ActiveMQConnectionFactory** () throw ()
- virtual **cms::Connection** * **createConnection** ()
Creates a connection with the default user identity.
- virtual **cms::Connection** * **createConnection** (const std::string &username, const std::string &password)
Creates a connection with the specified user identity.
- virtual **cms::Connection** * **createConnection** (const std::string &username, const std::string &password, const std::string &clientId)
Creates a connection with the specified user identity.
- void **setUsername** (const std::string &username)
Sets the username that should be used when creating a new connection.
- const std::string & **getUsername** () const

- Gets the username that this factory will use when creating a new connection instance.*

 - void **setPassword** (const std::string &password)

Sets the password that should be used when creating a new connection.
- const std::string & **getPassword** () const

Gets the password that this factory will use when creating a new connection instance.
- std::string **getClientId** () const

Gets the Configured Client Id.
- void **setClientId** (const std::string &clientId)

Sets the Client Id.
- void **setBrokerURI** (const std::string &uri)

Sets the Broker URI that should be used when creating a new connection instance.
- void **setBrokerURI** (const decaf::net::URI &uri)

Sets the Broker URI that should be used when creating a new connection instance.
- const decaf::net::URI & **getBrokerURI** () const

Gets the Broker URI that this factory will use when creating a new connection instance.
- void **setExceptionHandler** (cms::ExceptionHandler *listener)

Set an CMS ExceptionListener that will be set on eat connection once it has been created.
- cms::ExceptionHandler * **getExceptionHandler** () const

Returns the currently set ExceptionListener that will be set on any new Connection instance that is created by this factory.
- void **setPrefetchPolicy** (PrefetchPolicy *policy)

*Sets the **PrefetchPolicy** (p. 2110) instance that this factory should use when it creates new Connection instances.*
- PrefetchPolicy * **getPrefetchPolicy** () const

*Gets the pointer to the current **PrefetchPolicy** (p. 2110) that is in use by this - ConnectionFactory.*
- void **setRedeliveryPolicy** (RedeliveryPolicy *policy)

*Sets the **RedeliveryPolicy** (p. 2250) instance that this factory should use when it creates new Connection instances.*
- RedeliveryPolicy * **getRedeliveryPolicy** () const

*Gets the pointer to the current **RedeliveryPolicy** (p. 2250) that is in use by this - ConnectionFactory.*
- bool **isDispatchAsync** () const
- void **setDispatchAsync** (bool value)

Should messages be dispatched synchronously or asynchronously from the producer thread for non-durable topics in the broker? For fast consumers set this to false.
- bool **isAlwaysSyncSend** () const

Gets if the Connection should always send things Synchronously.
- void **setAlwaysSyncSend** (bool value)

Sets if the Connection should always send things Synchronously.
- bool **isUseAsyncSend** () const

Gets if the useAsyncSend option is set.
- void **setUseAsyncSend** (bool value)

Sets the useAsyncSend option.

- bool **isUseCompression** () const
Gets if the Connection is configured for Message body compression.
- void **setUseCompression** (bool value)
Sets whether Message body compression is enabled.
- void **setCompressionLevel** (int value)
Sets the Compression level used when Message body compression is enabled, a value of -1 causes the Compression Library to use the default setting which is a balance of speed and compression.
- int **getCompressionLevel** () const
Gets the currently configured Compression level for Message bodies.
- unsigned int **getSendTimeout** () const
Gets the assigned send timeout for this Connector.
- void **setSendTimeout** (unsigned int timeout)
Sets the send timeout to use when sending Message objects, this will cause all messages to be sent using a Synchronous request is non-zero.
- unsigned int **getCloseTimeout** () const
Gets the assigned close timeout for this Connector.
- void **setCloseTimeout** (unsigned int timeout)
Sets the close timeout to use when sending the disconnect request.
- unsigned int **getProducerWindowSize** () const
Gets the configured producer window size for Producers that are created from this connector.
- void **setProducerWindowSize** (unsigned int windowSize)
Sets the size in Bytes of messages that a producer can send before it is blocked to await a ProducerAck from the broker that frees enough memory to allow another message to be sent.
- bool **isMessagePrioritySupported** () const
- void **setMessagePrioritySupported** (bool value)
Set whether or not this factory should create Connection objects with the Message priority support function enabled.

Static Public Member Functions

- static **cms::Connection * createConnection** (const std::string &uri, const std::string &username, const std::string &password, const std::string &clientId="")
Creates a connection with the specified user identity.

Static Public Attributes

- static const std::string **DEFAULT_URI**

Protected Member Functions

- virtual **ActiveMQConnection** * **createActiveMQConnection** (const **Pointer**< **transport::Transport** > &transport, const **Pointer**< **decaf::util::Properties** > &properties)

Create a new **ActiveMQConnection** (p. 187) instance using the provided *Transport* and *Properties*.

6.17.1 Constructor & Destructor Documentation

6.17.1.1 **activemq::core::ActiveMQConnectionFactory::ActiveMQConnectionFactory** ()

6.17.1.2 **activemq::core::ActiveMQConnectionFactory::ActiveMQConnectionFactory** (const std::string & *uri*, const std::string & *username* = " ", const std::string & *password* = " ")

Constructor.

Parameters

<i>uri</i>	the URI of the Broker we are connecting to.
<i>username</i>	to authenticate with, defaults to ""
<i>password</i>	to authenticate with, defaults to ""

6.17.1.3 **activemq::core::ActiveMQConnectionFactory::ActiveMQConnectionFactory** (const **decaf::net::URI** & *uri*, const std::string & *username* = " ", const std::string & *password* = " ")

Constructor.

Parameters

<i>uri</i>	the URI of the Broker we are connecting to.
<i>username</i>	to authenticate with, defaults to ""
<i>password</i>	to authenticate with, defaults to ""

6.17.1.4 virtual **activemq::core::ActiveMQConnectionFactory::~~ActiveMQConnectionFactory** () throw () [virtual]

6.17.2 Member Function Documentation

6.17.2.1 `virtual ActiveMQConnection* activemq::core::ActiveMQConnectionFactory::createActiveMQConnection (const Pointer< transport::Transport > & transport, const Pointer< decaf::util::Properties > & properties)` [protected, virtual]

Create a new **ActiveMQConnection** (p. 187) instance using the provided Transport and Properties.

Subclasses can override this to control the actual type of **ActiveMQConnection** (p. 187) that is created.

Parameters

<i>transport</i>	The Transport that the Connection should use to communicate with the Broker.
<i>properties</i>	The Properties that are assigned to the new Connection instance.

Returns

a new **ActiveMQConnection** (p. 187) pointer instance.

Reimplemented in **activemq::core::ActiveMQXAConnectionFactory** (p. 435).

6.17.2.2 `virtual cms::Connection* activemq::core::ActiveMQConnectionFactory::createConnection ()` [virtual]

Creates a connection with the default user identity.

The connection is created in stopped mode. No messages will be delivered until the Connection.start method is explicitly called.

Returns

a Connection Pointer

Exceptions

<i>CMSException</i>	
---------------------	--

Implements **cms::ConnectionFactory** (p. 957).

6.17.2.3 `virtual cms::Connection* activemq::core::ActiveMQConnectionFactory::createConnection (const std::string & username, const std::string & password)` [virtual]

Creates a connection with the specified user identity.

The connection is created in stopped mode. No messages will be delivered until the Connection.start method is explicitly called. The username and password values

passed here do not change the defaults, subsequent calls to the parameterless create-Connection will continue to use the default values that were set in the Constructor.

Parameters

<i>username</i>	to authenticate with
<i>password</i>	to authenticate with

Returns

a Connection Pointer

Exceptions

<i>CMSException</i>	
---------------------	--

Implements **cms::ConnectionFactory** (p. 957).

6.17.2.4 `virtual cms::Connection* activemq::core::ActiveMQConnectionFactory::createConnection (const std::string & username, const std::string & password, const std::string & clientId) [virtual]`

Creates a connection with the specified user identity.

The connection is created in stopped mode. No messages will be delivered until the Connection.start method is explicitly called. The username and password values passed here do not change the defaults, subsequent calls to the parameterless create-Connection will continue to use the default values that were set in the Constructor.

Parameters

<i>username</i>	to authenticate with
<i>password</i>	to authenticate with
<i>clientId</i>	to assign to connection if "" then a random client Id is created for this connection.

Returns

a Connection Pointer

Exceptions

<i>CMSException</i>	
---------------------	--

Implements **cms::ConnectionFactory** (p. 958).

6.17.2.5 `static cms::Connection* activemq::core::ActiveMQConnectionFactory::createConnection (const std::string & uri, const std::string & username, const std::string & password, const std::string & clientId = " ") [static]`

Creates a connection with the specified user identity.

The connection is created in stopped mode. No messages will be delivered until the Connection.start method is explicitly called.

Parameters

<i>uri</i>	The URI of the Broker we are connecting to.
<i>username</i>	The name of the user to authenticate with.
<i>password</i>	The password for the user to authenticate with.
<i>clientId</i>	The unique client id to assign to connection, defaults to "".

Exceptions

<i>CMSException.</i>	
----------------------	--

6.17.2.6 `const decaf::net::URI& activemq::core::ActiveMQConnectionFactory::getBrokerURI () const`

Gets the Broker URI that this factory will use when creating a new connection instance.

Returns

brokerURI string

6.17.2.7 `std::string activemq::core::ActiveMQConnectionFactory::getClientId () const`

Gets the Configured Client Id.

Returns

the clientId.

6.17.2.8 `unsigned int activemq::core::ActiveMQConnectionFactory::getCloseTimeout () const`

Gets the assigned close timeout for this Connector.

Returns

the close timeout configured in the connection uri

6.17.2.9 `int activemq::core::ActiveMQConnectionFactory::getCompressionLevel () const`

Gets the currently configured Compression level for Message bodies.

Returns

the int value of the current compression level.

6.17.2.10 `cms::ExceptionListener* activemq::core::ActiveMQConnectionFactory::getExceptionListener () const`

Returns the currently set ExceptionListener that will be set on any new Connection instance that is created by this factory.

Returns

a pointer to a CMS ExceptionListener instance or NULL if not set.

6.17.2.11 `const std::string& activemq::core::ActiveMQConnectionFactory::getPassword () const`

Gets the password that this factory will use when creating a new connection instance.

Returns

password string, "" for default credentials

6.17.2.12 `PrefetchPolicy* activemq::core::ActiveMQConnectionFactory::getPrefetchPolicy () const`

Gets the pointer to the current **PrefetchPolicy** (p.2110) that is in use by this - ConnectionFactory.

Returns

a pointer to this objects **PrefetchPolicy** (p. 2110).

6.17.2.13 `unsigned int activemq::core::ActiveMQConnectionFactory::getProducerWindowSize () const`

Gets the configured producer window size for Producers that are created from this connector.

This only applies if there is no send timeout and the producer is able to send asynchronously.

Returns

size in bytes of messages that this producer can produce before it must block and wait for ProducerAck messages to free resources.

6.17.2.14 RedeliveryPolicy* activemq::core::ActiveMQConnectionFactory::getRedeliveryPolicy () const

Gets the pointer to the current **RedeliveryPolicy** (p.2250) that is in use by this - ConnectionFactory.

Returns

a pointer to this objects **RedeliveryPolicy** (p. 2250).

6.17.2.15 unsigned int activemq::core::ActiveMQConnectionFactory::getSendTimeout () const

Gets the assigned send timeout for this Connector.

Returns

the send timeout configured in the connection uri

6.17.2.16 const std::string& activemq::core::ActiveMQConnectionFactory::getUsername () const

Gets the username that this factory will use when creating a new connection instance.

Returns

username string, "" for default credentials

6.17.2.17 bool activemq::core::ActiveMQConnectionFactory::isAlwaysSyncSend () const

Gets if the Connection should always send things Synchronously.

Returns

true if sends should always be Synchronous.

6.17.2.18 **bool activemq::core::ActiveMQConnectionFactory::isDispatchAsync ()**
const

Returns

The value of the dispatch asynchronously option sent to the broker.

6.17.2.19 **bool activemq::core::ActiveMQConnectionFactory::isMessagePriority-Supported ()** const

Returns

true if the Connections that this factory creates should support the message based priority settings.

6.17.2.20 **bool activemq::core::ActiveMQConnectionFactory::isUseAsyncSend ()**
const

Gets if the useAsyncSend option is set.

Returns

true if on false if not.

6.17.2.21 **bool activemq::core::ActiveMQConnectionFactory::isUseCompression ()** const

Gets if the Connection is configured for Message body compression.

Returns

if the Message body will be Compressed or not.

6.17.2.22 **void activemq::core::ActiveMQConnectionFactory::setAlwaysSyncSend (bool value)**

Sets if the Connection should always send things Synchronously.

Parameters

<i>value</i>	true if sends should always be Synchronous.
--------------	---

6.17.2.23 `void activemq::core::ActiveMQConnectionFactory::setBrokerURI (const std::string & uri)`

Sets the Broker URI that should be used when creating a new connection instance.

Parameters

<i>brokerURI</i>	The string form of the Broker URI, this will be converted to a URI object.
------------------	--

6.17.2.24 `void activemq::core::ActiveMQConnectionFactory::setBrokerURI (const decaf::net::URI & uri)`

Sets the Broker URI that should be used when creating a new connection instance.

Parameters

<i>brokerURI</i>	The URI of the broker that this client will connect to.
------------------	---

6.17.2.25 `void activemq::core::ActiveMQConnectionFactory::setClientId (const std::string & clientId)`

Sets the Client Id.

Parameters

<i>clientId</i>	- The new clientId value.
-----------------	---------------------------

6.17.2.26 `void activemq::core::ActiveMQConnectionFactory::setCloseTimeout (unsigned int timeout)`

Sets the close timeout to use when sending the disconnect request.

Parameters

<i>timeout</i>	- The time to wait for a close message.
----------------	---

6.17.2.27 `void activemq::core::ActiveMQConnectionFactory::setCompressionLevel (int value)`

Sets the Compression level used when Message body compression is enabled, a value of -1 causes the Compression Library to use the default setting which is a balance of speed and compression.

The range of compression levels is [0..9] where 0 indicates best speed and 9 indicates best compression.

Parameters

<i>value</i>	A signed int value that controls the compression level.
--------------	---

6.17.2.28 `void activemq::core::ActiveMQConnectionFactory::setDispatchAsync (bool value)`

Should messages be dispatched synchronously or asynchronously from the producer thread for non-durable topics in the broker? For fast consumers set this to false.

For slow consumers set it to true so that dispatching will not block fast consumers. .

Parameters

<i>value</i>	The value of the dispatch asynchronously option sent to the broker.
--------------	---

6.17.2.29 `void activemq::core::ActiveMQConnectionFactory::set-ExceptionListener (cms::ExceptionListener * listener)`

Set an CMS ExceptionListener that will be set on eat connection once it has been created.

The factory does not take ownership of this pointer, the client must ensure that its lifetime is scoped to the connection that it is applied to.

Parameters

<i>listener</i>	The listener to set on the connection or NULL for no listener.
-----------------	--

6.17.2.30 `void activemq::core::ActiveMQConnectionFactory::setMessagePriority-Supported (bool value)`

Set whether or not this factory should create Connection objects with the Message priority support function enabled.

Parameters

<i>value</i>	Boolean indicating if Message priority should be enabled.
--------------	---

6.17.2.31 `void activemq::core::ActiveMQConnectionFactory::setPassword (const std::string & password)`

Sets the password that should be used when creating a new connection.

Parameters

<i>password</i>	string
-----------------	--------

6.17.2.32 void activemq::core::ActiveMQConnectionFactory::setPrefetchPolicy (PrefetchPolicy * policy)

Sets the **PrefetchPolicy** (p. 2110) instance that this factory should use when it creates new Connection instances.

The **PrefetchPolicy** (p. 2110) passed becomes the property of the factory and will be deleted when the factory is destroyed.

Parameters

<i>policy</i>	The new PrefetchPolicy (p. 2110) that the ConnectionFactory should clone for Connections.
---------------	--

6.17.2.33 void activemq::core::ActiveMQConnectionFactory::setProducerWindowSize (unsigned int windowSize)

Sets the size in Bytes of messages that a producer can send before it is blocked to await a ProducerAck from the broker that frees enough memory to allow another message to be sent.

Parameters

<i>windowSize</i>	- The size in bytes of the Producers memory window.
-------------------	---

6.17.2.34 void activemq::core::ActiveMQConnectionFactory::setRedeliveryPolicy (RedeliveryPolicy * policy)

Sets the **RedeliveryPolicy** (p. 2250) instance that this factory should use when it creates new Connection instances.

The **RedeliveryPolicy** (p. 2250) passed becomes the property of the factory and will be deleted when the factory is destroyed.

Parameters

<i>policy</i>	The new RedeliveryPolicy (p. 2250) that the ConnectionFactory should clone for Connections.
---------------	--

6.17.2.35 `void activemq::core::ActiveMQConnectionFactory::setSendTimeout (unsigned int timeout)`

Sets the send timeout to use when sending Message objects, this will cause all messages to be sent using a Synchronous request is non-zero.

Parameters

<i>timeout</i>	- The time to wait for a response.
----------------	------------------------------------

6.17.2.36 `void activemq::core::ActiveMQConnectionFactory::setUseAsyncSend (bool value)`

Sets the useAsyncSend option.

Parameters

<i>value</i>	- true to activate, false to disable.
--------------	---------------------------------------

6.17.2.37 `void activemq::core::ActiveMQConnectionFactory::setUseCompression (bool value)`

Sets whether Message body compression is enabled.

Parameters

<i>value</i>	Boolean indicating if Message body compression is enabled.
--------------	--

6.17.2.38 `void activemq::core::ActiveMQConnectionFactory::setUsername (const std::string & username)`

Sets the username that should be used when creating a new connection.

Parameters

<i>username</i>	string
-----------------	--------

6.17.3 Field Documentation

6.17.3.1 `const std::string activemq::core::ActiveMQConnectionFactory::DEFAULT_URI [static]`

The documentation for this class was generated from the following file:

- src/main/activemq/core/**ActiveMQConnectionFactory.h**

6.18 activemq::core::ActiveMQConnectionMetaData Class - Reference

This class houses all the various settings and information that is used by an instance of an **ActiveMQConnection** (p. 187) class.

```
#include <src/main/activemq/core/ActiveMQConnectionMeta-Data.h>
```

Inheritance diagram for `activemq::core::ActiveMQConnectionMetaData`:

Public Member Functions

- **ActiveMQConnectionMetaData** ()
- virtual **~ActiveMQConnectionMetaData** () throw ()
- virtual std::string **getCMSVersion** () const
Gets the CMS API version.
- virtual int **getCMSMajorVersion** () const
Gets the CMS major version number.
- virtual int **getCMSMinorVersion** () const
Gets the CMS minor version number.
- virtual std::string **getCMSProviderName** () const
Gets the CMS provider name.
- virtual std::string **getProviderVersion** () const
Gets the CMS provider version.
- virtual int **getProviderMajorVersion** () const
Gets the CMS provider major version number.
- virtual int **getProviderMinorVersion** () const
Gets the CMS provider minor version number.
- virtual std::vector< std::string > **getCMSXPropertyNames** () const
Gets an Vector of the CMSX property names.

6.18.1 Detailed Description

This class houses all the various settings and information that is used by an instance of an **ActiveMQConnection** (p. 187) class.

Since

3.0

6.18.2 Constructor & Destructor Documentation

6.18.2.1 `activemq::core::ActiveMQConnectionMetaData::ActiveMQConnectionMetaData ()`

6.18.2.2 `virtual activemq::core::ActiveMQConnectionMetaData::~ActiveMQConnectionMetaData () throw ()`
[virtual]

6.18.3 Member Function Documentation

6.18.3.1 `virtual int activemq::core::ActiveMQConnectionMetaData::getCMSMajorVersion () const` [virtual]

Gets the CMS major version number.

Returns

the CMS API major version number

Exceptions

<i>CMSException</i>	If the CMS Provider fails to retrieve the metadata due to some internal error.
---------------------	--

Implements `cms::ConnectionMetaData` (p. 979).

6.18.3.2 `virtual int activemq::core::ActiveMQConnectionMetaData::getCMSMinorVersion () const` [virtual]

Gets the CMS minor version number.

Returns

the CMS API minor version number

Exceptions

<i>CMSException</i>	If the CMS Provider fails to retrieve the metadata due to some internal error.
---------------------	--

Implements `cms::ConnectionMetaData` (p. 979).

6.18.3.3 `virtual std::string activemq::core::ActiveMQConnectionMetaData::getCMSProviderName () const` [virtual]

Gets the CMS provider name.

Returns

the CMS provider name

Exceptions

<i>CMSException</i>	If the CMS Provider fails to retrieve the metadata due to some internal error.
---------------------	--

Implements **cms::ConnectionMetaData** (p. 980).

6.18.3.4 `virtual std::string activemq::core::ActiveMQConnectionMetaData::getCMSVersion () const [virtual]`

Gets the CMS API version.

Returns

the CMS API Version in String form.

Exceptions

<i>CMSException</i>	If the CMS Provider fails to retrieve the metadata due to some internal error.
---------------------	--

Implements **cms::ConnectionMetaData** (p. 980).

6.18.3.5 `virtual std::vector<std::string> activemq::core::ActiveMQConnectionMetaData::getCMSXPropertyNames () const [virtual]`

Gets an Vector of the CMSX property names.

Returns

an Vector of CMSX property names

Exceptions

<i>CMSException</i>	If the CMS Provider fails to retrieve the metadata due to some internal error.
---------------------	--

Implements **cms::ConnectionMetaData** (p. 980).

6.18.3.6 `virtual int activemq::core::ActiveMQConnectionMetaData::getProviderMajorVersion () const` [virtual]

Gets the CMS provider major version number.

Returns

the CMS provider major version number

Exceptions

<i>CMSException</i>	If the CMS Provider fails to retrieve the metadata due to some internal error.
---------------------	--

Implements **cms::ConnectionMetaData** (p. 981).

6.18.3.7 `virtual int activemq::core::ActiveMQConnectionMetaData::getProviderMinorVersion () const` [virtual]

Gets the CMS provider minor version number.

Returns

the CMS provider minor version number

Exceptions

<i>CMSException</i>	If the CMS Provider fails to retrieve the metadata due to some internal error.
---------------------	--

Implements **cms::ConnectionMetaData** (p. 981).

6.18.3.8 `virtual std::string activemq::core::ActiveMQConnectionMetaData::getProviderVersion () const` [virtual]

Gets the CMS provider version.

Returns

the CMS provider version

Exceptions

<i>CMSException</i>	If the CMS Provider fails to retrieve the metadata due to some internal error.
---------------------	--

Implements **cms::ConnectionMetaData** (p. 981).

The documentation for this class was generated from the following file:

- src/main/activemq/core/**ActiveMQConnectionMetaData.h**

6.19 activemq::core::ActiveMQConstants Class Reference

Class holding constant values for various ActiveMQ specific things Each constant is defined as an enumeration and has functions that convert back and forth between string and enum values.

```
#include <src/main/activemq/core/ActiveMQConstants.h>
```

Data Structures

- class **StaticInitializer**

Public Types

- enum **TransactionState** { TRANSACTION_STATE_BEGIN = 0, TRANSACTION_STATE_PREPARE = 1, TRANSACTION_STATE_COMMITONEPHASE = 2, TRANSACTION_STATE_COMMITTWO PHASE = 3, TRANSACTION_STATE_ROLLBACK = 4, TRANSACTION_STATE_RECOVER = 5, TRANSACTION_STATE_FORGET = 6, TRANSACTION_STATE_END = 7 }
- enum **DestinationActions** { DESTINATION_ADD_OPERATION = 0, DESTINATION_REMOVE_OPERATION = 1 }
- enum **AckType** { ACK_TYPE_DELIVERED = 0, ACK_TYPE_POISON = 1, ACK_TYPE_CONSUMED = 2, ACK_TYPE_REDELIVERED = 3, ACK_TYPE_INDIVIDUAL = 4 }
- enum **DestinationOption** { CONSUMER_PREFETCHSIZE, CONSUMER_MAXPENDINGMSGLIMIT, CONSUMER_NOLOCAL, CONSUMER_DISPATCHASYNC, CONSUMER_RETROACTIVE, CONSUMER_SELECTOR, CONSUMER_EXCLUSIVE, CONSUMER_PRIORITY, NUM_OPTIONS }

These values represent the options that can be appended to an Destination name, i.e.

- enum **URIParam** { CONNECTION_SENDTIMEOUT, CONNECTION_PRODUCERWINDOWSIZE, CONNECTION_CLOSETIMEOUT, CONNECTION_ALWAYSSEND, CONNECTION_USEASYNCSEND, CONNECTION_USECOMPRESSION, CONNECTION_DISPATCHASYNC, PARAM_USERNAME, PARAM_PASSWORD, PARAM_CLIENTID, NUM_PARAMS }

These values represent the parameters that can be added to the connection URI that affect the ActiveMQ Core API.

Static Public Member Functions

- static const std::string & **toString** (const **DestinationOption** option)
- static **DestinationOption** **toDestinationOption** (const std::string &option)

- static const std::string & **toString** (const **URIParam** option)
- static **URIParam toURIOption** (const std::string &option)

6.19.1 Detailed Description

Class holding constant values for various ActiveMQ specific things Each constant is defined as an enumeration and has functions that convert back and forth between string and enum values.

6.19.2 Member Enumeration Documentation

6.19.2.1 enum activemq::core::ActiveMQConstants::AckType

Enumerator:

ACK_TYPE_DELIVERED
ACK_TYPE_POISON
ACK_TYPE_CONSUMED
ACK_TYPE_REDELIVERED
ACK_TYPE_INDIVIDUAL

6.19.2.2 enum activemq::core::ActiveMQConstants::DestinationActions

Enumerator:

DESTINATION_ADD_OPERATION
DESTINATION_REMOVE_OPERATION

6.19.2.3 enum activemq::core::ActiveMQConstants::DestinationOption

These values represent the options that can be appended to an Destination name, i.e.

/topic/foo?consumer.exclusive=true

Enumerator:

CONSUMER_PREFETCHSIZE
CUNSUMER_MAXPENDINGMSGLIMIT
CONSUMER_NOLOCAL
CONSUMER_DISPATCHASYNC
CONSUMER_RETROACTIVE
CONSUMER_SELECTOR
CONSUMER_EXCLUSIVE
CONSUMER_PRIORITY
NUM_OPTIONS

6.19.2.4 enum activemq::core::ActiveMQConstants::TransactionState

Enumerator:

TRANSACTION_STATE_BEGIN
TRANSACTION_STATE_PREPARE
TRANSACTION_STATE_COMMITONEPHASE
TRANSACTION_STATE_COMMITTWOPHASE
TRANSACTION_STATE_ROLLBACK
TRANSACTION_STATE_RECOVER
TRANSACTION_STATE_FORGET
TRANSACTION_STATE_END

6.19.2.5 enum activemq::core::ActiveMQConstants::URIParam

These values represent the parameters that can be added to the connection URI that affect the ActiveMQ Core API.

Enumerator:

CONNECTION_SENDTIMEOUT
CONNECTION_PRODUCERWINDOWSIZE
CONNECTION_CLOSETIMEOUT
CONNECTION_ALWAYSASYNCSEND
CONNECTION_USEASYNCSEND
CONNECTION_USECOMPRESSION
CONNECTION_DISPATCHASYNC
PARAM_USERNAME
PARAM_PASSWORD
PARAM_CLIENTID
NUM_PARAMS

6.19.3 Member Function Documentation

6.19.3.1 static DestinationOption activemq::core::ActiveMQConstants::toDestinationOption (const std::string & *option*) [inline, static]

6.19.3.2 static const std::string& activemq::core::ActiveMQConstants::toString (const DestinationOption *option*) [inline, static]

6.19.3.3 static const std::string& activemq::core::ActiveMQConstants::toString (const URIParam *option*) [inline, static]

6.19.3.4 static `URIParam activemq::core::ActiveMQConstants::toURIOption (const std::string & option)` [inline, static]

The documentation for this class was generated from the following file:

- `src/main/activemq/core/ActiveMQConstants.h`

6.20 activemq::core::ActiveMQConsumer Class Reference

```
#include <src/main/activemq/core/ActiveMQConsumer.h>
```

Inheritance diagram for `activemq::core::ActiveMQConsumer`:

Public Member Functions

- **ActiveMQConsumer** (**ActiveMQSession** *session, const **Pointer**< **commands::ConsumerId** > &id, const **Pointer**< **commands::ActiveMQDestination** > &destination, const std::string &name, const std::string &selector, int prefetch, int maxPendingMessageCount, bool noLocal, bool browser, bool dispatchAsync, **cms::MessageListener** *listener)
Constructor.
- virtual **~ActiveMQConsumer** () throw ()
- virtual void **start** ()
Starts the service.
- virtual void **stop** ()
Stops this service.
- virtual void **close** ()
Closes this object and deallocates the appropriate resources.
- virtual **cms::Message** * **receive** ()
Synchronously Receive a Message.
- virtual **cms::Message** * **receive** (int millisecs)
Synchronously Receive a Message, time out after defined interval.
- virtual **cms::Message** * **receiveNoWait** ()
Receive a Message, does not wait if there isn't a new message to read, returns NULL if nothing read.
- virtual void **setMessageListener** (**cms::MessageListener** *listener)
Sets the MessageListener that this class will send notifs on.
- virtual **cms::MessageListener** * **getMessageListener** () const
Gets the MessageListener that this class will send mew Message notification events to.
- virtual std::string **getMessageSelector** () const
Gets this message consumer's message selector expression.

- virtual void **acknowledge** (const **Pointer**< **commands::MessageDispatch** > &**dispatch**)
- virtual void **dispatch** (const **Pointer**< **MessageDispatch** > &message)
Dispatches a message to a particular consumer.
- void **acknowledge** ()
Method called to acknowledge all messages that have been received so far.
- void **commit** ()
Called to Commit the current set of messages in this Transaction.
- void **rollback** ()
Called to Roll back the current set of messages in this Transaction.
- void **doClose** ()
Performs the actual close operation on this consumer.
- void **dispose** ()
Cleans up this objects internal resources.
- const **Pointer** < **commands::ConsumerInfo** > &**getConsumerInfo** () const
Get the Consumer information for this consumer.
- const **Pointer** < **commands::ConsumerId** > &**getConsumerId** () const
Get the Consumer Id for this consumer.
- bool **isClosed** () const
- bool **isSynchronizationRegistered** () const
*Has this Consumer Transaction **Synchronization** (p. 2654) been added to the transaction.*
- void **setSynchronizationRegistered** (bool value)
*Sets the **Synchronization** (p. 2654) Registered state of this consumer.*
- bool **iterate** ()
Deliver any pending messages to the registered MessageListener if there is one, return true if not all dispatched, or false if no listener or all pending messages have been dispatched.
- void **deliverAcks** ()
Forces this consumer to send all pending acks to the broker.
- void **clearMessagesInProgress** ()
Called on a Failover to clear any pending messages.
- void **inProgressClearRequired** ()
Signals that a Failure occurred and that anything in-progress in the consumer should be cleared.
- long long **getLastDeliveredSequenceId** () const
Gets the currently set Last Delivered Sequence Id.
- void **setLastDeliveredSequenceId** (long long value)
Sets the value of the Last Delivered Sequence Id.
- int **getMessageAvailableCount** () const
- void **setRedeliveryPolicy** (**RedeliveryPolicy** *policy)
*Sets the **RedeliveryPolicy** (p. 2250) this Consumer should use when a rollback is performed on a transacted Consumer.*
- **RedeliveryPolicy** * **getRedeliveryPolicy** () const

Gets a pointer to this Consumer's Redelivery Policy object, the Consumer retains ownership of this pointer so the caller should not delete it.

- void **setFailureError** (**decaf::lang::Exception** *error)

Sets the Exception that has caused this Consumer to be in a failed state.

- **decaf::lang::Exception** * **getFailureError** () const

Gets the error that caused this Consumer to be in a Failed state, or NULL if there is no Error.

Protected Member Functions

- **Pointer< MessageDispatch > dequeue** (long long timeout)

Used by synchronous receive methods to wait for messages to come in.

- void **beforeMessagesConsumed** (const **Pointer< commands::MessageDispatch > &dispatch**)

Pre-consume processing.

- void **afterMessagesConsumed** (const **Pointer< commands::MessageDispatch > &dispatch**, bool messageExpired)

Post-consume processing.

6.20.1 Constructor & Destructor Documentation

- 6.20.1.1 **activemq::core::ActiveMQConsumer::ActiveMQConsumer** (**ActiveMQSession** * session, const **Pointer< commands::ConsumerId > &id**, const **Pointer< commands::ActiveMQDestination > &destination**, const std::string & name, const std::string & selector, int prefetch, int maxPendingMessageCount, bool noLocal, bool browser, bool dispatchAsync, cms::MessageListener * listener)

Constructor.

- 6.20.1.2 **virtual activemq::core::ActiveMQConsumer::~~ActiveMQConsumer** ()
throw () [virtual]

6.20.2 Member Function Documentation

- 6.20.2.1 **virtual void activemq::core::ActiveMQConsumer::acknowledge** (const **Pointer< commands::MessageDispatch > &dispatch**) [virtual]

- 6.20.2.2 **void activemq::core::ActiveMQConsumer::acknowledge** ()

Method called to acknowledge all messages that have been received so far.

Exceptions

CMSException	if an error occurs while ack'ing the message.
---------------------	---

6.20.2.3 `void activemq::core::ActiveMQConsumer::afterMessagesConsumed
(const Pointer< commands::MessageDispatch > & dispatch, bool
messageExpired) [protected]`

Post-consume processing.

Parameters

<i>dispatch</i>	- the consumed message
<i>message-Expired</i>	- flag indicating if the message has expired.

6.20.2.4 `void activemq::core::ActiveMQConsumer::beforeMessagesConsumed
(const Pointer< commands::MessageDispatch > & dispatch)
[protected]`

Pre-consume processing.

Parameters

<i>dispatch</i>	- the message being consumed.
-----------------	-------------------------------

6.20.2.5 `void activemq::core::ActiveMQConsumer::clearMessagesInProgress ()`

Called on a Failover to clear any pending messages.

6.20.2.6 `virtual void activemq::core::ActiveMQConsumer::close () [virtual]`

Closes this object and deallocates the appropriate resources.

The object is generally no longer usable after calling close.

Exceptions

<i>CMSException</i>	- If an error occurs while the resource is being closed.
---------------------	--

Implements **cms::Closeable** (p. 816).

6.20.2.7 `void activemq::core::ActiveMQConsumer::commit ()`

Called to Commit the current set of messages in this Transaction.

Exceptions

<i>ActiveMQException</i>	if an error occurs while performing the operation.
--------------------------	--

6.20.2.8 void activemq::core::ActiveMQConsumer::deliverAcks ()

Forces this consumer to send all pending acks to the broker.

Exceptions

<i>ActiveMQException</i>	if an error occurs while performing the operation.
--------------------------	--

6.20.2.9 Pointer<MessageDispatch> activemq::core::ActiveMQConsumer::dequeue (long long timeout) [protected]

Used by synchronous receive methods to wait for messages to come in.

Parameters

<i>timeout</i>	- The maximum number of milliseconds to wait before returning.
----------------	--

If -1, it will block until a messages is received or this consumer is closed. If 0, will not block at all. If > 0, will wait at a maximum the specified number of milliseconds before returning.

Returns

the message, if received within the allotted time. Otherwise NULL.

Exceptions

<i>InvalidStateException</i>	if this consumer is closed upon entering this method.
------------------------------	---

6.20.2.10 virtual void activemq::core::ActiveMQConsumer::dispatch (const Pointer< MessageDispatch > & message) [virtual]

Dispatches a message to a particular consumer.

Parameters

<i>message</i>	The message to be dispatched to a waiting consumer.
----------------	---

Implements **activemq::core::Dispatcher** (p. 1235).

6.20.2.11 void activemq::core::ActiveMQConsumer::dispose ()

Cleans up this objects internal resources.

Exceptions

<i>ActiveMQException</i>	if an error occurs while performing the operation.
--------------------------	--

6.20.2.12 void **activemq::core::ActiveMQConsumer::doClose** ()

Performs the actual close operation on this consumer.

Exceptions

<i>ActiveMQException</i>	if an error occurs while performing the operation.
--------------------------	--

6.20.2.13 const **Pointer<commands::ConsumerId>& activemq-
::core::ActiveMQConsumer::getConsumerId** ()
const

Get the Consumer Id for this consumer.

Returns

Reference to a Consumer Id Object

6.20.2.14 const **Pointer<commands::ConsumerInfo>& activemq-
::core::ActiveMQConsumer::getConsumerInfo** ()
const

Get the Consumer information for this consumer.

Returns

Reference to a Consumer Info Object

6.20.2.15 **decaf::lang::Exception* activemq::core::ActiveMQConsumer::get-
FailureError** () const

Gets the error that caused this Consumer to be in a Failed state, or NULL if there is no Error.

Returns

pointer to the error that faulted this Consumer or NULL.

6.20.2.16 `long long activemq::core::ActiveMQConsumer::getLastDeliveredSequenceId () const`

Gets the currently set Last Delivered Sequence Id.

Returns

long long containing the sequence id of the last delivered Message.

6.20.2.17 `int activemq::core::ActiveMQConsumer::getMessageAvailableCount () const`

Returns

the number of Message's this consumer is waiting to Dispatch.

6.20.2.18 `virtual cms::MessageListener* activemq::core::ActiveMQConsumer::getMessageListener () const [virtual]`

Gets the MessageListener that this class will send new Message notification events to.

Returns

The listener of messages received by this consumer

Exceptions

<i>CMSEException</i>	- If an internal error occurs.
----------------------	--------------------------------

Implements **cms::MessageConsumer** (p. 1878).

6.20.2.19 `virtual std::string activemq::core::ActiveMQConsumer::getMessageSelector () const [virtual]`

Gets this message consumer's message selector expression.

Returns

This Consumer's selector expression or "".

Exceptions

<i>CMSEException</i>	- If an internal error occurs.
----------------------	--------------------------------

Implements **cms::MessageConsumer** (p. 1878).

6.20.2.20 RedeliveryPolicy* activemq::core::ActiveMQConsumer::getRedeliveryPolicy () const

Gets a pointer to this Consumer's Redelivery Policy object, the Consumer retains ownership of this pointer so the caller should not delete it.

Returns

a Pointer to a **RedeliveryPolicy** (p. 2250) that is in use by this Consumer.

6.20.2.21 void activemq::core::ActiveMQConsumer::inProgressClearRequired ()

Signals that a Failure occurred and that anything in-progress in the consumer should be cleared.

6.20.2.22 bool activemq::core::ActiveMQConsumer::isClosed () const**Returns**

if this Consumer has been closed.

6.20.2.23 bool activemq::core::ActiveMQConsumer::isSynchronizationRegistered () const

Has this Consumer Transaction **Synchronization** (p. 2654) been added to the transaction.

Returns

true if the synchronization has been added.

6.20.2.24 bool activemq::core::ActiveMQConsumer::iterate ()

Deliver any pending messages to the registered MessageListener if there is one, return true if not all dispatched, or false if no listener or all pending messages have been dispatched.

**6.20.2.25 virtual cms::Message* activemq::core::ActiveMQConsumer::receive ()
[virtual]**

Synchronously Receive a Message.

Returns

new message which the caller owns and must delete.

Exceptions

<i>CMSEException</i>	- If an internal error occurs.
----------------------	--------------------------------

Implements **cms::MessageConsumer** (p. 1879).

6.20.2.26 `virtual cms::Message* activemq::core::ActiveMQConsumer::receive (int
milliseconds) [virtual]`

Synchronously Receive a Message, time out after defined interval.

Returns null if nothing read.

Returns

new message which the caller owns and must delete.

Exceptions

<i>CMSEException</i>	- If an internal error occurs.
----------------------	--------------------------------

Implements **cms::MessageConsumer** (p. 1879).

6.20.2.27 `virtual cms::Message* activemq::core::ActiveMQConsumer::receiveNo-
Wait () [virtual]`

Receive a Message, does not wait if there isn't a new message to read, returns NULL if nothing read.

Returns

new message which the caller owns and must delete.

Exceptions

<i>CMSEException</i>	- If an internal error occurs.
----------------------	--------------------------------

Implements **cms::MessageConsumer** (p. 1880).

6.20.2.28 `void activemq::core::ActiveMQConsumer::rollback ()`

Called to Roll back the current set of messages in this Transaction.

Exceptions

<i>ActiveMQException</i>	if an error occurs while performing the operation.
--------------------------	--

6.20.2.29 void **activemq::core::ActiveMQConsumer::setFailureError** (
 decaf::lang::Exception * error)

Sets the Exception that has caused this Consumer to be in a failed state.

Parameters

<i>error</i>	The error that is to be thrown when a Receive call is made.
--------------	---

6.20.2.30 void **activemq::core::ActiveMQConsumer::setLastDeliveredSequenceId** (
 long long value)

Sets the value of the Last Delivered Sequence Id.

Parameters

<i>value</i>	The new value to assign to the Last Delivered Sequence Id property.
--------------	---

6.20.2.31 virtual void **activemq::core::ActiveMQConsumer::setMessageListener** (
 cms::MessageListener * listener) [virtual]

Sets the MessageListener that this class will send notifs on.

Parameters

<i>listener</i>	The listener of messages received by this consumer.
-----------------	---

Exceptions

<i>CMSException</i>	- If an internal error occurs.
---------------------	--------------------------------

Implements **cms::MessageConsumer** (p. 1880).

6.20.2.32 void **activemq::core::ActiveMQConsumer::setRedeliveryPolicy** (
 RedeliveryPolicy * policy)

Sets the **RedeliveryPolicy** (p. 2250) this Consumer should use when a rollback is performed on a transacted Consumer.

The Consumer takes ownership of the passed pointer. The Consumer's redelivery policy can never be null, a call to this method with a NULL pointer is ignored.

Parameters

<i>policy</i>	Pointer to a Redelivery Policy object that his Consumer will use.
---------------	---

6.20.2.33 `void activemq::core::ActiveMQConsumer::setSynchronization-Registered (bool value)`

Sets the **Synchronization** (p. 2654) Registered state of this consumer.

Parameters

<i>value</i>	- true if registered false otherwise.
--------------	---------------------------------------

6.20.2.34 `virtual void activemq::core::ActiveMQConsumer::start () [virtual]`

Starts the service.

Exceptions

<i>CMSEException</i>	if an internal error occurs while starting.
----------------------	---

Implements **cms::Startable** (p. 2534).

6.20.2.35 `virtual void activemq::core::ActiveMQConsumer::stop () [virtual]`

Stops this service.

Exceptions

<i>CMSEException</i>	- if an internal error occurs while stopping the Service.
----------------------	---

Implements **cms::Stoppable** (p. 2602).

The documentation for this class was generated from the following file:

- `src/main/activemq/core/ActiveMQConsumer.h`

6.21 `activemq::library::ActiveMQCPP` Class Reference

```
#include <src/main/activemq/library/ActiveMQCPP.h>
```

Public Member Functions

- `virtual ~ActiveMQCPP ()`

Static Public Member Functions

- `static void initializeLibrary ()`

Initialize the ActiveMQ-CPP Library constructs, this method will init all the internal - Registry objects and initialize the Decaf library.

- static void **initializeLibrary** (int argc, char **argv)

Initialize the ActiveMQ-CPP Library constructs, this method will initialize all the internal Registry objects and initialize the Decaf library.

- static void **shutdownLibrary** ()

Shutdown the ActiveMQ-CPP Library, freeing any resources that could not be freed up to this point.

Protected Member Functions

- **ActiveMQCPP** ()
- **ActiveMQCPP** (const **ActiveMQCPP** &)
- **ActiveMQCPP** & **operator=** (const **ActiveMQCPP** &)

6.21.1 Constructor & Destructor Documentation

6.21.1.1 **activemq::library::ActiveMQCPP::ActiveMQCPP** () [inline, protected]

6.21.1.2 **activemq::library::ActiveMQCPP::ActiveMQCPP** (const **ActiveMQCPP** &) [protected]

6.21.1.3 **virtual activemq::library::ActiveMQCPP::~~ActiveMQCPP** () [inline, virtual]

6.21.2 Member Function Documentation

6.21.2.1 **static void activemq::library::ActiveMQCPP::initializeLibrary** () [static]

Initialize the ActiveMQ-CPP Library constructs, this method will init all the internal - Registry objects and initialize the Decaf library.

Exceptions

<i>runtime_error</i>	if an error occurs while initializing this library.
----------------------	---

6.21.2.2 **static void activemq::library::ActiveMQCPP::initializeLibrary** (int argc, char ** argv) [static]

Initialize the ActiveMQ-CPP Library constructs, this method will initialize all the internal Registry objects and initialize the Decaf library.

This method takes the args passed to the main method of process for use is setting system properties and configuring the ActiveMQ-CPP Library.

Parameters

<i>argc</i>	- the count of arguments passed to this Process.
<i>argv</i>	- the array of string arguments passed to this process.

Exceptions

<i>runtime_error</i>	if an error occurs while initializing this library.
----------------------	---

6.21.2.3 **ActiveMQCPP& activemq::library::ActiveMQCPP::operator= (const ActiveMQCPP &)** *[protected]*

6.21.2.4 **static void activemq::library::ActiveMQCPP::shutdownLibrary ()** *[static]*

Shutdown the ActiveMQ-CPP Library, freeing any resources that could not be freed up to this point.

All the user created ActiveMQ-CPP objects and Decaf Library objects should have been destroyed by the time this method is called.

The documentation for this class was generated from the following file:

- src/main/activemq/library/**ActiveMQCPP.h**

6.22 activemq::commands::ActiveMQDestination Class Reference

```
#include <src/main/activemq/commands/ActiveMQDestination.h>
```

Inheritance diagram for activemq::commands::ActiveMQDestination:

Data Structures

- struct **DestinationFilter**

Public Member Functions

- **ActiveMQDestination ()**
- **ActiveMQDestination (const std::string &physicalName)**
- virtual **~ActiveMQDestination ()** throw ()
- virtual **ActiveMQDestination * cloneDataStructure ()** const
Clone this obbject and return a new instance that the caller now owns, this will be an exact copy of this one.

- virtual void **copyDataStructure** (const **DataStructure** *src)
Copy the contents of the passed object into this objects members, overwriting any existing data.
- virtual bool **equals** (const **DataStructure** *value) const
*Compares the **DataStructure** (p. 1133) passed in to this one, and returns if they are equivalent.*
- virtual unsigned char **getDataStructureType** () const
*Get the **DataStructure** (p. 1133) Type as defined in CommandTypes.h.*
- virtual std::string **toString** () const
*Returns a string containing the information for this **DataStructure** (p. 1133) such as its type and value of its elements.*
- virtual const std::string & **getPhysicalName** () const
Fetch this destination's physical name.
- virtual std::string & **getPhysicalName** ()
- virtual void **setPhysicalName** (const std::string &physicalName)
Set this destination's physical name.
- virtual bool **isAdvisory** () const
- virtual void **setAdvisory** (bool advisory)
- virtual bool **isConsumerAdvisory** () const
- virtual bool **isProducerAdvisory** () const
- virtual bool **isConnectionAdvisory** () const
- virtual bool **isExclusive** () const
- virtual void **setExclusive** (bool exclusive)
- virtual bool **isOrdered** () const
- virtual void **setOrdered** (bool ordered)
- virtual std::string **getOrderedTarget** () const
- virtual void **setOrderedTarget** (const std::string &orderedTarget)
- virtual **cms::Destination::DestinationType** **getDestinationType** () const =0
Returns the Type of Destination that this object represents.
- virtual bool **isTemporary** () const
Returns true if a temporary Destination.
- virtual bool **isTopic** () const
Returns true if a Topic Destination.
- virtual bool **isQueue** () const
Returns true if a Queue Destination.
- virtual bool **isComposite** () const
Returns true if this destination represents a collection of destinations; allowing a set of destinations to be published to or subscribed from in one CMS operation.
- virtual bool **isWildcard** () const
- const **activemq::util::ActiveMQProperties** & **getOptions** () const
- virtual const **cms::Destination** * **getCMSDestination** () const

Static Public Member Functions

- static std::string **createTemporaryName** (const std::string &clientId)
Create a temporary name from the clientId.
- static std::string **getClientId** (const **ActiveMQDestination** *destination)
From a temporary destination find the clientId of the Connection that created it.
- static **Pointer** < **ActiveMQDestination** > **createDestination** (int type, const std::string &name)
Creates a Destination given the String Name to use and a Type.

Static Public Attributes

- static const unsigned char **ID_ACTIVEMQDESTINATION** = 0

Protected Attributes

- bool **exclusive**
- bool **ordered**
- bool **advisory**
- std::string **orderedTarget**
- std::string **physicalName**
- **util::ActiveMQProperties** **options**

Static Protected Attributes

- static const std::string **ADVISORY_PREFIX**
prefix for Advisory message destinations
- static const std::string **CONSUMER_ADVISORY_PREFIX**
prefix for consumer advisory destinations
- static const std::string **PRODUCER_ADVISORY_PREFIX**
prefix for producer advisory destinations
- static const std::string **CONNECTION_ADVISORY_PREFIX**
prefix for connection advisory destinations
- static const std::string **DEFAULT_ORDERED_TARGET**
The default target for ordered destinations.
- static const std::string **TEMP_PREFIX**
- static const std::string **TEMP_POSTFIX**
- static const std::string **COMPOSITE_SEPARATOR**
- static const std::string **QUEUE_QUALIFIED_PREFIX**
- static const std::string **TOPIC_QUALIFIED_PREFIX**
- static const std::string **TEMP_QUEUE_QUALIFIED_PREFIX**
- static const std::string **TEMP_TOPIC_QUALIFIED_PREFIX**

6.22.1 Constructor & Destructor Documentation

6.22.1.1 `activemq::commands::ActiveMQDestination::ActiveMQDestination ()`

6.22.1.2 `activemq::commands::ActiveMQDestination::ActiveMQDestination (const std::string & physicalName)`

6.22.1.3 `virtual activemq::commands::ActiveMQDestination::~~ActiveMQDestination () throw ()` `[virtual]`

6.22.2 Member Function Documentation

6.22.2.1 `virtual ActiveMQDestination* activemq::commands::ActiveMQDestination::cloneDataStructure () const` `[inline, virtual]`

Clone this object and return a new instance that the caller now owns, this will be an exact copy of this one.

Returns

new copy of this object.

Implements `activemq::commands::DataStructure` (p. 1133).

Reimplemented in `activemq::commands::ActiveMQTempDestination` (p. 380), `activemq::commands::ActiveMQQueue` (p. 322), `activemq::commands::ActiveMQTopic` (p. 416), `activemq::commands::ActiveMQTempQueue` (p. 388), and `activemq::commands::ActiveMQTempTopic` (p. 397).

6.22.2.2 `virtual void activemq::commands::ActiveMQDestination::copyDataStructure (const DataStructure * src)` `[virtual]`

Copy the contents of the passed object into this objects members, overwriting any existing data.

Returns

src - Source Object

Implements `activemq::commands::DataStructure` (p. 1134).

Reimplemented in `activemq::commands::ActiveMQTempDestination` (p. 380), `activemq::commands::ActiveMQQueue` (p. 323), `activemq::commands::ActiveMQTopic` (p. 416), `activemq::commands::ActiveMQTempQueue` (p. 388), and `activemq::commands::ActiveMQTempTopic` (p. 398).

Referenced by `activemq::commands::ActiveMQTempDestination::copyDataStructure()`.

6.22.2.3 `static Pointer<ActiveMQDestination> activemq::commands::ActiveMQDestination::createDestination (int type, const std::string & name)`
`[static]`

Creates a Destination given the String Name to use and a Type.

Parameters

<i>type</i>	- The Type of Destination to Create
<i>name</i>	- The Name to use in the creation of the Destination

Returns

Pointer to a new **ActiveMQDestination** (p. 246) instance.

6.22.2.4 `static std::string activemq::commands::ActiveMQDestination::createTemporaryName (const std::string & clientId)` `[inline, static]`

Create a temporary name from the clientId.

Parameters

<i>clientId</i>	
-----------------	--

Returns

6.22.2.5 `virtual bool activemq::commands::ActiveMQDestination::equals (const DataStructure * value) const` `[virtual]`

Compares the **DataStructure** (p. 1133) passed in to this one, and returns if they are equivalent.

Equivalent here means that they are of the same type, and that each element of the objects are the same.

Returns

true if **DataStructure** (p. 1133)'s are Equal.

Implements **activemq::commands::DataStructure** (p. 1135).

Reimplemented in **activemq::commands::ActiveMQTempDestination** (p. 381), **activemq::commands::ActiveMQQueue** (p. 323), **activemq::commands::ActiveMQTopic** (p. 416), **activemq::commands::ActiveMQTempQueue** (p. 389), and **activemq::commands::ActiveMQTempTopic** (p. 398).

Referenced by `activemq::commands::ActiveMQTempDestination::equals()`.

6.22.2.6 `static std::string activemq::commands::ActiveMQDestination::getClientId (const ActiveMQDestination * destination) [static]`

From a temporary destination find the clientId of the Connection that created it.

Parameters

<i>destination</i>	
--------------------	--

Returns

the clientId or null if not a temporary destination

6.22.2.7 `virtual const cms::Destination* activemq::commands::ActiveMQDestination::getCMSDestination () const [inline, virtual]`

Returns

the **cms::Destination** (p. 1210) interface pointer that the objects that derive from this class implement.

Reimplemented in **activemq::commands::ActiveMQQueue** (p. 324), **activemq::commands::ActiveMQTopic** (p. 417), **activemq::commands::ActiveMQTempQueue** (p. 389), and **activemq::commands::ActiveMQTempTopic** (p. 399).

6.22.2.8 `virtual unsigned char activemq::commands::ActiveMQDestination::getDataStructureType () const [virtual]`

Get the **DataStructure** (p. 1133) Type as defined in CommandTypes.h.

Returns

The type of the data structure

Implements **activemq::commands::DataStructure** (p. 1137).

Reimplemented in **activemq::commands::ActiveMQTempDestination** (p. 381), **activemq::commands::ActiveMQQueue** (p. 324), **activemq::commands::ActiveMQTopic** (p. 417), **activemq::commands::ActiveMQTempQueue** (p. 390), and **activemq::commands::ActiveMQTempTopic** (p. 399).

6.22.2.9 `virtual cms::Destination::DestinationType activemq::commands::ActiveMQDestination::getDestinationType () const [pure virtual]`

Returns the Type of Destination that this object represents.

Returns

int type qualifier.

Implemented in `activemq::commands::ActiveMQQueue` (p. 325), `activemq::commands::ActiveMQTopic` (p. 418), `activemq::commands::ActiveMQTempQueue` (p. 390), and `activemq::commands::ActiveMQTempTopic` (p. 400).

6.22.2.10 `const activemq::util::ActiveMQProperties& activemq::commands::ActiveMQDestination::getOptions () const` `[inline]`

Returns

a reference (const) to the options properties for this Destination.

6.22.2.11 `virtual std::string activemq::commands::ActiveMQDestination::getOrderedTarget () const` `[inline, virtual]`

Returns

Returns the orderedTarget.

6.22.2.12 `virtual const std::string& activemq::commands::ActiveMQDestination::getPhysicalName () const` `[inline, virtual]`

Fetch this destination's physical name.

Returns

const string containing the name

6.22.2.13 `virtual std::string& activemq::commands::ActiveMQDestination::getPhysicalName ()` `[inline, virtual]`

6.22.2.14 `virtual bool activemq::commands::ActiveMQDestination::isAdvisory () const` `[inline, virtual]`

Returns

Returns the advisory.

6.22.2.15 `virtual bool activemq::commands::ActiveMQDestination::isComposite () const [inline, virtual]`

Returns true if this destination represents a collection of destinations; allowing a set of destinations to be published to or subscribed from in one CMS operation.

Returns

true if this destination represents a collection of child destinations.

6.22.2.16 `virtual bool activemq::commands::ActiveMQDestination::isConnectionAdvisory () const [inline, virtual]`

Returns

true if this is a destination for Connection advisories

6.22.2.17 `virtual bool activemq::commands::ActiveMQDestination::isConsumerAdvisory () const [inline, virtual]`

Returns

true if this is a destination for Consumer advisories

6.22.2.18 `virtual bool activemq::commands::ActiveMQDestination::isExclusive () const [inline, virtual]`

Returns

Returns the exclusive.

6.22.2.19 `virtual bool activemq::commands::ActiveMQDestination::isOrdered () const [inline, virtual]`

Returns

Returns the ordered.

6.22.2.20 `virtual bool activemq::commands::ActiveMQDestination::isProducerAdvisory () const [inline, virtual]`

Returns

true if this is a destination for Producer advisories

6.22.2.21 `virtual bool activemq::commands::ActiveMQDestination::isQueue ()`
`const [inline, virtual]`

Returns true if a Queue Destination.

Returns

true/false

6.22.2.22 `virtual bool activemq::commands::ActiveMQDestination::isTemporary ()`
`const [inline, virtual]`

Returns true if a temporary Destination.

Returns

true/false

References cms::Destination::TEMPORARY_QUEUE, and cms::Destination::TEMPORARY_TOPIC.

6.22.2.23 `virtual bool activemq::commands::ActiveMQDestination::isTopic () const`
`[inline, virtual]`

Returns true if a Topic Destination.

Returns

true/false

References cms::Destination::TEMPORARY_TOPIC, and cms::Destination::TOPIC.

6.22.2.24 `virtual bool activemq::commands::ActiveMQDestination::isWildcard ()`
`const [inline, virtual]`

Returns

true if the destination matches multiple possible destinations

6.22.2.25 `virtual void activemq::commands::ActiveMQDestination::setAdvisory (`
`bool advisory) [inline, virtual]`

Parameters

<i>advisory</i>	The advisory to set.
-----------------	----------------------

6.22.2.26 `virtual void activemq::commands::ActiveMQDestination::setExclusive (bool exclusive) [inline, virtual]`

Parameters

<i>exclusive</i>	The exclusive to set.
------------------	-----------------------

6.22.2.27 `virtual void activemq::commands::ActiveMQDestination::setOrdered (bool ordered) [inline, virtual]`

Parameters

<i>ordered</i>	The ordered to set.
----------------	---------------------

6.22.2.28 `virtual void activemq::commands::ActiveMQDestination::setOrderedTarget (const std::string & orderedTarget) [inline, virtual]`

Parameters

<i>ordered-Target</i>	The orderedTarget to set.
-----------------------	---------------------------

6.22.2.29 `virtual void activemq::commands::ActiveMQDestination::setPhysicalName (const std::string & physicalName) [virtual]`

Set this destination's physical name.

Returns

const string containing the name

6.22.2.30 `virtual std::string activemq::commands::ActiveMQDestination::toString () const [virtual]`

Returns a string containing the information for this **DataSet** (p. 1133) such as its type and value of its elements.

Returns

formatted string useful for debugging.

Reimplemented from **activemq::commands::BaseDataSet** (p. 531).

Reimplemented in **activemq::commands::ActiveMQTempDestination** (p. 382), **activemq::commands::ActiveMQQueue** (p. 325), **activemq::commands::ActiveMQTopic** (p. 418), **activemq::commands::ActiveMQTempQueue** (p. 391), and **activemq::commands::ActiveMQTempTopic** (p. 400).

6.22.3 Field Documentation

6.22.3.1 **bool activemq::commands::ActiveMQDestination::advisory**
[protected]

6.22.3.2 **const std::string activemq::commands::ActiveMQDestination::ADVISORY_PREFIX** [static, protected]

prefix for Advisory message destinations

6.22.3.3 **const std::string activemq::commands::ActiveMQDestination::COMPOSITE_SEPARATOR** [static, protected]

6.22.3.4 **const std::string activemq::commands::ActiveMQDestination::CONNECTION_ADVISORY_PREFIX** [static, protected]

prefix for connection advisory destinations

6.22.3.5 **const std::string activemq::commands::ActiveMQDestination::CONSUMER_ADVISORY_PREFIX** [static, protected]

prefix for consumer advisory destinations

6.22.3.6 **const std::string activemq::commands::ActiveMQDestination::DEFAULT_ORDERED_TARGET** [static, protected]

The default target for ordered destinations.

6.22.3.7 **bool activemq::commands::ActiveMQDestination::exclusive**
[protected]

6.22.3.8 **const unsigned char activemq::commands::ActiveMQDestination::ID_ACTIVEMQDESTINATION = 0**
[static]

6.22.3.9 **util::ActiveMQProperties** **activemq::commands::ActiveMQDestination::options** [protected]

6.22.3.10 **bool** **activemq::commands::ActiveMQDestination::ordered** [protected]

6.22.3.11 **std::string** **activemq::commands::ActiveMQDestination::orderedTarget** [protected]

6.22.3.12 **std::string** **activemq::commands::ActiveMQDestination::physicalName** [protected]

6.22.3.13 **const std::string** **activemq::commands::ActiveMQDestination::PRODUCER_ADVISORY_PREFIX** [static, protected]

prefix for producer advisory destinations

6.22.3.14 **const std::string** **activemq::commands::ActiveMQDestination::QUEUE_QUALIFIED_PREFIX** [static, protected]

6.22.3.15 **const std::string** **activemq::commands::ActiveMQDestination::TEMP_POSTFIX** [static, protected]

6.22.3.16 **const std::string** **activemq::commands::ActiveMQDestination::TEMP_PREFIX** [static, protected]

6.22.3.17 **const std::string** **activemq::commands::ActiveMQDestination::TEMP_QUEUE_QUALIFIED_PREFIX** [static, protected]

6.22.3.18 **const std::string** **activemq::commands::ActiveMQDestination::TEMP_TOPIC_QUALIFIED_PREFIX** [static, protected]

6.22.3.19 **const std::string** **activemq::commands::ActiveMQDestination::TOPIC_QUALIFIED_PREFIX** [static, protected]

The documentation for this class was generated from the following file:

- `src/main/activemq/commands/ActiveMQDestination.h`

6.23 activemq::wireformat::openwire::marshal::generated::ActiveMQDestinationMarshaller Class Reference

Marshaling code for Open Wire Format for **ActiveMQDestinationMarshaller** (p. 258).

```
#include <src/main/activemq/wireformat/openwire/marshal/generated/ActiveMQDestinationMarshaller.h>
```

Inheritance diagram for activemq::wireformat::openwire::marshal::generated::ActiveMQDestinationMarshaller:

Public Member Functions

- **ActiveMQDestinationMarshaller** ()
- virtual **~ActiveMQDestinationMarshaller** ()
- virtual void **tightUnmarshal** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataInputStream** *dataIn, **utils::BooleanStream** *bs)
Tight Un-marhsal to the given stream.
- virtual int **tightMarshal1** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **utils::BooleanStream** *bs)
Tight Marhsal to the given stream.
- virtual void **tightMarshal2** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataOutputStream** *dataOut, **utils::BooleanStream** *bs)
Tight Marhsal to the given stream.
- virtual void **looseUnmarshal** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataInputStream** *dataIn)
Loose Un-marhsal to the given stream.
- virtual void **looseMarshal** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataOutputStream** *dataOut)
Tight Marhsal to the given stream.

6.23.1 Detailed Description

Marshaling code for Open Wire Format for **ActiveMQDestinationMarshaller** (p. 258).

NOTE!: This file is auto generated - do not modify! if you need to make a change, please see the Java Classes in the activemq-openwire-generator module

6.23.2 Constructor & Destructor Documentation

6.23 activemq::wireformat::openwire::marshal::generated::ActiveMQDestinationMarshaller Class

Reference

259

6.23.2.1 **activemq::wireformat::openwire::marshal::generated::ActiveMQDestinationMarshaller::ActiveMQDestinationMarshaller ()**
[inline]

6.23.2.2 **virtual activemq::wireformat::openwire::marshal::generated::ActiveMQDestinationMarshaller::~~ActiveMQDestinationMarshaller ()**
[inline, virtual]

6.23.3 Member Function Documentation

6.23.3.1 **virtual void activemq::wireformat::openwire::marshal::generated::ActiveMQDestinationMarshaller::looseMarshal (OpenWireFormat * *format*, commands::DataStructure * *command*, decaf::io::DataOutputStream * *ds*)** [virtual]

Tight Marshal to the given stream.

Parameters

<i>format</i>	- The OpenWireFormat properties
<i>command</i>	- the object to Marshal
<i>ds</i>	- DataOutputStream to marshal to

Exceptions

<i>IOException</i>	if an error occurs.
--------------------	---------------------

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1123).

Reimplemented in **activemq::wireformat::openwire::marshal::generated::ActiveMQQueueMarshaller** (p. 331), **activemq::wireformat::openwire::marshal::generated::ActiveMQTempQueueMarshaller** (p. 393), **activemq::wireformat::openwire::marshal::generated::ActiveMQTempTopicMarshaller** (p. 403), **activemq::wireformat::openwire::marshal::generated::ActiveMQTopicMarshaller** (p. 421), and **activemq::wireformat::openwire::marshal::generated::ActiveMQTempDestinationMarshaller** (p. 384).

6.23.3.2 **virtual void activemq::wireformat::openwire::marshal::generated::ActiveMQDestinationMarshaller::looseUnmarshal (OpenWireFormat * *format*, commands::DataStructure * *command*, decaf::io::DataInputStream * *dis*)**
[virtual]

Loose Un-marshal to the given stream.

Parameters

<i>format</i>	- The OpenWireFormat properties
<i>command</i>	- the object to Un-Marshal
<i>dis</i>	- the DataInputStream to Un-Marshal from

Exceptions

<i>IOException</i> if an error occurs.
--

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1125).

Reimplemented in **activemq::wireformat::openwire::marshal::generated::ActiveMQQueueMarshaller** (p. 331), **activemq::wireformat::openwire::marshal::generated::ActiveMQTempQueueMarshaller** (p. 394), **activemq::wireformat::openwire::marshal::generated::ActiveMQTempTopicMarshaller** (p. 403), **activemq::wireformat::openwire::marshal::generated::ActiveMQTopicMarshaller** (p. 421), and **activemq::wireformat::openwire::marshal::generated::ActiveMQTempDestinationMarshaller** (p. 384).

6.23.3.3 **virtual int activemq::wireformat::openwire::marshal::generated::ActiveMQDestinationMarshaller::tightMarshal1 (OpenWireFormat * *format*, commands::DataStructure * *command*, utils::BooleanStream * *bs*)**
[virtual]

Tight Marshal to the given stream.

Parameters

<i>format</i>	The OpenWireFormat properties
<i>command</i>	The object to Marshal
<i>bs</i>	The boolean stream to marshal to.

Exceptions

<i>IOException</i> if an error occurs.
--

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1127).

Reimplemented in **activemq::wireformat::openwire::marshal::generated::ActiveMQQueueMarshaller** (p. 331), **activemq::wireformat::openwire::marshal::generated::ActiveMQTempQueueMarshaller** (p. 394), **activemq::wireformat::openwire::marshal::generated::ActiveMQTempTopicMarshaller** (p. 403), **activemq::wireformat::openwire::marshal::generated::ActiveMQTopicMarshaller** (p. 421), and **activemq::wireformat::openwire::marshal::generated::ActiveMQTempDestinationMarshaller** (p. 385).

6.23.3.4 **virtual void activemq::wireformat::openwire::marshal::generated::ActiveMQDestinationMarshaller::tightMarshal2 (OpenWireFormat * *format*, commands::DataStructure * *command*, decaf::io::DataOutputStream * *ds*, utils::BooleanStream * *bs*)** [virtual]

Tight Marshal to the given stream.

6.23 activemq::wireformat::openwire::marshal::generated::ActiveMQ-DestinationMarshaller Class

Reference

261

Parameters

<i>format</i>	- The OpenWireFormat properties
<i>command</i>	- the object to Marshal
<i>ds</i>	- the DataOutputStream to Marshal to
<i>bs</i>	- boolean stream to marshal to.

Exceptions

<i>IOException</i>	if an error occurs.
--------------------	---------------------

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1129).

Reimplemented in **activemq::wireformat::openwire::marshal::generated::ActiveMQQueueMarshaller** (p. 332), **activemq::wireformat::openwire::marshal::generated::ActiveMQTempQueueMarshaller** (p. 395), **activemq::wireformat::openwire::marshal::generated::ActiveMQTempTopicMarshaller** (p. 404), **activemq::wireformat::openwire::marshal::generated::ActiveMQTopicMarshaller** (p. 422), and **activemq::wireformat::openwire::marshal::generated::ActiveMQTempDestinationMarshaller** (p. 385).

6.23.3.5 virtual void **activemq::wireformat::openwire::marshal::generated::ActiveMQDestinationMarshaller::tightUnmarshal** (**OpenWireFormat** * *format*, **commands::DataStructure** * *command*, **decaf::io::DataInputStream** * *dis*, **utils::BooleanStream** * *bs*) [virtual]

Tight Un-marhsal to the given stream.

Parameters

<i>format</i>	- The OpenWireFormat properties
<i>command</i>	- the object to Un-Marshal
<i>dis</i>	- the DataInputStream to Un-Marshal from
<i>bs</i>	- boolean stream to unmarshal from.

Exceptions

<i>IOException</i>	if an error occurs.
--------------------	---------------------

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1131).

Reimplemented in **activemq::wireformat::openwire::marshal::generated::ActiveMQQueueMarshaller** (p. 332), **activemq::wireformat::openwire::marshal::generated::ActiveMQTempQueueMarshaller** (p. 395), **activemq::wireformat::openwire::marshal::generated::ActiveMQTempTopicMarshaller** (p. 404), **activemq::wireformat::openwire::marshal::generated::ActiveMQTopicMarshaller** (p. 422), and **activemq::wireformat::openwire::marshal::generated::ActiveMQTempDestinationMarshaller** (p. 386).

The documentation for this class was generated from the following file:

- `src/main/activemq/wireformat/openwire/marshal/generated/ActiveMQDestination-Marshaller.h`

6.24 activemq::exceptions::ActiveMQException Class Reference

```
#include <src/main/activemq/exceptions/ActiveMQException.h>
```

Inheritance diagram for `activemq::exceptions::ActiveMQException`:

Public Member Functions

- **ActiveMQException** () throw ()
Default Constructor.
- **ActiveMQException** (const **ActiveMQException** &ex) throw ()
Copy Constructor.
- **ActiveMQException** (const **decaf::lang::Exception** &ex) throw ()
Copy Constructor.
- **ActiveMQException** (const char *file, const int lineNumber, const char *msg,...) throw ()
Constructor - Initializes the file name and line number where this message occurred.
- virtual ~**ActiveMQException** () throw ()
- virtual **ActiveMQException** * **clone** () const
Clones this exception.
- virtual **cms::CMSException** **convertToCMSException** () const
Converts this exception to a new CMSException.

6.24.1 Constructor & Destructor Documentation

6.24.1.1 `activemq::exceptions::ActiveMQException::ActiveMQException ()`
throw ()

Default Constructor.

6.24.1.2 `activemq::exceptions::ActiveMQException::ActiveMQException (const ActiveMQException & ex)` throw ()

Copy Constructor.

Parameters

<i>ex</i>	The Exception whose internal data is copied into this instance.
-----------	---

6.24.1.3 **activemq::exceptions::ActiveMQException::ActiveMQException** (const **decaf::lang::Exception** & *ex*) throw ()

Copy Constructor.

Parameters

<i>ex</i>	The Exception whose internal data is copied into this instance.
-----------	---

6.24.1.4 **activemq::exceptions::ActiveMQException::ActiveMQException** (const char * *file*, const int *lineNumber*, const char * *msg*, ...) throw ()

Constructor - Initializes the file name and line number where this message occurred.

Sets the message to report, using an optional list of arguments to parse into the message.

Parameters

<i>file</i>	The file name where exception occurs.
<i>lineNumber</i>	The line number where the exception occurred.
<i>msg</i>	The message to report.
...	The list of primitives that are formatted into the message.

6.24.1.5 **virtual activemq::exceptions::ActiveMQException::~ActiveMQException** () throw () [virtual]

6.24.2 Member Function Documentation

6.24.2.1 **virtual ActiveMQException* activemq::exceptions::ActiveMQException::clone** () const [virtual]

Clones this exception.

This is useful for cases where you need to preserve the type of the original exception as well as the message. All subclasses should override.

Returns

Copy of this Exception object

Reimplemented from **decaf::lang::Exception** (p. 1282).

Reimplemented in **activemq::exceptions::BrokerException** (p. 564), and **activemq::exceptions::ConnectionFailedException** (p. 959).

```
6.24.2.2 virtual cms::CMSException activemq::exceptions::-
ActiveMQException::convertToCMSException ( ) const
[virtual]
```

Converts this exception to a new CMSException.

Returns

a CMSException with the data from this exception

The documentation for this class was generated from the following file:

- src/main/activemq/exceptions/**ActiveMQException.h**

6.25 activemq::commands::ActiveMQMapMessage Class Reference

```
#include <src/main/activemq/commands/ActiveMQMapMessage.h>
```

Inheritance diagram for activemq::commands::ActiveMQMapMessage:

Public Member Functions

- **ActiveMQMapMessage** ()
- virtual **~ActiveMQMapMessage** () throw ()
- virtual unsigned char **getDataStructureType** () const
*Get the **DataStructure** (p. 1133) Type as defined in CommandTypes.h.*
- virtual bool **isMarshalAware** () const
Indicates that this command is aware of Marshaling, and needs to have its Marshaling methods invoked.
- virtual **ActiveMQMapMessage** * **cloneDataStructure** () const
Clone this object and return a new instance that the caller now owns, this will be an exact copy of this one.
- virtual void **copyDataStructure** (const **DataStructure** *src)
Copy the contents of the passed object into this objects members, overwriting any existing data.
- virtual void **beforeMarshal** (**wireformat::WireFormat** *wireFormat)
Called before marshaling is started to prepare the object to be marshaled.
- virtual std::string **toString** () const
*Returns a string containing the information for this **DataStructure** (p. 1133) such as its type and value of its elements.*
- virtual bool **equals** (const **DataStructure** *value) const
*Compares the **DataStructure** (p. 1133) passed in to this one, and returns if they are equivalent.*

- virtual void **clearBody** () throw (cms::CMSException)
Clears out the body of the message.
- virtual **cms::MapMessage * clone** () const
Clone this message exactly, returns a new instance that the caller is required to delete.
- virtual bool **isEmpty** () const
*Returns true if there are no values stored in the **MapMessage** (p. 1779) body.*
Returns
*true if the body of the **MapMessage** (p. 1779) contains no elements.*

Exceptions

CMSException (p. 826)	<i>if the operation fails due to an internal error.</i>
------------------------------	---

- virtual std::vector< std::string > **getMapNames** () const
*Returns an Enumeration of all the names in the **MapMessage** (p. 1779) object.*
Returns
*STL Vector of String values, each of which is the name of an item in the **Map-Message** (p. 1779)*

Exceptions

CMSException (p. 826)	<i>- if the operation fails due to an internal error.</i>
------------------------------	---

- virtual bool **itemExists** (const std::string &name) const
*Indicates whether an item exists in this **MapMessage** (p. 1779) object.*

Parameters

name	<i>String name of the Object in question</i>
------	--

Returns

boolean value indicating if the name is in the map

Exceptions

CMSException (p. 826)	<i>- if the operation fails due to an internal error.</i>
------------------------------	---

- virtual bool **getBoolean** (const std::string &name) const
Returns the Boolean value of the Specified name.

Parameters

name	<i>Name of the value to fetch from the map</i>
------	--

Exceptions

CMSException (p. 826)	<i>- if the operation fails due to an internal error.</i>
MessageFormatException (p. 1906)	<i>- if this type conversion is invalid.</i>

- virtual void **setBoolean** (const std::string &name, bool value)
Sets a boolean value with the specified name into the Map.

Parameters

name	<i>the name of the boolean</i>
value	<i>the boolean value to set in the Map</i>

Exceptions

CMSEException (p. 826)	- if the operation fails due to an internal error.
MessageNotWritable-Exception	- if the Message (p. 1839) is in Read-only Mode.

- virtual unsigned char **getByte** (const std::string &name) const

Returns the Byte value of the Specified name.

Parameters

name	Name of the value to fetch from the map
------	---

Exceptions

CMSEException (p. 826)	- if the operation fails due to an internal error.
MessageFormatException (p. 1906)	- if this type conversion is invalid.

- virtual void **setByte** (const std::string &name, unsigned char value)

Sets a Byte value with the specified name into the Map.

Parameters

name	the name of the Byte
value	the Byte value to set in the Map

Exceptions

CMSEException (p. 826)	- if the operation fails due to an internal error.
MessageNotWritableException (p. 1923)	- if the Message (p. 1839) is in Read-only Mode.

- virtual std::vector< unsigned char > **getBytes** (const std::string &name) const

Returns the Bytes value of the Specified name.

Parameters

name	Name of the value to fetch from the map
------	---

Exceptions

CMSEException (p. 826)	- if the operation fails due to an internal error.
MessageFormatException (p. 1906)	- if this type conversion is invalid.

- virtual void **setBytes** (const std::string &name, const std::vector< unsigned char > &value)

Sets a Bytes value with the specified name into the Map.

Parameters

name	The name of the Bytes
value	The Bytes value to set in the Map

Exceptions

CMSException (p. 826)	- if the operation fails due to an internal error.
MessageNot-WriteableException (p. 1923)	- if the Message (p. 1839) is in Read-only Mode.

- virtual char **getChar** (const std::string &name) const

Returns the Char value of the Specified name.

Parameters

name	<i>name of the value to fetch from the map</i>
------	--

Exceptions

CMSException (p. 826)	- if the operation fails due to an internal error.
MessageFormatException (p. 1906)	- if this type conversion is invalid.

- virtual void **setChar** (const std::string &name, char value)

Sets a Char value with the specified name into the Map.

Parameters

name	<i>the name of the Char</i>
value	<i>the Char value to set in the Map</i>

Exceptions

CMSException (p. 826)	- if the operation fails due to an internal error.
MessageNot-WriteableException (p. 1923)	- if the Message (p. 1839) is in Read-only Mode.

- virtual double **getDouble** (const std::string &name) const

Returns the Double value of the Specified name.

Parameters

name	<i>Name of the value to fetch from the map</i>
------	--

Exceptions

CMSException (p. 826)	- if the operation fails due to an internal error.
MessageFormatException (p. 1906)	- if this type conversion is invalid.

- virtual void **setDouble** (const std::string &name, double value)

Sets a Double value with the specified name into the Map.

Parameters

name	<i>The name of the Double</i>
value	<i>The Double value to set in the Map</i>

Exceptions

CMSException (p. 826)	- if the operation fails due to an internal error.
MessageNot-WriteableException (p. 1923)	- if the Message (p. 1839) is in Read-only Mode.

- virtual float **getFloat** (const std::string &name) const

Returns the Float value of the Specified name.

Parameters

name	Name of the value to fetch from the map
------	---

Exceptions

CMSException (p. 826)	- if the operation fails due to an internal error.
MessageFormatException (p. 1906)	- if this type conversion is invalid.

- virtual void **setFloat** (const std::string &name, float value)

Sets a Float value with the specified name into the Map.

Parameters

name	The name of the Float
value	The Float value to set in the Map

Exceptions

CMSException (p. 826)	- if the operation fails due to an internal error.
MessageNot-WriteableException (p. 1923)	- if the Message (p. 1839) is in Read-only Mode.

- virtual int **getInt** (const std::string &name) const

Returns the Int value of the Specified name.

Parameters

name	Name of the value to fetch from the map
------	---

Exceptions

CMSException (p. 826)	- if the operation fails due to an internal error.
MessageFormatException (p. 1906)	- if this type conversion is invalid.

- virtual void **setInt** (const std::string &name, int value)

Sets a Int value with the specified name into the Map.

Parameters

name	The name of the Int
value	The Int value to set in the Map

Exceptions

CMSException (p. 826)	- if the operation fails due to an internal error.
MessageNot-WriteableException (p. 1923)	- if the Message (p. 1839) is in Read-only Mode.

- virtual long long **getLong** (const std::string &name) const

Returns the Long value of the Specified name.

Parameters

name	Name of the value to fetch from the map
------	---

Exceptions

CMSException (p. 826)	- if the operation fails due to an internal error.
MessageFormatException (p. 1906)	- if this type conversion is invalid.

- virtual void **setLong** (const std::string &name, long long value)

Sets a Long value with the specified name into the Map.

Parameters

name	The name of the Long
value	The Long value to set in the Map

Exceptions

CMSException (p. 826)	- if the operation fails due to an internal error.
MessageNot-WriteableException (p. 1923)	- if the Message (p. 1839) is in Read-only Mode.

- virtual short **getShort** (const std::string &name) const

Returns the Short value of the Specified name.

Parameters

name	Name of the value to fetch from the map
------	---

Exceptions

CMSException (p. 826)	- if the operation fails due to an internal error.
MessageFormatException (p. 1906)	- if this type conversion is invalid.

- virtual void **setShort** (const std::string &name, short value)

Sets a Short value with the specified name into the Map.

Parameters

name	The name of the Short
value	The Short value to set in the Map

Exceptions

CMSException (p. 826)	- if the operation fails due to an internal error.
MessageNot-WriteableException (p. 1923)	- if the Message (p. 1839) is in Read-only Mode.

- virtual std::string **getString** (const std::string &name) const

Returns the String value of the Specified name.

Parameters

name	Name of the value to fetch from the map
------	---

Exceptions

CMSException (p. 826)	- if the operation fails due to an internal error.
MessageFormatException (p. 1906)	- if this type conversion is invalid.

- virtual void **setString** (const std::string &name, const std::string &value)

Sets a String value with the specified name into the Map.

Parameters

name	The name of the String
value	The String value to set in the Map

Exceptions

CMSException (p. 826)	- if the operation fails due to an internal error.
MessageNot-WriteableException (p. 1923)	- if the Message (p. 1839) is in Read-only Mode.

Static Public Attributes

- static const unsigned char **ID_ACTIVEMQMAPMESSAGE** = 25

Protected Member Functions

- **util::PrimitiveMap & getMap** ()

Fetches a reference to this objects PrimitiveMap, if one needs to be created or unmarshaled, this will perform the correct steps.

- const **util::PrimitiveMap & getMap** () const
- virtual void **checkMapIsUnmarshalled** () const

Performs the unmarshal on the Map if needed, otherwise just returns.

6.25.1 Constructor & Destructor Documentation

6.25.1.1 activemq::commands::ActiveMQMapMessage::ActiveMQMapMessage ()

6.25.1.2 virtual **activemq::commands::ActiveMQMapMessage::~ActiveMQMapMessage**() **throw**() [virtual]

6.25.2 Member Function Documentation

6.25.2.1 virtual void **activemq::commands::ActiveMQMapMessage::beforeMarshal**(**wireformat::WireFormat** * *wireFormat*) [virtual]

Called before marshaling is started to prepare the object to be marshaled.

Parameters

<i>wireFormat</i>	The wireformat object to control marshaling
-------------------	---

Exceptions

<i>IOException</i>	if an I/O error occurs.
--------------------	-------------------------

Implements **activemq::wireformat::MarshalAware** (p. 1794).

6.25.2.2 virtual void **activemq::commands::ActiveMQMapMessage::checkMapsUnmarshalled**() **const** [protected, virtual]

Performs the unmarshal on the Map if needed, otherwise just returns.

Exceptions

<i>NullPointerException</i>	if the internal Map is Null.
-----------------------------	------------------------------

6.25.2.3 virtual void **activemq::commands::ActiveMQMapMessage::clearBody**() **throw**(**cms::CMSEException**) [virtual]

Clears out the body of the message.

This does not clear the headers or properties.

Exceptions

<i>CMSEException</i>	- if an internal error occurs.
----------------------	--------------------------------

Reimplemented from **activemq::commands::ActiveMQMessageTemplate**< **cms::MapMessage** > (p. 296).

6.25.2.4 `virtual cms::MapMessage* activemq::commands::-`
`ActiveMQMapMessage::clone () const [inline,`
`virtual]`

Clone this message exactly, returns a new instance that the caller is required to delete.

Returns

new copy of this message

Implements **cms::Message** (p. 1844).

6.25.2.5 `virtual ActiveMQMapMessage* activemq::commands::-`
`ActiveMQMapMessage::cloneDataStructure () const`
`[virtual]`

Clone this object and return a new instance that the caller now owns, this will be an exact copy of this one.

Returns

new copy of this object.

Reimplemented from **activemq::commands::Message** (p. 1826).

6.25.2.6 `virtual void activemq::commands::ActiveMQMapMessage-`
`::copyDataStructure (const DataStructure * src)`
`[virtual]`

Copy the contents of the passed object into this objects members, overwriting any existing data.

Returns

src - Source Object

Reimplemented from **activemq::commands::Message** (p. 1827).

6.25.2.7 `virtual bool activemq::commands::ActiveMQMapMessage::equals (const`
`DataStructure * value) const [virtual]`

Compares the **DataStructure** (p. 1133) passed in to this one, and returns if they are equivalent.

Equivalent here means that they are of the same type, and that each element of the objects are the same.

Returns

true if **DataStructure** (p. 1133)'s are Equal.

Reimplemented from **activemq::commands::ActiveMQMessageTemplate**< **cms::MapMessage** > (p. 297).

6.25.2.8 virtual bool **activemq::commands::ActiveMQMapMessage::getBoolean** (const std::string & *name*) const [virtual]

Returns the Boolean value of the Specified name.

Parameters

<i>name</i>	Name of the value to fetch from the map
-------------	---

Exceptions

CMSException (p. 826)	- if the operation fails due to an internal error.
MessageFormatException (p. 1906)	- if this type conversion is invalid.

Implements **cms::MapMessage** (p. 1782).

6.25.2.9 virtual unsigned char **activemq::commands::ActiveMQMapMessage::getBytes** (const std::string & *name*) const [virtual]

Returns the Byte value of the Specified name.

Parameters

<i>name</i>	Name of the value to fetch from the map
-------------	---

Exceptions

CMSException (p. 826)	- if the operation fails due to an internal error.
MessageFormatException (p. 1906)	- if this type conversion is invalid.

Implements **cms::MapMessage** (p. 1782).

6.25.2.10 `virtual std::vector<unsigned char> activemq::commands::ActiveMQMapMessage::getBytes (const std::string & name) const`
`[virtual]`

Returns the Bytes value of the Specified name.

Parameters

<i>name</i>	Name of the value to fetch from the map
-------------	---

Exceptions

<i>CMSException</i> (p. 826)	- if the operation fails due to an internal error.
<i>MessageFormatException</i> (p. 1906)	- if this type conversion is invalid.

Implements **cms::MapMessage** (p. 1783).

6.25.2.11 `virtual char activemq::commands::ActiveMQMapMessage::getChar (const std::string & name) const` `[virtual]`

Returns the Char value of the Specified name.

Parameters

<i>name</i>	name of the value to fetch from the map
-------------	---

Exceptions

<i>CMSException</i> (p. 826)	- if the operation fails due to an internal error.
<i>MessageFormatException</i> (p. 1906)	- if this type conversion is invalid.

Implements **cms::MapMessage** (p. 1783).

6.25.2.12 `virtual unsigned char activemq::commands::ActiveMQMapMessage::getDataStructureType () const`
`[virtual]`

Get the **DataStructure** (p. 1133) Type as defined in CommandTypes.h.

Returns

The type of the data structure

Reimplemented from **activemq::commands::Message** (p. 1829).

6.25.2.13 virtual double **activemq::commands::ActiveMQMapMessage::getDouble** (
const std::string & *name*) const [virtual]

Returns the Double value of the Specified name.

Parameters

<i>name</i>	Name of the value to fetch from the map
-------------	---

Exceptions

CMSException (p. 826)	- if the operation fails due to an internal error.
MessageFormatException (p. 1906)	- if this type conversion is invalid.

Implements **cms::MapMessage** (p. 1783).

6.25.2.14 virtual float **activemq::commands::ActiveMQMapMessage::getFloat** (
const std::string & *name*) const [virtual]

Returns the Float value of the Specified name.

Parameters

<i>name</i>	Name of the value to fetch from the map
-------------	---

Exceptions

CMSException (p. 826)	- if the operation fails due to an internal error.
MessageFormatException (p. 1906)	- if this type conversion is invalid.

Implements **cms::MapMessage** (p. 1784).

6.25.2.15 `virtual int activemq::commands::ActiveMQMapMessage::getInt (const std::string & name) const` [virtual]

Returns the Int value of the Specified name.

Parameters

<i>name</i>	Name of the value to fetch from the map
-------------	---

Exceptions

CMSException (p. 826)	- if the operation fails due to an internal error.
MessageFormatException (p. 1906)	- if this type conversion is invalid.

Implements `cms::MapMessage` (p. 1784).

6.25.2.16 `virtual long long activemq::commands::ActiveMQMapMessage::getLong (const std::string & name) const` [virtual]

Returns the Long value of the Specified name.

Parameters

<i>name</i>	Name of the value to fetch from the map
-------------	---

Exceptions

CMSException (p. 826)	- if the operation fails due to an internal error.
MessageFormatException (p. 1906)	- if this type conversion is invalid.

Implements `cms::MapMessage` (p. 1785).

6.25.2.17 `util::PrimitiveMap& activemq::commands::ActiveMQMapMessage::getMap ()` [protected]

Fetches a reference to this objects PrimitiveMap, if one needs to be created or unmarshaled, this will perform the correct steps.

Returns

reference to a PrimitiveMap;

Exceptions

<i>NullPointerException</i>	if the internal Map is Null.
-----------------------------	------------------------------

6.25.2.18 `const util::PrimitiveMap& activemq::commands::ActiveMQMapMessage::getMap () const` [protected]

6.25.2.19 `virtual std::vector< std::string > activemq::commands::ActiveMQMapMessage::getMapNames () const`
[virtual]

Returns an Enumeration of all the names in the **MapMessage** (p. 1779) object.

Returns

STL Vector of String values, each of which is the name of an item in the **MapMessage** (p. 1779)

Exceptions

CMSException (p. 826)	- if the operation fails due to an internal error.
---------------------------------	--

Implements **cms::MapMessage** (p. 1785).

6.25.2.20 `virtual short activemq::commands::ActiveMQMapMessage::getShort (const std::string & name) const` [virtual]

Returns the Short value of the Specified name.

Parameters

<i>name</i>	Name of the value to fetch from the map
-------------	---

Exceptions

CMSException (p. 826)	- if the operation fails due to an internal error.
MessageFormatException (p. 1906)	- if this type conversion is invalid.

Implements **cms::MapMessage** (p. 1785).

6.25.2.21 `virtual std::string activemq::commands::ActiveMQMapMessage::getString
(const std::string & name) const [virtual]`

Returns the String value of the Specified name.

Parameters

<i>name</i>	Name of the value to fetch from the map
-------------	---

Exceptions

<i>CMSException</i> (p. 826)	- if the operation fails due to an internal error.
<i>MessageFormatException</i> (p. 1906)	- if this type conversion is invalid.

Implements **cms::MapMessage** (p. 1786).

6.25.2.22 `virtual bool activemq::commands::ActiveMQMapMessage::isEmpty ()
const [virtual]`

Returns true if there are no values stored in the **MapMessage** (p. 1779) body.

Returns

true if the body of the **MapMessage** (p. 1779) contains no elements.

Exceptions

<i>CMSException</i> (p. 826)	if the operation fails due to an internal error.
--	--

Implements **cms::MapMessage** (p. 1786).

6.25.2.23 `virtual bool activemq::commands::ActiveMQMap-
Message::isMarshalAware () const [inline,
virtual]`

Indicates that this command is aware of Marshaling, and needs to have its Marshaling methods invoked.

Returns

boolean indicating desire to be in marshaling stages

Reimplemented from **activemq::commands::Message** (p. 1832).

6.25.2.24 virtual bool **activemq::commands::ActiveMQMapMessage::itemExists** (
const std::string & *name*) const [virtual]

Indicates whether an item exists in this **MapMessage** (p. 1779) object.

Parameters

<i>name</i>	String name of the Object in question
-------------	---------------------------------------

Returns

boolean value indicating if the name is in the map

Exceptions

CMSException (p. 826)	- if the operation fails due to an internal error.
---------------------------------	--

Implements **cms::MapMessage** (p. 1787).

6.25.2.25 virtual void **activemq::commands::ActiveMQMapMessage::setBoolean** (
const std::string & *name*, bool *value*) [virtual]

Sets a boolean value with the specified name into the Map.

Parameters

<i>name</i>	the name of the boolean
<i>value</i>	the boolean value to set in the Map

Exceptions

CMSException (p. 826)	- if the operation fails due to an internal error.
MessageNotWritableException	- if the Message (p. 1839) is in Read-only Mode.

Implements **cms::MapMessage** (p. 1787).

6.25.2.26 virtual void **activemq::commands::ActiveMQMapMessage::setByte** (const
std::string & *name*, unsigned char *value*) [virtual]

Sets a Byte value with the specified name into the Map.

Parameters

<i>name</i>	the name of the Byte
<i>value</i>	the Byte value to set in the Map

Exceptions

<i>CMSException</i> (p. 826)	- if the operation fails due to an internal error.
<i>MessageNot-WriteableException</i> (p. 1923)	- if the Message (p. 1839) is in Read-only Mode.

Implements **cms::MapMessage** (p. 1787).

6.25.2.27 `virtual void activemq::commands::ActiveMQMapMessage::setBytes
(const std::string & name, const std::vector< unsigned char > & value)
[virtual]`

Sets a Bytes value with the specified name into the Map.

Parameters

<i>name</i>	The name of the Bytes
<i>value</i>	The Bytes value to set in the Map

Exceptions

<i>CMSException</i> (p. 826)	- if the operation fails due to an internal error.
<i>MessageNot-WriteableException</i> (p. 1923)	- if the Message (p. 1839) is in Read-only Mode.

Implements **cms::MapMessage** (p. 1788).

6.25.2.28 `virtual void activemq::commands::ActiveMQMapMessage::setChar (const
std::string & name, char value) [virtual]`

Sets a Char value with the specified name into the Map.

Parameters

<i>name</i>	the name of the Char
<i>value</i>	the Char value to set in the Map

Exceptions

<i>CMSException</i> (p. 826)	- if the operation fails due to an internal error.
<i>MessageNot-WriteableException</i> (p. 1923)	- if the Message (p. 1839) is in Read-only Mode.

Implements **cms::MapMessage** (p. 1788).

6.25.2.29 virtual void **activemq::commands::ActiveMQMapMessage::setDouble** (
const std::string & *name*, double *value*) [virtual]

Sets a Double value with the specified name into the Map.

Parameters

<i>name</i>	The name of the Double
<i>value</i>	The Double value to set in the Map

Exceptions

CMSException (p. 826)	- if the operation fails due to an internal error.
MessageNotWriteableException (p. 1923)	- if the Message (p. 1839) is in Read-only Mode.

Implements **cms::MapMessage** (p. 1789).

6.25.2.30 virtual void **activemq::commands::ActiveMQMapMessage::setFloat** (
const std::string & *name*, float *value*) [virtual]

Sets a Float value with the specified name into the Map.

Parameters

<i>name</i>	The name of the Float
<i>value</i>	The Float value to set in the Map

Exceptions

CMSException (p. 826)	- if the operation fails due to an internal error.
MessageNotWriteableException (p. 1923)	- if the Message (p. 1839) is in Read-only Mode.

Implements **cms::MapMessage** (p. 1789).

6.25.2.31 virtual void **activemq::commands::ActiveMQMapMessage::setInt** (const
std::string & *name*, int *value*) [virtual]

Sets a Int value with the specified name into the Map.

Parameters

<i>name</i>	The name of the Int
<i>value</i>	The Int value to set in the Map

Exceptions

CMSException (p. 826)	- if the operation fails due to an internal error.
MessageNot-WriteableException (p. 1923)	- if the Message (p. 1839) is in Read-only Mode.

Implements **cms::MapMessage** (p. 1789).

6.25.2.32 `virtual void activemq::commands::ActiveMQMapMessage::setLong (const std::string & name, long long value) [virtual]`

Sets a Long value with the specified name into the Map.

Parameters

<i>name</i>	The name of the Long
<i>value</i>	The Long value to set in the Map

Exceptions

CMSException (p. 826)	- if the operation fails due to an internal error.
MessageNot-WriteableException (p. 1923)	- if the Message (p. 1839) is in Read-only Mode.

Implements **cms::MapMessage** (p. 1790).

6.25.2.33 `virtual void activemq::commands::ActiveMQMapMessage::setShort (const std::string & name, short value) [virtual]`

Sets a Short value with the specified name into the Map.

Parameters

<i>name</i>	The name of the Short
<i>value</i>	The Short value to set in the Map

Exceptions

CMSException (p. 826)	- if the operation fails due to an internal error.
MessageNotWriteableException (p. 1923)	- if the Message (p. 1839) is in Read-only Mode.

Implements **cms::MapMessage** (p. 1790).

6.25.2.34 virtual void **activemq::commands::ActiveMQMapMessage::setString** (
const std::string & *name*, const std::string & *value*) [virtual]

Sets a String value with the specified name into the Map.

Parameters

<i>name</i>	The name of the String
<i>value</i>	The String value to set in the Map

Exceptions

CMSException (p. 826)	- if the operation fails due to an internal error.
MessageNotWriteableException (p. 1923)	- if the Message (p. 1839) is in Read-only Mode.

Implements **cms::MapMessage** (p. 1791).

6.25.2.35 virtual std::string **activemq::commands::ActiveMQMapMessage::toString** (
) const [virtual]

Returns a string containing the information for this **DataStructure** (p. 1133) such as its type and value of its elements.

Returns

formatted string useful for debugging.

Reimplemented from **activemq::commands::Message** (p. 1836).

6.25.3 Field Documentation

```
6.25.3.1  const unsigned char activemq::commands::ActiveM-
          QMapMessage::ID_ACTIVEMQMAPMESSAGE = 25
          [static]
```

The documentation for this class was generated from the following file:

- src/main/activemq/commands/**ActiveMQMapMessage.h**

6.26 activemq::wireformat::openwire::marshal::generated::ActiveMQMapMessageMarshaller Class Reference

Marshaling code for Open Wire Format for **ActiveMQMapMessageMarshaller** (p. 284).

```
#include <src/main/activemq/wireformat/openwire/marshal/generated/-
ActiveMQMapMessageMarshaller.h>
```

Inheritance diagram for activemq::wireformat::openwire::marshal::generated::ActiveMQMapMessageMarshaller:

Public Member Functions

- **ActiveMQMapMessageMarshaller ()**
- virtual **~ActiveMQMapMessageMarshaller ()**
- virtual **commands::DataStructure * createObject () const**
Creates a new instance of the class that this class is a marshaling director for.
- virtual unsigned char **getDataStructureType () const**
Gets the DataStructureType that this class marshals/unmarshals.
- virtual void **tightUnmarshal (OpenWireFormat *wireFormat, commands::DataStructure *dataStructure, decaf::io::DataInputStream *dataIn, utils::BooleanStream *bs)**
Tight Un-marhsal to the given stream.
- virtual int **tightMarshal1 (OpenWireFormat *wireFormat, commands::DataStructure *dataStructure, utils::BooleanStream *bs)**
Tight Marhsal to the given stream.
- virtual void **tightMarshal2 (OpenWireFormat *wireFormat, commands::DataStructure *dataStructure, decaf::io::DataOutputStream *dataOut, utils::BooleanStream *bs)**
Tight Marhsal to the given stream.
- virtual void **looseUnmarshal (OpenWireFormat *wireFormat, commands::DataStructure *dataStructure, decaf::io::DataInputStream *dataIn)**
Loose Un-marhsal to the given stream.
- virtual void **looseMarshal (OpenWireFormat *wireFormat, commands::DataStructure *dataStructure, decaf::io::DataOutputStream *dataOut)**
Tight Marhsal to the given stream.

6.26 activemq::wireformat::openwire::marshal::generated::ActiveMQMapMessageMarshaller Class

Reference

285

6.26.1 Detailed Description

Marshaling code for Open Wire Format for **ActiveMQMapMessageMarshaller** (p. 284).

NOTE!: This file is auto generated - do not modify! if you need to make a change, please see the Java Classes in the activemq-openwire-generator module

6.26.2 Constructor & Destructor Documentation

6.26.2.1 **activemq::wireformat::openwire::marshal::generated::ActiveMQMapMessageMarshaller::ActiveMQMapMessageMarshaller ()**
[inline]

6.26.2.2 **virtual activemq::wireformat::openwire::marshal::generated::ActiveMQMapMessageMarshaller::~~ActiveMQMapMessageMarshaller ()**
[inline, virtual]

6.26.3 Member Function Documentation

6.26.3.1 **virtual commands::DataStructure* activemq::wireformat::openwire::marshal::generated::ActiveMQMapMessageMarshaller::createObject () const** [virtual]

Creates a new instance of the class that this class is a marshaling director for.

Returns

newly allocated Command

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1120).

6.26.3.2 **virtual unsigned char activemq::wireformat::openwire::marshal::generated::ActiveMQMapMessageMarshaller::getDataStructureType () const**
[virtual]

Gets the DataStructureType that this class marshals/unmarshals.

Returns

byte Id of this classes DataStructureType

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1122).

6.26.3.3 `virtual void activemq::wireformat::openwire::marshal::generated::ActiveMQMapMessageMarshaller::looseMarshal (OpenWireFormat * format, commands::DataStructure * command, decaf::io::DataOutputStream * ds) [virtual]`

Tight Marhsal to the given stream.

Parameters

<i>format</i>	- The OpenwireFormat properties
<i>command</i>	- the object to Marshal
<i>ds</i>	- DataOutputStream to marshal to

Exceptions

<i>IOException</i>	if an error occurs.
--------------------	---------------------

Reimplemented from `activemq::wireformat::openwire::marshal::generated::MessageMarshaller` (p. 1919).

6.26.3.4 `virtual void activemq::wireformat::openwire::marshal::generated::ActiveMQMapMessageMarshaller::looseUnmarshal (OpenWireFormat * format, commands::DataStructure * command, decaf::io::DataInputStream * dis) [virtual]`

Loose Un-marhsal to the given stream.

Parameters

<i>format</i>	- The OpenwireFormat properties
<i>command</i>	- the object to Un-Marshal
<i>dis</i>	- the DataInputStream to Un-Marshal from

Exceptions

<i>IOException</i>	if an error occurs.
--------------------	---------------------

Reimplemented from `activemq::wireformat::openwire::marshal::generated::MessageMarshaller` (p. 1919).

6.26.3.5 `virtual int activemq::wireformat::openwire::marshal::generated::ActiveMQMapMessageMarshaller::tightMarshal1 (OpenWireFormat * format, commands::DataStructure * command, utils::BooleanStream * bs) [virtual]`

Tight Marhsal to the given stream.

6.26 activemq::wireformat::openwire::marshal::generated::ActiveMQMapMessageMarshaller Class

Reference

287

Parameters

<i>format</i>	The OpenwireFormat properties
<i>command</i>	The object to Marshal
<i>bs</i>	The boolean stream to marshal to.

Exceptions

<i>IOException</i>	if an error occurs.
--------------------	---------------------

Reimplemented from **activemq::wireformat::openwire::marshal::generated::MessageMarshaller** (p. 1920).

6.26.3.6 virtual void **activemq::wireformat::openwire::marshal::generated::ActiveMQMapMessageMarshaller::tightMarshal2** (**OpenWireFormat** * *format*, **commands::DataStructure** * *command*, **decaf::io::DataOutputStream** * *ds*, **utils::BooleanStream** * *bs*) [virtual]

Tight Marhsal to the given stream.

Parameters

<i>format</i>	- The OpenwireFormat properties
<i>command</i>	- the object to Marshal
<i>ds</i>	- the DataOutputStream to Marshal to
<i>bs</i>	- boolean stream to marshal to.

Exceptions

<i>IOException</i>	if an error occurs.
--------------------	---------------------

Reimplemented from **activemq::wireformat::openwire::marshal::generated::MessageMarshaller** (p. 1921).

6.26.3.7 virtual void **activemq::wireformat::openwire::marshal::generated::ActiveMQMapMessageMarshaller::tightUnmarshal** (**OpenWireFormat** * *format*, **commands::DataStructure** * *command*, **decaf::io::DataInputStream** * *dis*, **utils::BooleanStream** * *bs*) [virtual]

Tight Un-marhsal to the given stream.

Parameters

<i>format</i>	- The OpenwireFormat properties
<i>command</i>	- the object to Un-Marshal
<i>dis</i>	- the DataInputStream to Un-Marshal from
<i>bs</i>	- boolean stream to unmarshal from.

Exceptions

<i>IOException</i> if an error occurs.
--

Reimplemented from **activemq::wireformat::openwire::marshal::generated::MessageMarshaller** (p. 1921).

The documentation for this class was generated from the following file:

- src/main/activemq/wireformat/openwire/marshal/generated/**ActiveMQMap-MessageMarshaller.h**

6.27 activemq::commands::ActiveMQMessage Class Reference

```
#include <src/main/activemq/commands/ActiveMQMessage.h>
```

Inheritance diagram for **activemq::commands::ActiveMQMessage**:

Public Member Functions

- **ActiveMQMessage** ()
- virtual **~ActiveMQMessage** () throw ()
- virtual unsigned char **getDataStructureType** () const
*Get the **DataStructure** (p. 1133) Type as defined in CommandTypes.h.*
- virtual void **copyDataStructure** (const **DataStructure** *src)
Copy the contents of the passed object into this objects members, overwriting any existing data.
- virtual **ActiveMQMessage** * **cloneDataStructure** () const
Clone this obbject and return a new instance that the caller now owns, this will be an exact copy of this one.
- virtual std::string **toString** () const
*Returns a string containing the information for this **DataStructure** (p. 1133) such as its type and value of its elements.*
- virtual bool **equals** (const **DataStructure** *value) const
*Compares the **DataStructure** (p. 1133) passed in to this one, and returns if they are equivalent.*
- virtual **cms::Message** * **clone** () const
Clone this message exactly, returns a new instance that the caller is required to delete.

Static Public Attributes

- static const unsigned char **ID_ACTIVEMQMESSAGE** = 23

6.27.1 Constructor & Destructor Documentation

6.27.1.1 `activemq::commands::ActiveMQMessage::ActiveMQMessage ()`

6.27.1.2 `virtual activemq::commands::ActiveMQMessage::~~ActiveMQMessage () throw () [inline, virtual]`

6.27.2 Member Function Documentation

6.27.2.1 `virtual cms::Message* activemq::commands::ActiveMQMessage::clone () const [inline, virtual]`

Clone this message exactly, returns a new instance that the caller is required to delete.

Returns

new copy of this message

Implements **cms::Message** (p. 1844).

6.27.2.2 `virtual ActiveMQMessage* activemq::commands::ActiveMQMessage::cloneDataStructure () const [virtual]`

Clone this obbject and return a new instance that the caller now owns, this will be an exact copy of this one.

Returns

new copy of this object.

Reimplemented from **activemq::commands::Message** (p. 1826).

6.27.2.3 `virtual void activemq::commands::ActiveMQMessage::copyDataStructure (const DataStructure * src) [virtual]`

Copy the contents of the passed object into this objects members, overwriting any existing data.

Returns

src - Source Object

Reimplemented from **activemq::commands::Message** (p. 1827).

6.27.2.4 `virtual bool activemq::commands::ActiveMQMessage::equals (const
DataStructure * value) const [virtual]`

Compares the **DataStructure** (p. 1133) passed in to this one, and returns if they are equivalent.

Equivalent here means that they are of the same type, and that each element of the objects are the same.

Returns

true if **DataStructure** (p. 1133)'s are Equal.

Reimplemented from **activemq::commands::ActiveMQMessageTemplate**< **cms::Message** > (p. 297).

6.27.2.5 `virtual unsigned char activemq::commands::ActiveMQMessage::getData-
StructureType () const [virtual]`

Get the **DataStructure** (p. 1133) Type as defined in CommandTypes.h.

Returns

The type of the data structure

Reimplemented from **activemq::commands::Message** (p. 1829).

6.27.2.6 `virtual std::string activemq::commands::ActiveMQMessage::toString ()
const [virtual]`

Returns a string containing the information for this **DataStructure** (p. 1133) such as its type and value of its elements.

Returns

formatted string useful for debugging.

Reimplemented from **activemq::commands::Message** (p. 1836).

6.27.3 Field Documentation

6.27.3.1 `const unsigned char activemq::commands::ActiveMQMessage::ID_ACTIVE-
MQMESSAGE = 23 [static]`

The documentation for this class was generated from the following file:

- src/main/activemq/commands/**ActiveMQMessage.h**

6.28 activemq::wireformat::openwire::marshal::generated::ActiveMQMessageMarshaller Class

Reference

291

6.28 activemq::wireformat::openwire::marshal::generated::ActiveMQMessageMarshaller Class Reference

Marshaling code for Open Wire Format for **ActiveMQMessageMarshaller** (p. 291).

```
#include <src/main/activemq/wireformat/openwire/marshal/generated/-  
ActiveMQMessageMarshaller.h>
```

Inheritance diagram for activemq::wireformat::openwire::marshal::generated::ActiveMQMessageMarshaller:

Public Member Functions

- **ActiveMQMessageMarshaller** ()
- virtual **~ActiveMQMessageMarshaller** ()
- virtual **commands::DataStructure * createObject** () const
Creates a new instance of the class that this class is a marshaling director for.
- virtual unsigned char **getDataStructureType** () const
Gets the DataStructureType that this class marshals/unmarshals.
- virtual void **tightUnmarshal** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataInputStream** *dataIn, **utils::BooleanStream** *bs)
Tight Un-marhsal to the given stream.
- virtual int **tightMarshal1** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **utils::BooleanStream** *bs)
Tight Marshal to the given stream.
- virtual void **tightMarshal2** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataOutputStream** *dataOut, **utils::BooleanStream** *bs)
Tight Marshal to the given stream.
- virtual void **looseUnmarshal** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataInputStream** *dataIn)
Loose Un-marhsal to the given stream.
- virtual void **looseMarshal** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataOutputStream** *dataOut)
Tight Marshal to the given stream.

6.28.1 Detailed Description

Marshaling code for Open Wire Format for **ActiveMQMessageMarshaller** (p. 291).

NOTE!: This file is auto generated - do not modify! if you need to make a change, please see the Java Classes in the activemq-openwire-generator module

6.28.2 Constructor & Destructor Documentation

6.28.2.1 **activemq::wireformat::openwire::marshal::generated::ActiveMQMessageMarshaller::ActiveMQMessageMarshaller ()**
[inline]

6.28.2.2 **virtual activemq::wireformat::openwire::marshal::generated::ActiveMQMessageMarshaller::~~ActiveMQMessageMarshaller ()** [inline, virtual]

6.28.3 Member Function Documentation

6.28.3.1 **virtual commands::DataStructure* activemq::wireformat::openwire::marshal::generated::ActiveMQMessageMarshaller::createObject ()**
const [virtual]

Creates a new instance of the class that this class is a marshaling director for.

Returns

newly allocated Command

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1120).

6.28.3.2 **virtual unsigned char activemq::wireformat::openwire::marshal::generated::ActiveMQMessageMarshaller::getDataStructureType ()** const
[virtual]

Gets the DataStructureType that this class marshals/unmarshals.

Returns

byte Id of this classes DataStructureType

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1122).

6.28.3.3 **virtual void activemq::wireformat::openwire::marshal::generated::ActiveMQMessageMarshaller::looseMarshal (OpenWireFormat * format, commands::DataStructure * command, decaf::io::DataOutputStream * ds)** [virtual]

Tight Marshal to the given stream.

Parameters

<i>format</i>	- The OpenWireFormat properties
<i>command</i>	- the object to Marshal
<i>ds</i>	- DataOutputStream to marshal to

6.28 activemq::wireformat::openwire::marshal::generated::ActiveMQMessage-Marshaller Class

Reference

293

Exceptions

<i>IOException</i>	if an error occurs.
--------------------	---------------------

Reimplemented from **activemq::wireformat::openwire::marshal::generated::MessageMarshaller** (p. 1919).

6.28.3.4 virtual void **activemq::wireformat::openwire::marshal::generated::ActiveMQMessageMarshaller::looseUnmarshal** (**OpenWireFormat** * *format*, **commands::DataStructure** * *command*, **decaf::io::DataInputStream** * *dis*)
[virtual]

Loose Un-marhsal to the given stream.

Parameters

<i>format</i>	- The OpenWireFormat properties
<i>command</i>	- the object to Un-Marshal
<i>dis</i>	- the DataInputStream to Un-Marshal from

Exceptions

<i>IOException</i>	if an error occurs.
--------------------	---------------------

Reimplemented from **activemq::wireformat::openwire::marshal::generated::MessageMarshaller** (p. 1919).

6.28.3.5 virtual int **activemq::wireformat::openwire::marshal::generated::ActiveMQMessageMarshaller::tightMarshal1** (**OpenWireFormat** * *format*, **commands::DataStructure** * *command*, **utils::BooleanStream** * *bs*)
[virtual]

Tight Marhsal to the given stream.

Parameters

<i>format</i>	The OpenWireFormat properties
<i>command</i>	The object to Marshal
<i>bs</i>	The boolean stream to marshal to.

Exceptions

<i>IOException</i>	if an error occurs.
--------------------	---------------------

Reimplemented from **activemq::wireformat::openwire::marshal::generated::MessageMarshaller** (p. 1920).

6.28.3.6 virtual void `activemq::wireformat::openwire::marshal::generated::ActiveMQMessageMarshaller::tightMarshal2` (`OpenWireFormat * format`, `commands::DataStructure * command`, `decaf::io::DataOutputStream * ds`, `utils::BooleanStream * bs`) `[virtual]`

Tight Marshal to the given stream.

Parameters

<i>format</i>	- The OpenWireFormat properties
<i>command</i>	- the object to Marshal
<i>ds</i>	- the DataOutputStream to Marshal to
<i>bs</i>	- boolean stream to marshal to.

Exceptions

<i>IOException</i>	if an error occurs.
--------------------	---------------------

Reimplemented from `activemq::wireformat::openwire::marshal::generated::MessageMarshaller` (p. 1921).

6.28.3.7 virtual void `activemq::wireformat::openwire::marshal::generated::ActiveMQMessageMarshaller::tightUnmarshal` (`OpenWireFormat * format`, `commands::DataStructure * command`, `decaf::io::DataInputStream * dis`, `utils::BooleanStream * bs`) `[virtual]`

Tight Un-marshal to the given stream.

Parameters

<i>format</i>	- The OpenWireFormat properties
<i>command</i>	- the object to Un-Marshal
<i>dis</i>	- the DataInputStream to Un-Marshal from
<i>bs</i>	- boolean stream to unmarshal from.

Exceptions

<i>IOException</i>	if an error occurs.
--------------------	---------------------

Reimplemented from `activemq::wireformat::openwire::marshal::generated::MessageMarshaller` (p. 1921).

The documentation for this class was generated from the following file:

- `src/main/activemq/wireformat/openwire/marshal/generated/ActiveMQMessageMarshaller.h`

6.29 activemq::commands::ActiveMQMessageTemplate< T > - Class Template Reference

```
#include <src/main/activemq/commands/ActiveMQMessage-  
Template.h>
```

Inheritance diagram for activemq::commands::ActiveMQMessageTemplate< T >:

Public Member Functions

- **ActiveMQMessageTemplate** ()
- virtual **~ActiveMQMessageTemplate** () throw ()
- virtual void **acknowledge** () const
- virtual void **onSend** ()
*Allows derived **Message** (p. 1821) classes to perform tasks before a message is sent.*
- virtual bool **equals** (const **DataStructure** *value) const
*Compares the **DataStructure** (p. 1133) passed in to this one, and returns if they are equivalent.*
- virtual void **clearBody** ()
- virtual void **clearProperties** ()
- virtual std::vector< std::string > **getPropertyNames** () const
- virtual bool **propertyExists** (const std::string &name) const
- virtual bool **getBooleanProperty** (const std::string &name) const
- virtual unsigned char **getByteProperty** (const std::string &name) const
- virtual double **getDoubleProperty** (const std::string &name) const
- virtual float **getFloatProperty** (const std::string &name) const
- virtual int **getIntProperty** (const std::string &name) const
- virtual long long **getLongProperty** (const std::string &name) const
- virtual short **getShortProperty** (const std::string &name) const
- virtual std::string **getStringProperty** (const std::string &name) const
- virtual void **setBooleanProperty** (const std::string &name, bool value)
- virtual void **setByteProperty** (const std::string &name, unsigned char value)
- virtual void **setDoubleProperty** (const std::string &name, double value)
- virtual void **setFloatProperty** (const std::string &name, float value)
- virtual void **setIntProperty** (const std::string &name, int value)
- virtual void **setLongProperty** (const std::string &name, long long value)
- virtual void **setShortProperty** (const std::string &name, short value)
- virtual void **setStringProperty** (const std::string &name, const std::string &value)
- virtual std::string **getCMSCorrelationID** () const
- virtual void **setCMSCorrelationID** (const std::string &correlationId)
- virtual int **getCMSDeliveryMode** () const
- virtual void **setCMSDeliveryMode** (int mode)
- virtual const cms::Destination * **getCMSDestination** () const
- virtual void **setCMSDestination** (const cms::Destination *destination)

- virtual long long **getCMSExpiration** () const
- virtual void **setCMSExpiration** (long long expireTime)
- virtual std::string **getCMSMessageID** () const
- virtual void **setCMSMessageID** (const std::string &id AMQCPP_UNUSED)
- virtual int **getCMSPriority** () const
- virtual void **setCMSPriority** (int priority)
- virtual bool **getCMSRedelivered** () const
- virtual void **setCMSRedelivered** (bool redelivered AMQCPP_UNUSED)
- virtual const cms::Destination * **getCMSReplyTo** () const
- virtual void **setCMSReplyTo** (const cms::Destination *destination)
- virtual long long **getCMSTimestamp** () const
- virtual void **setCMSTimestamp** (long long timeStamp)
- virtual std::string **getCMSType** () const
- virtual void **setCMSType** (const std::string &type)

Protected Member Functions

- void **failIfWriteOnlyBody** () const
- void **failIfReadOnlyBody** () const
- void **failIfReadOnlyProperties** () const

```
template<typename T> class activemq::commands::ActiveMQMessageTemplate< T >
```

6.29.1 Constructor & Destructor Documentation

```
6.29.1.1 template<typename T> activemq::commands::ActiveMQ-
MessageTemplate< T >::ActiveMQMessageTemplate ( )
[inline]
```

```
6.29.1.2 template<typename T> virtual activemq::commands::ActiveMQMessage-
Template< T >::~~ActiveMQMessageTemplate ( ) throw () [inline,
virtual]
```

6.29.2 Member Function Documentation

```
6.29.2.1 template<typename T> virtual void activemq::commands::ActiveM-
QMessageTemplate< T >::acknowledge ( ) const [inline,
virtual]
```

```
6.29.2.2 template<typename T> virtual void activemq::commands::Active-
MQMessageTemplate< T >::clearBody ( ) [inline,
virtual]
```

Reimplemented in **activemq::commands::ActiveMQBytesMessage** (p. 168), **activemq::commands::ActiveMQStreamMessage** (p. 361), **activemq::commands::ActiveMQMapMessage** (p. 271), and **activemq::commands::ActiveMQTextMessage** (p. 406).

6.29.2.3 `template<typename T> virtual void activemq::commands::ActiveMQMessageTemplate< T >::clearProperties ()` `[inline, virtual]`

6.29.2.4 `template<typename T> virtual bool activemq::commands::ActiveMQMessageTemplate< T >::equals (const DataStructure * value)` `const [inline, virtual]`

Compares the **DataStructure** (p. 1133) passed in to this one, and returns if they are equivalent.

Equivalent here means that they are of the same type, and that each element of the objects are the same.

Returns

true if **DataStructure** (p. 1133)'s are Equal.

Reimplemented from **activemq::commands::Message** (p. 1827).

Reimplemented in **activemq::commands::ActiveMQBytesMessage** (p. 169), **activemq::commands::ActiveMQStreamMessage** (p. 362), **activemq::commands::ActiveMQMapMessage** (p. 272), **activemq::commands::ActiveMQBlobMessage** (p. 159), **activemq::commands::ActiveMQTextMessage** (p. 408), **activemq::commands::ActiveMQObjectMessage** (p. 303), and **activemq::commands::ActiveMQMessage** (p. 290).

6.29.2.5 `template<typename T> void activemq::commands::ActiveMQMessageTemplate< T >::failIfReadOnlyBody ()` `const [inline, protected]`

6.29.2.6 `template<typename T> void activemq::commands::ActiveMQMessageTemplate< T >::failIfReadOnlyProperties ()` `const [inline, protected]`

6.29.2.7 `template<typename T> void activemq::commands::ActiveMQMessageTemplate< T >::failIfWriteOnlyBody ()` `const [inline, protected]`

6.29.2.8 `template<typename T> virtual bool activemq::commands::ActiveMQMessageTemplate< T >::getBooleanProperty (const std::string & name)` `const [inline, virtual]`

6.29.2.9 `template<typename T> virtual unsigned char activemq::commands::ActiveMQMessageTemplate< T >::getByteProperty (const std::string & name)` `const [inline, virtual]`

6.29.2.10 `template<typename T> virtual std::string activemq::commands::ActiveMQMessageTemplate< T >::getCMSCorrelationID ()` `const [inline, virtual]`

- 6.29.2.11 `template<typename T> virtual int activemq::commands::ActiveMQ-MessageTemplate< T >::getCMSDeliveryMode () const [inline, virtual]`
- 6.29.2.12 `template<typename T> virtual const cms::Destination* activemq::commands::ActiveMQMessageTemplate< T >::getCMSDestination () const [inline, virtual]`
- 6.29.2.13 `template<typename T> virtual long long activemq::commands::ActiveMQMessageTemplate< T >::getCMSExpiration () const [inline, virtual]`
- 6.29.2.14 `template<typename T> virtual std::string activemq::commands::ActiveMQMessageTemplate< T >::getCMSMessageID () const [inline, virtual]`
- 6.29.2.15 `template<typename T> virtual int activemq::commands::ActiveMQMessageTemplate< T >::getCMSPriority () const [inline, virtual]`
- 6.29.2.16 `template<typename T> virtual bool activemq::commands::ActiveMQMessageTemplate< T >::getCMSRedelivered () const [inline, virtual]`
- 6.29.2.17 `template<typename T> virtual const cms::Destination* activemq::commands::ActiveMQMessageTemplate< T >::getCMSReplyTo () const [inline, virtual]`
- 6.29.2.18 `template<typename T> virtual long long activemq::commands::ActiveMQMessageTemplate< T >::getCMSTimestamp () const [inline, virtual]`
- 6.29.2.19 `template<typename T> virtual std::string activemq::commands::ActiveMQMessageTemplate< T >::getCMSType () const [inline, virtual]`
- 6.29.2.20 `template<typename T> virtual double activemq::commands::ActiveMQMessageTemplate< T >::getDoubleProperty (const std::string & name) const [inline, virtual]`
- 6.29.2.21 `template<typename T> virtual float activemq::commands::ActiveMQMessageTemplate< T >::getFloatProperty (const std::string & name) const [inline, virtual]`
- 6.29.2.22 `template<typename T> virtual int activemq::commands::ActiveMQMessageTemplate< T >::getIntProperty (const std::string & name) const [inline, virtual]`

- 6.29.2.23 `template<typename T> virtual long long activemq::commands::ActiveMQ-MessageTemplate< T >::getLongProperty (const std::string & name) const [inline, virtual]`
- 6.29.2.24 `template<typename T> virtual std::vector<std::string> activemq::commands::ActiveMQMessageTemplate< T >::getPropertyNames () const [inline, virtual]`
- 6.29.2.25 `template<typename T> virtual short activemq::commands::ActiveMQ-MessageTemplate< T >::getShortProperty (const std::string & name) const [inline, virtual]`
- 6.29.2.26 `template<typename T> virtual std::string activemq::commands::ActiveMQ-MessageTemplate< T >::getStringProperty (const std::string & name) const [inline, virtual]`
- 6.29.2.27 `template<typename T> virtual void activemq::commands::ActiveMQMessageTemplate< T >::onSend () [inline, virtual]`

Allows derived **Message** (p. 1821) classes to perform tasks before a message is sent.

Reimplemented from **activemq::commands::Message** (p. 1833).

Reimplemented in **activemq::commands::ActiveMQBytesMessage** (p. 171), and **activemq::commands::ActiveMQStreamMessage** (p. 363).

- 6.29.2.28 `template<typename T> virtual bool activemq::commands::ActiveMQ-MessageTemplate< T >::propertyExists (const std::string & name) const [inline, virtual]`
- 6.29.2.29 `template<typename T> virtual void activemq::commands::ActiveMQ-MessageTemplate< T >::setBooleanProperty (const std::string & name, bool value) [inline, virtual]`
- 6.29.2.30 `template<typename T> virtual void activemq::commands::ActiveMQ-MessageTemplate< T >::setByteProperty (const std::string & name, unsigned char value) [inline, virtual]`
- 6.29.2.31 `template<typename T> virtual void activemq::commands::ActiveMQ-MessageTemplate< T >::setCMSCorrelationID (const std::string & correlationId) [inline, virtual]`
- 6.29.2.32 `template<typename T> virtual void activemq::commands::ActiveMQ-MessageTemplate< T >::setCMSDeliveryMode (int mode) [inline, virtual]`

- 6.29.2.33 `template<typename T> virtual void activemq::commands::ActiveMQ-MessageTemplate< T >::setCMSDestination (const cms::Destination * destination) [inline, virtual]`
- 6.29.2.34 `template<typename T> virtual void activemq::commands::ActiveMQ-MessageTemplate< T >::setCMSExpiration (long long expireTime) [inline, virtual]`
- 6.29.2.35 `template<typename T> virtual void activemq::commands::ActiveMQ-MessageTemplate< T >::setCMSMessageID (const std::string &id AMQCPP_UNUSED) [inline, virtual]`
- 6.29.2.36 `template<typename T> virtual void activemq::commands::ActiveMQ-MessageTemplate< T >::setCMSPriority (int priority) [inline, virtual]`
- 6.29.2.37 `template<typename T> virtual void activemq::commands::ActiveMQ-MessageTemplate< T >::setCMSRedelivered (bool redelivered AMQCPP_UNUSED) [inline, virtual]`
- 6.29.2.38 `template<typename T> virtual void activemq::commands::ActiveMQ-MessageTemplate< T >::setCMSReplyTo (const cms::Destination * destination) [inline, virtual]`
- 6.29.2.39 `template<typename T> virtual void activemq::commands::ActiveMQ-MessageTemplate< T >::setCMSTimestamp (long long timeStamp) [inline, virtual]`
- 6.29.2.40 `template<typename T> virtual void activemq::commands::ActiveMQ-MessageTemplate< T >::setCMSType (const std::string & type) [inline, virtual]`
- 6.29.2.41 `template<typename T> virtual void activemq::commands::ActiveMQ-MessageTemplate< T >::setDoubleProperty (const std::string & name, double value) [inline, virtual]`
- 6.29.2.42 `template<typename T> virtual void activemq::commands::ActiveMQ-MessageTemplate< T >::setFloatProperty (const std::string & name, float value) [inline, virtual]`
- 6.29.2.43 `template<typename T> virtual void activemq::commands::ActiveMQ-MessageTemplate< T >::setIntProperty (const std::string & name, int value) [inline, virtual]`
- 6.29.2.44 `template<typename T> virtual void activemq::commands::ActiveMQ-MessageTemplate< T >::setLongProperty (const std::string & name, long long value) [inline, virtual]`

- 6.29.2.45 `template<typename T> virtual void activemq::commands::ActiveMQ-MessageTemplate< T >::setShortProperty (const std::string & name, short value) [inline, virtual]`
- 6.29.2.46 `template<typename T> virtual void activemq::commands::ActiveMQ-MessageTemplate< T >::setStringProperty (const std::string & name, const std::string & value) [inline, virtual]`

The documentation for this class was generated from the following file:

- `src/main/activemq/commands/ActiveMQMessageTemplate.h`

6.30 activemq::commands::ActiveMQObjectMessage Class - Reference

```
#include <src/main/activemq/commands/ActiveMQObjectMessage.h>
```

Inheritance diagram for `activemq::commands::ActiveMQObjectMessage`:

Public Member Functions

- **ActiveMQObjectMessage** ()
- virtual `~ActiveMQObjectMessage` () throw ()
- virtual unsigned char **getDataStructureType** () const
*Get the **DataStructure** (p. 1133) Type as defined in CommandTypes.h.*
- virtual **ActiveMQObjectMessage** * **cloneDataStructure** () const
Clone this object and return a new instance that the caller now owns, this will be an exact copy of this one.
- virtual void **copyDataStructure** (const **DataStructure** *src)
Copy the contents of the passed object into this objects members, overwriting any existing data.
- virtual std::string **toString** () const
*Returns a string containing the information for this **DataStructure** (p. 1133) such as its type and value of its elements.*
- virtual bool **equals** (const **DataStructure** *value) const
*Compares the **DataStructure** (p. 1133) passed in to this one, and returns if they are equivalent.*
- virtual **cms::Message** * **clone** () const
Clone this message exactly, returns a new instance that the caller is required to delete.

Static Public Attributes

- static const unsigned char **ID_ACTIVEMQOBJECTMESSAGE** = 26

6.30.1 Constructor & Destructor Documentation

6.30.1.1 `activemq::commands::ActiveMQObjectMessage::ActiveMQObjectMessage ()`

6.30.1.2 `virtual activemq::commands::ActiveMQObjectMessage::~~ActiveMQObjectMessage () throw () [inline, virtual]`

6.30.2 Member Function Documentation

6.30.2.1 `virtual cms::Message* activemq::commands::ActiveMQObjectMessage::clone () const [inline, virtual]`

Clone this message exactly, returns a new instance that the caller is required to delete.

Returns

new copy of this message

Implements **cms::Message** (p. 1844).

6.30.2.2 `virtual ActiveMQObjectMessage* activemq::commands::ActiveMQObjectMessage::cloneDataStructure () const [virtual]`

Clone this object and return a new instance that the caller now owns, this will be an exact copy of this one.

Returns

new copy of this object.

Reimplemented from **activemq::commands::Message** (p. 1826).

6.30.2.3 `virtual void activemq::commands::ActiveMQObjectMessage::copyDataStructure (const DataStructure * src) [virtual]`

Copy the contents of the passed object into this objects members, overwriting any existing data.

Returns

src - Source Object

Reimplemented from **activemq::commands::Message** (p. 1827).

6.30.2.4 `virtual bool activemq::commands::ActiveMQObjectMessage::equals (const DataStructure * value) const [virtual]`

Compares the **DataStructure** (p. 1133) passed in to this one, and returns if they are equivalent.

Equivalent here means that they are of the same type, and that each element of the objects are the same.

Returns

true if **DataStructure** (p. 1133)'s are Equal.

Reimplemented from **activemq::commands::ActiveMQMessageTemplate< cms::ObjectMessage >** (p. 297).

6.30.2.5 `virtual unsigned char activemq::commands::ActiveMQObjectMessage::getDataStructureType () const [virtual]`

Get the **DataStructure** (p. 1133) Type as defined in CommandTypes.h.

Returns

The type of the data structure

Reimplemented from **activemq::commands::Message** (p. 1829).

6.30.2.6 `virtual std::string activemq::commands::ActiveMQObjectMessage::toString () const [virtual]`

Returns a string containing the information for this **DataStructure** (p. 1133) such as its type and value of its elements.

Returns

formatted string useful for debugging.

Reimplemented from **activemq::commands::Message** (p. 1836).

6.30.3 Field Documentation

6.30.3.1 `const unsigned char activemq::commands::ActiveMQObjectMessage::ID_ACTIVEMQOBJECTMESSAGE = 26 [static]`

The documentation for this class was generated from the following file:

- src/main/activemq/commands/**ActiveMQObjectMessage.h**

6.31 activemq::wireformat::openwire::marshal::generated::ActiveMQObjectMessageMarshaller Class Reference

Marshaling code for Open Wire Format for **ActiveMQObjectMessageMarshaller** (p. 304).

```
#include <src/main/activemq/wireformat/openwire/marshal/generated/ActiveMQObjectMessageMarshaller.h>
```

Inheritance diagram for activemq::wireformat::openwire::marshal::generated::ActiveMQObjectMessageMarshaller:

Public Member Functions

- **ActiveMQObjectMessageMarshaller** ()
- virtual **~ActiveMQObjectMessageMarshaller** ()
- virtual **commands::DataStructure * createObject** () const
Creates a new instance of the class that this class is a marshaling director for.
- virtual unsigned char **getDataStructureType** () const
Gets the DataStructureType that this class marshals/unmarshals.
- virtual void **tightUnmarshal** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataInputStream** *dataIn, **utils::BooleanStream** *bs)
Tight Un-marhsal to the given stream.
- virtual int **tightMarshal1** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **utils::BooleanStream** *bs)
Tight Marhsal to the given stream.
- virtual void **tightMarshal2** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataOutputStream** *dataOut, **utils::BooleanStream** *bs)
Tight Marhsal to the given stream.
- virtual void **looseUnmarshal** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataInputStream** *dataIn)
Loose Un-marhsal to the given stream.
- virtual void **looseMarshal** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataOutputStream** *dataOut)
Tight Marhsal to the given stream.

6.31.1 Detailed Description

Marshaling code for Open Wire Format for **ActiveMQObjectMessageMarshaller** (p. 304).

NOTE!: This file is auto generated - do not modify! if you need to make a change, please see the Java Classes in the activemq-openwire-generator module

6.31 activemq::wireformat::openwire::marshal::generated::ActiveMQObjectMessageMarshaller Class

Reference

305

6.31.2 Constructor & Destructor Documentation

6.31.2.1 **activemq::wireformat::openwire::marshal::generated::ActiveMQObjectMessageMarshaller::ActiveMQObjectMessageMarshaller ()**
[inline]

6.31.2.2 **virtual activemq::wireformat::openwire::marshal::generated::ActiveMQObjectMessageMarshaller::~~ActiveMQObjectMessageMarshaller ()**
[inline, virtual]

6.31.3 Member Function Documentation

6.31.3.1 **virtual commands::DataStructure* activemq::wireformat::openwire::marshal::generated::ActiveMQObjectMessageMarshaller::createObject () const** [virtual]

Creates a new instance of the class that this class is a marshaling director for.

Returns

newly allocated Command

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1120).

6.31.3.2 **virtual unsigned char activemq::wireformat::openwire::marshal::generated::ActiveMQObjectMessageMarshaller::getDataStructureType () const**
[virtual]

Gets the DataStructureType that this class marshals/unmarshals.

Returns

byte Id of this classes DataStructureType

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1122).

6.31.3.3 **virtual void activemq::wireformat::openwire::marshal::generated::ActiveMQObjectMessageMarshaller::looseMarshal (OpenWireFormat * *format*, commands::DataStructure * *command*, decaf::io::DataOutputStream * *ds*)** [virtual]

Tight Marshal to the given stream.

Parameters

<i>format</i>	- The OpenWireFormat properties
<i>command</i>	- the object to Marshal
<i>ds</i>	- DataOutputStream to marshal to

Exceptions

<i>IOException</i>	if an error occurs.
--------------------	---------------------

Reimplemented from **activemq::wireformat::openwire::marshal::generated::MessageMarshaller** (p. 1919).

6.31.3.4 **virtual void activemq::wireformat::openwire::marshal::generated::ActiveMQObjectMessageMarshaller::looseUnmarshal (OpenWireFormat * *format*, commands::DataStructure * *command*, decaf::io::DataInputStream * *dis*)** [virtual]

Loose Un-marhsal to the given stream.

Parameters

<i>format</i>	- The OpenWireFormat properties
<i>command</i>	- the object to Un-Marshal
<i>dis</i>	- the DataInputStream to Un-Marshal from

Exceptions

<i>IOException</i>	if an error occurs.
--------------------	---------------------

Reimplemented from **activemq::wireformat::openwire::marshal::generated::MessageMarshaller** (p. 1919).

6.31.3.5 **virtual int activemq::wireformat::openwire::marshal::generated::ActiveMQObjectMessageMarshaller::tightMarshal1 (OpenWireFormat * *format*, commands::DataStructure * *command*, utils::BooleanStream * *bs*)** [virtual]

Tight Marhsal to the given stream.

Parameters

<i>format</i>	The OpenWireFormat properties
<i>command</i>	The object to Marshal
<i>bs</i>	The boolean stream to marshal to.

Exceptions

<i>IOException</i>	if an error occurs.
--------------------	---------------------

Reimplemented from **activemq::wireformat::openwire::marshal::generated::MessageMarshaller** (p. 1920).

6.31 activemq::wireformat::openwire::marshal::generated::ActiveMQObjectMessageMarshaller Class

Reference

307

6.31.3.6 virtual void activemq::wireformat::openwire::marshal::generated::ActiveMQObjectMessageMarshaller::tightMarshal2 (OpenWireFormat * *format*, commands::DataStructure * *command*, decaf::io::DataOutputStream * *ds*, utils::BooleanStream * *bs*) [virtual]

Tight Marhsal to the given stream.

Parameters

<i>format</i>	- The OpenWireFormat properties
<i>command</i>	- the object to Marshal
<i>ds</i>	- the DataOutputStream to Marshal to
<i>bs</i>	- boolean stream to marshal to.

Exceptions

<i>IOException</i>	if an error occurs.
--------------------	---------------------

Reimplemented from **activemq::wireformat::openwire::marshal::generated::MessageMarshaller** (p. 1921).

6.31.3.7 virtual void activemq::wireformat::openwire::marshal::generated::ActiveMQObjectMessageMarshaller::tightUnmarshal (OpenWireFormat * *format*, commands::DataStructure * *command*, decaf::io::DataInputStream * *dis*, utils::BooleanStream * *bs*) [virtual]

Tight Un-marhsal to the given stream.

Parameters

<i>format</i>	- The OpenWireFormat properties
<i>command</i>	- the object to Un-Marshal
<i>dis</i>	- the DataInputStream to Un-Marshal from
<i>bs</i>	- boolean stream to unmarshal from.

Exceptions

<i>IOException</i>	if an error occurs.
--------------------	---------------------

Reimplemented from **activemq::wireformat::openwire::marshal::generated::MessageMarshaller** (p. 1921).

The documentation for this class was generated from the following file:

- src/main/activemq/wireformat/openwire/marshal/generated/**ActiveMQObjectMessageMarshaller.h**

6.32 activemq::core::ActiveMQProducer Class Reference

```
#include <src/main/activemq/core/ActiveMQProducer.h>
```

Inheritance diagram for activemq::core::ActiveMQProducer:

Public Member Functions

- **ActiveMQProducer** (**ActiveMQSession** *session, const **Pointer**< **commands::ProducerId** > &producerId, const **Pointer**< **commands::ActiveMQDestination** > &destination, long long sendTimeout)

*Constructor, creates an instance of an **ActiveMQProducer** (p. 308).*

- virtual **~ActiveMQProducer** () throw ()
- virtual void **close** ()

Closes this object and deallocates the appropriate resources.

- virtual void **send** (**cms::Message** *message)

Sends the message to the default producer destination, but does not take ownership of the message, caller must still destroy it.

- virtual void **send** (**cms::Message** *message, int deliveryMode, int priority, long long timeToLive)

Sends the message to the default producer destination, but does not take ownership of the message, caller must still destroy it.

- virtual void **send** (const **cms::Destination** *destination, **cms::Message** *message)

Sends the message to the designated destination, but does not take ownership of the message, caller must still destroy it.

- virtual void **send** (const **cms::Destination** *destination, **cms::Message** *message, int deliveryMode, int priority, long long timeToLive)

Sends the message to the designated destination, but does not take ownership of the message, caller must still destroy it.

- virtual void **setDeliveryMode** (int mode)

Sets the delivery mode for this Producer.

- virtual int **getDeliveryMode** () const

Gets the delivery mode for this Producer.

- virtual void **setDisableMessageID** (bool value)

Sets if Message Ids are disabled for this Producer.

- virtual bool **getDisableMessageID** () const

Gets if Message Ids are disabled for this Producer.

- virtual void **setDisableMessageTimeStamp** (bool value)

Sets if Message Time Stamps are disabled for this Producer.

- virtual bool **getDisableMessageTimeStamp** () const

Gets if Message Time Stamps are disabled for this Producer.

- virtual void **setPriority** (int priority)

- Sets the Priority that this Producers sends messages at.*
- virtual int **getPriority** () const
- Gets the Priority level that this producer sends messages at.*
- virtual void **setTimeToLive** (long long time)
- Sets the Time to Live that this Producers sends messages with.*
- virtual long long **getTimeToLive** () const
- Gets the Time to Live that this producer sends messages with.*
- virtual void **setSendTimeout** (long long time)
- Sets the Send Timeout that this Producers sends messages with.*
- virtual long long **getSendTimeout** () const
- Gets the Send Timeout that this producer sends messages with.*
- bool **isClosed** () const
- const **Pointer** < **commands::ProducerInfo** > & **getProducerInfo** () const
- Retries this object ProducerInfo pointer.*
- const **Pointer** < **commands::ProducerId** > & **getProducerId** () const
- Retries this object ProducerId or NULL if closed.*
- virtual void **onProducerAck** (const **commands::ProducerAck** &ack)
- Handles the work of Processing a ProducerAck Command from the Broker.*
- void **dispose** ()
- Performs Producer object cleanup but doesn't attempt to send the Remove command to the broker.*

6.32.1 Constructor & Destructor Documentation

- 6.32.1.1 **activemq::core::ActiveMQProducer::ActiveMQProducer** (**ActiveMQSession** * session, const **Pointer**< **commands::ProducerId** > & producerId, const **Pointer**< **commands::ActiveMQDestination** > & destination, long long sendTimeout)

Constructor, creates an instance of an **ActiveMQProducer** (p. 308).

Parameters

<i>session</i>	The Session which is the parent of this Producer.
<i>producerId</i>	Pointer to a ProducerId object which identifies this producer.
<i>destination</i>	The assigned Destination this Producer sends to, or null if not set. The Producer does not own the Pointer passed.
<i>send-Timeout</i>	The configured send timeout for this Producer.

- 6.32.1.2 **virtual activemq::core::ActiveMQProducer::~~ActiveMQProducer** () throw
() [virtual]

6.32.2 Member Function Documentation

6.32.2.1 `virtual void activemq::core::ActiveMQProducer::close () [virtual]`

Closes this object and deallocates the appropriate resources.

The object is generally no longer usable after calling close.

Exceptions

<i>CMSEException</i>	- If an error occurs while the resource is being closed.
----------------------	--

Implements **cms::Closeable** (p. 816).

6.32.2.2 `void activemq::core::ActiveMQProducer::dispose ()`

Performs Producer object cleanup but doesn't attempt to send the Remove command to the broker.

Called when the parent resource is closed first to avoid the message send and avoid any exceptions that might be thrown from an attempt to send a remove command to a failed transport.

6.32.2.3 `virtual int activemq::core::ActiveMQProducer::getDeliveryMode () const [inline, virtual]`

Gets the delivery mode for this Producer.

Returns

The DeliveryMode

Implements **cms::MessageProducer** (p. 1926).

6.32.2.4 `virtual bool activemq::core::ActiveMQProducer::getDisableMessageID () const [inline, virtual]`

Gets if Message IDs are disabled for this Producer.

Returns

a boolean indicating state enable / disable (true / false) for MessageIds.

Implements **cms::MessageProducer** (p. 1927).

6.32.2.5 `virtual bool activemq::core::ActiveMQProducer::getDisableMessageTimeStamp () const [inline, virtual]`

Gets if Message Time Stamps are disabled for this Producer.

Returns

boolean indicating state of enable / disable (true / false)

Implements **cms::MessageProducer** (p. 1927).

6.32.2.6 `virtual int activemq::core::ActiveMQProducer::getPriority () const`
`[inline, virtual]`

Gets the Priority level that this producer sends messages at.

Returns

int based priority level

Implements **cms::MessageProducer** (p. 1927).

6.32.2.7 `const Pointer<commands::ProducerId>& activemq-`
`::core::ActiveMQProducer::getProducerId () const`
`[inline]`

Retries this object ProducerId or NULL if closed.

Returns

ProducerId Reference

6.32.2.8 `const Pointer<commands::ProducerInfo>& activemq-`
`::core::ActiveMQProducer::getProducerInfo () const`
`[inline]`

Retries this object ProducerInfo pointer.

Returns

ProducerInfo Reference

6.32.2.9 `virtual long long activemq::core::ActiveMQProducer::getSendTimeout ()`
`const [inline, virtual]`

Gets the Send Timeout that this producer sends messages with.

Returns

The default send timeout value in milliseconds.

6.32.2.10 `virtual long long activemq::core::ActiveMQProducer::getTimeToLive ()`
`const [inline, virtual]`

Gets the Time to Live that this producer sends messages with.

Returns

The default time to live value in milliseconds.

Implements **cms::MessageProducer** (p. 1928).

6.32.2.11 `bool activemq::core::ActiveMQProducer::isClosed () const [inline]`

Returns

true if this Producer has been closed.

6.32.2.12 `virtual void activemq::core::ActiveMQProducer::onProducerAck (const`
`commands::ProducerAck & ack) [virtual]`

Handles the work of Processing a ProducerAck Command from the Broker.

Parameters

<i>ack</i>	- The ProducerAck message received from the Broker.
------------	---

6.32.2.13 `virtual void activemq::core::ActiveMQProducer::send (cms::Message *`
`message) [virtual]`

Sends the message to the default producer destination, but does not take ownership of the message, caller must still destroy it.

Uses default values for deliveryMode, priority, and time to live.

Parameters

<i>message</i>	The message to be sent.
----------------	-------------------------

Exceptions

<i>CMSException</i>	- if an internal error occurs while sending the message.
<i>MessageFormat-Exception</i>	- if an Invalid Message is given.
<i>InvalidDestination-Exception</i>	- if a client uses this method with a MessageProducer with an invalid destination.
<i>Unsupported-OperationException</i>	- if a client uses this method with a MessageProducer that did not specify a destination at creation time.

Implements **cms::MessageProducer** (p. 1928).

6.32.2.14 `virtual void activemq::core::ActiveMQProducer::send (cms::Message * message, int deliveryMode, int priority, long long timeToLive) [virtual]`

Sends the message to the default producer destination, but does not take ownership of the message, caller must still destroy it.

Parameters

<i>message</i>	The message to be sent.
<i>delivery-Mode</i>	The delivery mode to be used.
<i>priority</i>	The priority for this message.
<i>timeToLive</i>	The time to live value for this message in milliseconds.

Exceptions

<i>CMSException</i>	- if an internal error occurs while sending the message.
<i>MessageFormat-Exception</i>	- if an Invalid Message is given.
<i>InvalidDestination-Exception</i>	- if a client uses this method with a MessageProducer with an invalid destination.
<i>Unsupported-OperationException</i>	- if a client uses this method with a MessageProducer that did not specify a destination at creation time.

Implements **cms::MessageProducer** (p. 1929).

6.32.2.15 `virtual void activemq::core::ActiveMQProducer::send (const cms::Destination * destination, cms::Message * message) [virtual]`

Sends the message to the designated destination, but does not take ownership of the message, caller must still destroy it.

Uses default values for deliveryMode, priority, and time to live.

Parameters

<i>destination</i>	The destination on which to send the message
<i>message</i>	the message to be sent.

Exceptions

<i>CMSException</i>	- if an internal error occurs while sending the message.
<i>MessageFormat-Exception</i>	- if an Invalid Message is given.
<i>InvalidDestination-Exception</i>	- if a client uses this method with a MessageProducer with an invalid destination.

<i>Unsupported-OperationException</i>	- if a client uses this method with a MessageProducer that did not specify a destination at creation time.
---------------------------------------	--

Implements **cms::MessageProducer** (p. 1929).

6.32.2.16 **virtual void activemq::core::ActiveMQProducer::send (const cms::Destination * *destination*, cms::Message * *message*, int *deliveryMode*, int *priority*, long long *timeToLive*)** [virtual]

Sends the message to the designated destination, but does not take ownership of the message, caller must still destroy it.

Parameters

<i>destination</i>	The destination on which to send the message
<i>message</i>	The message to be sent.
<i>delivery-Mode</i>	The delivery mode to be used.
<i>priority</i>	The priority for this message.
<i>timeToLive</i>	The time to live value for this message in milliseconds.

Exceptions

<i>CMSException</i>	- if an internal error occurs while sending the message.
<i>MessageFormatException</i>	- if an Invalid Message is given.
<i>InvalidDestination-Exception</i>	- if a client uses this method with a MessageProducer with an invalid destination.
<i>Unsupported-OperationException</i>	- if a client uses this method with a MessageProducer that did not specify a destination at creation time.

Implements **cms::MessageProducer** (p. 1930).

6.32.2.17 **virtual void activemq::core::ActiveMQProducer::setDeliveryMode (int *mode*)** [inline, virtual]

Sets the delivery mode for this Producer.

Parameters

<i>mode</i>	- The DeliveryMode to use for Message sends.
-------------	--

Implements **cms::MessageProducer** (p. 1931).

6.32.2.18 `virtual void activemq::core::ActiveMQProducer::setDisableMessageID (bool value) [inline, virtual]`

Sets if Message Ids are disabled for this Producer.

Parameters

<i>value</i>	- boolean indicating enable / disable (true / false)
--------------	--

Implements **cms::MessageProducer** (p. 1931).

6.32.2.19 `virtual void activemq::core::ActiveMQProducer::setDisableMessageTimeStamp (bool value) [inline, virtual]`

Sets if Message Time Stamps are disabled for this Producer.

Parameters

<i>value</i>	- boolean indicating enable / disable (true / false)
--------------	--

Implements **cms::MessageProducer** (p. 1931).

6.32.2.20 `virtual void activemq::core::ActiveMQProducer::setPriority (int priority) [inline, virtual]`

Sets the Priority that this Producers sends messages at.

Parameters

<i>priority</i>	int value for Priority level
-----------------	------------------------------

Implements **cms::MessageProducer** (p. 1932).

6.32.2.21 `virtual void activemq::core::ActiveMQProducer::setSendTimeout (long long time) [inline, virtual]`

Sets the Send Timeout that this Producers sends messages with.

Parameters

<i>time</i>	The new default send timeout value in milliseconds.
-------------	---

6.32.2.22 `virtual void activemq::core::ActiveMQProducer::setTimeToLive (long long time) [inline, virtual]`

Sets the Time to Live that this Producers sends messages with.

Parameters

<i>time</i>	The new default time to live value in milliseconds.
-------------	---

Implements **cms::MessageProducer** (p. 1932).

The documentation for this class was generated from the following file:

- src/main/activemq/core/**ActiveMQProducer.h**

6.33 activemq::util::ActiveMQProperties Class Reference

Implementation of the CMSProperties interface that delegates to a **decaf::util::Properties** (p. 2200) object.

```
#include <src/main/activemq/util/ActiveMQProperties.h>
```

Inheritance diagram for activemq::util::ActiveMQProperties:

Public Member Functions

- **ActiveMQProperties** ()
- virtual **~ActiveMQProperties** () throw ()
- virtual **decaf::util::Properties** & **getProperties** ()
- virtual const **decaf::util::Properties** & **getProperties** () const
- virtual void **setProperties** (decaf::util::Properties &props)
- virtual int **size** () const
Returns the current count of all the Properties that are currently stored in the - Properties object.
- virtual bool **isEmpty** () const
Returns true if the properties object is empty.
- virtual const char * **getProperty** (const std::string &name) const
Looks up the value for the given property.
- virtual std::string **getProperty** (const std::string &name, const std::string &defaultValue) const
Looks up the value for the given property.
- virtual void **setProperty** (const std::string &name, const std::string &value)
Sets the value for a given property.
- virtual bool **hasProperty** (const std::string &name) const
Check to see if the Property exists in the set.
- virtual std::string **remove** (const std::string &name)
Removes the property with the given name.
- virtual std::vector< std::string > **propertyNames** () const
Returns a vector containing all the names of the properties currently stored in the Properties object.

- virtual std::vector< std::pair < std::string, std::string > > **toArray** () const
Method that serializes the contents of the property map to an array.
- virtual void **copy** (const CMSProperties *source)
- virtual CMSProperties * **clone** () const
Clones this object.
- virtual void **clear** ()
Clears all properties from the map.
- virtual std::string **toString** () const
Formats the contents of the Properties Object into a string that can be logged, etc.

6.33.1 Detailed Description

Implementation of the CMSProperties interface that delegates to a **decaf::util::Properties** (p. 2200) object.

Since

2.0

6.33.2 Constructor & Destructor Documentation

6.33.2.1 **activemq::util::ActiveMQProperties::ActiveMQProperties** ()

6.33.2.2 **virtual activemq::util::ActiveMQProperties::~~ActiveMQProperties** ()
throw () [virtual]

6.33.3 Member Function Documentation

6.33.3.1 **virtual void activemq::util::ActiveMQProperties::clear** () [inline, virtual]

Clears all properties from the map.

Implements **cms::CMSProperties** (p. 831).

6.33.3.2 **virtual CMSProperties* activemq::util::ActiveMQProperties::clone** () const
[virtual]

Clones this object.

Returns

a replica of this object.

Implements **cms::CMSProperties** (p. 831).

6.33.3.3 `virtual void activemq::util::ActiveMQProperties::copy (const CMSProperties *
source) [virtual]`

6.33.3.4 `virtual decaf::util::Properties& activemq::util::Active-
MQProperties::getProperties () [inline,
virtual]`

6.33.3.5 `virtual const decaf::util::Properties& activemq::util::Active-
MQProperties::getProperties () const [inline,
virtual]`

6.33.3.6 `virtual const char* activemq::util::ActiveMQProperties::getProperty (const
std::string & name) const [inline, virtual]`

Looks up the value for the given property.

Parameters

<i>name</i>	The name of the property to be looked up.
-------------	---

Returns

the value of the property with the given name, if it exists. If it does not exist, returns NULL.

Implements **cms::CMSProperties** (p. 832).

6.33.3.7 `virtual std::string activemq::util::ActiveMQProperties::getProperty (const
std::string & name, const std::string & defaultValue) const [inline,
virtual]`

Looks up the value for the given property.

Parameters

<i>name</i>	the name of the property to be looked up.
<i>defaultValue</i>	The value to be returned if the given property does not exist.

Returns

The value of the property specified by *name*, if it exists, otherwise the *default-Value*.

Implements **cms::CMSProperties** (p. 832).

6.33.3.8 `virtual bool activemq::util::ActiveMQProperties::hasProperty (const
std::string & name) const [inline, virtual]`

Check to see if the Property exists in the set.

Parameters

<i>name</i>	the name of the property to check
-------------	-----------------------------------

Returns

true if property exists, false otherwise.

Implements **cms::CMSProperties** (p. 833).

6.33.3.9 `virtual bool activemq::util::ActiveMQProperties::isEmpty () const`
[inline, virtual]

Returns true if the properties object is empty.

Returns

true if empty

Implements **cms::CMSProperties** (p. 833).

6.33.3.10 `virtual std::vector<std::string> activemq::util::ActiveMQProperties::propertyNames () const` [inline, virtual]

Returns a vector containing all the names of the properties currently stored in the - Properties object.

Returns

an STL `std::vector<std::string>` with all the currently stored property names.

Implements **cms::CMSProperties** (p. 833).

6.33.3.11 `virtual std::string activemq::util::ActiveMQProperties::remove (const std::string & name)` [inline, virtual]

Removes the property with the given name.

If the property existed in the collection then it is removed and returned, otherwise an empty string is returned.

Parameters

<i>name</i>	the name of the property to be removed.
-------------	---

Returns

the value that was removed from the Properties, or empty string.

Implements **cms::CMSProperties** (p. 833).

6.33.3.12 `virtual void activemq::util::ActiveMQProperties::setProperties (decaf::util::Properties & props) [inline, virtual]`

6.33.3.13 `virtual void activemq::util::ActiveMQProperties::setProperty (const std::string & name, const std::string & value) [inline, virtual]`

Sets the value for a given property.

If the property already exists, overwrites the value.

Parameters

<i>name</i>	The name of the value to be written.
<i>value</i>	The value to be written.

Implements **cms::CMSProperties** (p. 834).

6.33.3.14 `virtual int activemq::util::ActiveMQProperties::size () const [inline, virtual]`

Returns the current count of all the Properties that are currently stored in the Properties object.

Returns

the number of properties currently stored.

Implements **cms::CMSProperties** (p. 834).

6.33.3.15 `virtual std::vector< std::pair< std::string, std::string > > activemq::util::ActiveMQProperties::toArray () const [inline, virtual]`

Method that serializes the contents of the property map to an array.

Returns

list of pairs where the first is the name and the second is the value.

Implements **cms::CMSProperties** (p. 834).

6.33.3.16 `virtual std::string activemq::util::ActiveMQProperties::toString () const`
`[inline, virtual]`

Formats the contents of the Properties Object into a string that can be logged, etc.

Returns

string value of this object.

Implements **cms::CMSProperties** (p. 834).

The documentation for this class was generated from the following file:

- `src/main/activemq/util/ActiveMQProperties.h`

6.34 activemq::commands::ActiveMQQueue Class Reference

```
#include <src/main/activemq/commands/ActiveMQQueue.h>
```

Inheritance diagram for `activemq::commands::ActiveMQQueue`:

Public Member Functions

- **ActiveMQQueue** ()
- **ActiveMQQueue** (const std::string &name)
- virtual **~ActiveMQQueue** () throw ()
- virtual unsigned char **getDataStructureType** () const
*Get the **DataStructure** (p. 1133) Type as defined in CommandTypes.h.*
- virtual **ActiveMQQueue * cloneDataStructure** () const
Clone this object and return a new instance that the caller now owns, this will be an exact copy of this one.
- virtual void **copyDataStructure** (const **DataStructure** *src)
Copy the contents of the passed object into this objects members, overwriting any existing data.
- virtual std::string **toString** () const
*Returns a string containing the information for this **DataStructure** (p. 1133) such as its type and value of its elements.*
- virtual bool **equals** (const **DataStructure** *value) const
*Compares the **DataStructure** (p. 1133) passed in to this one, and returns if they are equivalent.*
- virtual const **cms::Destination** * **getCMSDestination** () const
- virtual **cms::Destination::DestinationType** **getDestinationType** () const
Returns the Type of Destination that this object represents.
- virtual **cms::Destination** * **clone** () const

Creates a new instance of this destination type that is a copy of this one, and returns it.

- virtual void **copy** (const **cms::Destination** &source)
Copies the contents of the given Destination object to this one.
- virtual const **cms::CMSProperties** & **getCMSProperties** () const
Retrieve any properties that might be part of the destination that was specified.
- virtual bool **equals** (const **cms::Destination** &other) const
Compares two Destination instances to determine if they represent the same logic Destination.
- virtual std::string **getQueueName** () const
Gets the name of this queue.

Static Public Attributes

- static const unsigned char **ID_ACTIVEMQQUEUE** = 100

6.34.1 Constructor & Destructor Documentation

6.34.1.1 **activemq::commands::ActiveMQQueue::ActiveMQQueue ()**

6.34.1.2 **activemq::commands::ActiveMQQueue::ActiveMQQueue (const std::string & name)**

6.34.1.3 **virtual activemq::commands::ActiveMQQueue::~~ActiveMQQueue ()**
throw () *[virtual]*

6.34.2 Member Function Documentation

6.34.2.1 **virtual cms::Destination* activemq::commands::ActiveMQQueue::clone () const** *[inline, virtual]*

Creates a new instance of this destination type that is a copy of this one, and returns it.

Returns

cloned copy of this object

Implements **cms::Destination** (p. 1211).

6.34.2.2 **virtual ActiveMQQueue* activemq::commands::ActiveMQQueue::clone-DataStructure () const** *[virtual]*

Clone this obbject and return a new instance that the caller now owns, this will be an exact copy of this one.

Returns

new copy of this object.

Reimplemented from **activemq::commands::ActiveMQDestination** (p. 249).

6.34.2.3 `virtual void activemq::commands::ActiveMQQueue::copy (const cms::Destination & source) [inline, virtual]`

Copies the contents of the given Destination object to this one.

Parameters

<i>source</i>	The source Destination object.
---------------	--------------------------------

Implements **cms::Destination** (p. 1211).

6.34.2.4 `virtual void activemq::commands::ActiveMQQueue::copyDataStructure (const DataStructure * src) [virtual]`

Copy the contents of the passed object into this objects members, overwriting any existing data.

Returns

src - Source Object

Reimplemented from **activemq::commands::ActiveMQDestination** (p. 249).

6.34.2.5 `virtual bool activemq::commands::ActiveMQQueue::equals (const DataStructure * value) const [virtual]`

Compares the **DataStructure** (p. 1133) passed in to this one, and returns if they are equivalent.

Equivalent here means that they are of the same type, and that each element of the objects are the same.

Returns

true if **DataStructure** (p. 1133)'s are Equal.

Reimplemented from **activemq::commands::ActiveMQDestination** (p. 250).

6.34.2.6 `virtual bool activemq::commands::ActiveMQQueue::equals (const cms::Destination & other) const [virtual]`

Compares two Destination instances to determine if they represent the same logic - Destination.

Parameters

<i>other</i>	The other destination to compare this one to.
--------------	---

Returns

true if the two destinations are the same.

Implements **cms::Destination** (p. 1212).

6.34.2.7 `virtual const cms::Destination* activemq::commands::ActiveMQQueue::getCMSDestination () const [inline, virtual]`

Returns

the **cms::Destination** (p. 1210) interface pointer that the objects that derive from this class implement.

Reimplemented from **activemq::commands::ActiveMQDestination** (p. 251).

6.34.2.8 `virtual const cms::CMSProperties& activemq::commands::ActiveMQQueue::getCMSProperties () const [inline, virtual]`

Retrieve any properties that might be part of the destination that was specified.

This is a deviation from the JMS spec but necessary due to C++ restrictions.

Returns

A {const} reference to a CMSProperties object.

Implements **cms::Destination** (p. 1212).

6.34.2.9 `virtual unsigned char activemq::commands::ActiveMQQueue::getDataStructureType () const [virtual]`

Get the **DataStructure** (p. 1133) Type as defined in CommandTypes.h.

Returns

The type of the data structure

Reimplemented from **activemq::commands::ActiveMQDestination** (p. 251).

6.34.2.10 `virtual cms::Destination::DestinationType activemq::commands::ActiveMQQueue::getDestinationType () const [inline, virtual]`

Returns the Type of Destination that this object represents.

Returns

int type qualifier.

Implements **activemq::commands::ActiveMQDestination** (p. 251).

References cms::Destination::QUEUE.

6.34.2.11 `virtual std::string activemq::commands::ActiveMQQueue::getQueueName () const [inline, virtual]`

Gets the name of this queue.

Returns

The queue name.

Exceptions

<i>CMSException</i>	- If an internal error occurs.
---------------------	--------------------------------

Implements **cms::Queue** (p. 2222).

6.34.2.12 `virtual std::string activemq::commands::ActiveMQQueue::toString () const [virtual]`

Returns a string containing the information for this **DataStructure** (p. 1133) such as its type and value of its elements.

Returns

formatted string useful for debugging.

Reimplemented from **activemq::commands::ActiveMQDestination** (p. 255).

6.34.3 Field Documentation

6.34.3.1 `const unsigned char activemq::commands::ActiveMQQueue::ID_ACTIVEM-
QUEUE = 100 [static]`

The documentation for this class was generated from the following file:

- src/main/activemq/commands/**ActiveMQQueue.h**

6.35 activemq::core::ActiveMQQueueBrowser Class Reference

```
#include <src/main/activemq/core/ActiveMQQueueBrowser.h>
```

Inheritance diagram for activemq::core::ActiveMQQueueBrowser:

Public Member Functions

- **ActiveMQQueueBrowser** (**ActiveMQSession** *session, const **Pointer**< **commands::ConsumerId** > &consumerId, const **Pointer**< **commands::ActiveMQDestination** > &destination, const std::string &selector, bool dispatchAsync)
- virtual ~**ActiveMQQueueBrowser** () throw ()
- virtual const **cms::Queue** * **getQueue** () const
- virtual std::string **getMessageSelector** () const
- virtual **cms::MessageEnumeration** * **getEnumeration** ()
Gets a pointer to an Enumeration object for browsing the Messages currently in the Queue in the order that a client would receive them.
- virtual void **close** ()
Closes this object and deallocates the appropriate resources.
- virtual bool **hasMoreMessages** ()
Returns true if there are more Message in the Browser that can be retrieved via the nextMessage method.
- virtual **cms::Message** * **nextMessage** ()
Returns the Next Message in the Queue if one is present, if no more Message's are available then an Exception is thrown.

Friends

- class **Browser**

6.35.1 Constructor & Destructor Documentation

- 6.35.1.1 **activemq::core::ActiveMQQueueBrowser::ActiveMQQueueBrowser** (**ActiveMQSession** * session, const **Pointer**< **commands::ConsumerId** > & consumerId, const **Pointer**< **commands::ActiveMQDestination** > & destination, const std::string & selector, bool dispatchAsync)
- 6.35.1.2 virtual **activemq::core::ActiveMQQueueBrowser::~~ActiveMQQueueBrowser** () throw () [virtual]

6.35.2 Member Function Documentation

6.35.2.1 `virtual void activemq::core::ActiveMQQueueBrowser::close ()`
`[virtual]`

Closes this object and deallocates the appropriate resources.

The object is generally no longer usable after calling close.

Exceptions

<i>CMSEException</i>	- If an error occurs while the resource is being closed.
----------------------	--

Implements **cms::Closeable** (p. 816).

6.35.2.2 `virtual cms::MessageEnumeration* activemq::core-
::ActiveMQQueueBrowser::getEnumeration ()`
`[virtual]`

Gets a pointer to an Enumeration object for browsing the Messages currently in the Queue in the order that a client would receive them.

The pointer returned is owned by the browser and should not be deleted by the client application.

Returns

a pointer to a Queue Enumeration, this Pointer is owned by the QueueBrowser and should not be deleted by the client.

Exceptions

<i>CMSEException</i>	if an internal error occurs.
----------------------	------------------------------

Implements **cms::QueueBrowser** (p. 2228).

6.35.2.3 `virtual std::string activemq::core::ActiveMQQueueBrowser::getMessage-
Selector () const` `[virtual]`

Returns

the MessageSelector that is used on when this browser was created or empty string if no selector was present.

Exceptions

<i>CMSEException</i>	if an internal error occurs.
----------------------	------------------------------

Implements **cms::QueueBrowser** (p. 2228).

6.35.2.4 `virtual const cms::Queue* activemq::core::ActiveMQQueueBrowser::get-Queue () const` [virtual]

Returns

the Queue that this browser is listening on.

Exceptions

<i>CMSEException</i> if an internal error occurs.

Implements **cms::QueueBrowser** (p. 2229).

6.35.2.5 `virtual bool activemq::core::ActiveMQQueueBrowser::hasMoreMessages ()` [virtual]

Returns true if there are more Message in the Browser that can be retrieved via the `nextMessage` method.

If this method returns false and the `nextMessage` method is called then an Exception will be thrown.

Returns

true if more Message's are available in the Browser.

Implements **cms::MessageEnumeration** (p. 1904).

6.35.2.6 `virtual cms::Message* activemq::core::ActiveMQQueueBrowser::next-Message ()` [virtual]

Returns the Next Message in the Queue if one is present, if no more Message's are available then an Exception is thrown.

If a Message object pointer is returned then that object becomes the property of the caller and must be deleted by the caller when finished.

Returns

The next Message in the Queue.

Exceptions

<i>CMSEException</i> if no more Message's currently in the Queue.

Implements **cms::MessageEnumeration** (p. 1905).

6.35.3.1 friend class **Browser** [friend]

The documentation for this class was generated from the following file:

- src/main/activemq/core/**ActiveMQQueueBrowser.h**

6.36 activemq::wireformat::openwire::marshal::generated::ActiveMQQueueMarshaller Class Reference

Marshaling code for Open Wire Format for **ActiveMQQueueMarshaller** (p. 329).

```
#include <src/main/activemq/wireformat/openwire/marshal/generated/-  
ActiveMQQueueMarshaller.h>
```

Inheritance diagram for activemq::wireformat::openwire::marshal::generated::ActiveMQQueueMarshaller:

Public Member Functions

- **ActiveMQQueueMarshaller** ()
- virtual **~ActiveMQQueueMarshaller** ()
- virtual **commands::DataStructure * createObject** () const
Creates a new instance of the class that this class is a marshaling director for.
- virtual unsigned char **getDataStructureType** () const
Gets the DataStructureType that this class marshals/unmarshals.
- virtual void **tightUnmarshal** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataInputStream** *dataIn, **utils::BooleanStream** *bs)
Tight Un-marhsal to the given stream.
- virtual int **tightMarshal1** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **utils::BooleanStream** *bs)
Tight Marshal to the given stream.
- virtual void **tightMarshal2** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataOutputStream** *dataOut, **utils::BooleanStream** *bs)
Tight Marshal to the given stream.
- virtual void **looseUnmarshal** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataInputStream** *dataIn)
Loose Un-marhsal to the given stream.
- virtual void **looseMarshal** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataOutputStream** *dataOut)
Tight Marshal to the given stream.

6.36.1 Detailed Description

Marshaling code for Open Wire Format for **ActiveMQQueueMarshaller** (p. 329).

NOTE!: This file is auto generated - do not modify! if you need to make a change, please see the Java Classes in the activemq-openwire-generator module

6.36.2 Constructor & Destructor Documentation

6.36.2.1 **activemq::wireformat::openwire::marshal::generated::ActiveMQQueueMarshaller::ActiveMQQueueMarshaller ()**
[inline]

6.36.2.2 **virtual activemq::wireformat::openwire::marshal::generated::ActiveMQQueueMarshaller::~~ActiveMQQueueMarshaller ()** [inline, virtual]

6.36.3 Member Function Documentation

6.36.3.1 **virtual commands::DataStructure* activemq::wireformat::openwire::marshal::generated::ActiveMQQueueMarshaller::createObject ()** const
[virtual]

Creates a new instance of the class that this class is a marshaling director for.

Returns

newly allocated Command

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1120).

6.36.3.2 **virtual unsigned char activemq::wireformat::openwire::marshal::generated::ActiveMQQueueMarshaller::getDataStructureType ()** const
[virtual]

Gets the DataStructureType that this class marshals/unmarshals.

Returns

byte Id of this classes DataStructureType

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1122).

6.36 activemq::wireformat::openwire::marshal::generated::ActiveMQQueue-Marshaller Class

Reference

331

6.36.3.3 virtual void activemq::wireformat::openwire::marshal::generated::ActiveMQQueueMarshaller::looseMarshal (OpenWireFormat * *format*, commands::DataStructure * *command*, decaf::io::DataOutputStream * *ds*) [virtual]

Tight Marhsal to the given stream.

Parameters

<i>format</i>	- The OpenWireFormat properties
<i>command</i>	- the object to Marshal
<i>ds</i>	- DataOutputStream to marshal to

Exceptions

<i>IOException</i>	if an error occurs.
--------------------	---------------------

Reimplemented from **activemq::wireformat::openwire::marshal::generated::ActiveMQDestinationMarshaller** (p. 259).

6.36.3.4 virtual void activemq::wireformat::openwire::marshal::generated::ActiveMQQueueMarshaller::looseUnmarshal (OpenWireFormat * *format*, commands::DataStructure * *command*, decaf::io::DataInputStream * *dis*) [virtual]

Loose Un-marhsal to the given stream.

Parameters

<i>format</i>	- The OpenWireFormat properties
<i>command</i>	- the object to Un-Marshal
<i>dis</i>	- the DataInputStream to Un-Marshal from

Exceptions

<i>IOException</i>	if an error occurs.
--------------------	---------------------

Reimplemented from **activemq::wireformat::openwire::marshal::generated::ActiveMQDestinationMarshaller** (p. 259).

6.36.3.5 virtual int activemq::wireformat::openwire::marshal::generated::ActiveMQQueueMarshaller::tightMarshal1 (OpenWireFormat * *format*, commands::DataStructure * *command*, utils::BooleanStream * *bs*) [virtual]

Tight Marhsal to the given stream.

Parameters

<i>format</i>	The OpenwireFormat properties
<i>command</i>	The object to Marshal
<i>bs</i>	The boolean stream to marshal to.

Exceptions

<i>IOException</i>	if an error occurs.
--------------------	---------------------

Reimplemented from **activemq::wireformat::openwire::marshal::generated::ActiveMQDestinationMarshaller** (p. 260).

6.36.3.6 virtual void **activemq::wireformat::openwire::marshal::generated::ActiveMQQueueMarshaller::tightMarshal2** (**OpenWireFormat** * *format*, **commands::DataStructure** * *command*, **decaf::io::DataOutputStream** * *ds*, **utils::BooleanStream** * *bs*) [virtual]

Tight Marshal to the given stream.

Parameters

<i>format</i>	- The OpenWireFormat properties
<i>command</i>	- the object to Marshal
<i>ds</i>	- the DataOutputStream to Marshal to
<i>bs</i>	- boolean stream to marshal to.

Exceptions

<i>IOException</i>	if an error occurs.
--------------------	---------------------

Reimplemented from **activemq::wireformat::openwire::marshal::generated::ActiveMQDestinationMarshaller** (p. 260).

6.36.3.7 virtual void **activemq::wireformat::openwire::marshal::generated::ActiveMQQueueMarshaller::tightUnmarshal** (**OpenWireFormat** * *format*, **commands::DataStructure** * *command*, **decaf::io::DataInputStream** * *dis*, **utils::BooleanStream** * *bs*) [virtual]

Tight Un-marshal to the given stream.

Parameters

<i>format</i>	- The OpenWireFormat properties
<i>command</i>	- the object to Un-Marshal
<i>dis</i>	- the DataInputStream to Un-Marshal from
<i>bs</i>	- boolean stream to unmarshal from.

Exceptions

<i>IOException</i>	if an error occurs.
--------------------	---------------------

Reimplemented from **activemq::wireformat::openwire::marshal::generated::ActiveMQDestinationMarshaller** (p. 261).

The documentation for this class was generated from the following file:

- src/main/activemq/wireformat/openwire/marshal/generated/**ActiveMQQueueMarshaller.h**

6.37 activemq::core::ActiveMQSession Class Reference

```
#include <src/main/activemq/core/ActiveMQSession.h>
```

Inheritance diagram for activemq::core::ActiveMQSession:

Public Member Functions

- **ActiveMQSession** (**ActiveMQConnection** *connection, const **Pointer**<**commands::SessionId** > &id, **cms::Session::AcknowledgeMode** ackMode, const **decaf::util::Properties** &properties)
- virtual ~**ActiveMQSession** () throw ()
- virtual void **redispatch** (**MessageDispatchChannel** &unconsumedMessages)
Redispatches the given set of unconsumed messages to the consumers.
- virtual void **start** ()
Stops asynchronous message delivery.
- virtual void **stop** ()
Starts asynchronous message delivery.
- bool **isStarted** () const
Indicates whether or not the session is currently in the started state.
- virtual bool **isAutoAcknowledge** () const
- virtual bool **isDupsOkAcknowledge** () const
- virtual bool **isClientAcknowledge** () const
- virtual bool **isIndividualAcknowledge** () const
- void **fire** (const **exceptions::ActiveMQException** &ex)
Fires the given exception to the exception listener of the connection.
- virtual void **dispatch** (const **Pointer**< **MessageDispatch** > &message)
Dispatches a message to a particular consumer.
- virtual void **close** ()
Closes this session as well as any active child consumers or producers.
- virtual void **commit** ()
Commits all messages done in this transaction and releases any locks currently held.

- virtual void **rollback** ()
Rolls back all messages done in this transaction and releases any locks currently held.
- virtual void **recover** ()
Stops message delivery in this session, and restarts message delivery with the oldest unacknowledged message.
- virtual **cms::MessageConsumer** * **createConsumer** (const **cms::Destination** *destination)
Creates a MessageConsumer for the specified destination.
- virtual **cms::MessageConsumer** * **createConsumer** (const **cms::Destination** *destination, const std::string &selector)
Creates a MessageConsumer for the specified destination, using a message selector.
- virtual **cms::MessageConsumer** * **createConsumer** (const **cms::Destination** *destination, const std::string &selector, bool noLocal)
Creates a MessageConsumer for the specified destination, using a message selector.
- virtual **cms::MessageConsumer** * **createDurableConsumer** (const **cms::Topic** *destination, const std::string &name, const std::string &selector, bool noLocal=false)
Creates a durable subscriber to the specified topic, using a Message selector.
- virtual **cms::MessageProducer** * **createProducer** (const **cms::Destination** *destination)
Creates a MessageProducer to send messages to the specified destination.
- virtual **cms::QueueBrowser** * **createBrowser** (const **cms::Queue** *queue)
Creates a new QueueBrowser to peek at Messages on the given Queue.
- virtual **cms::QueueBrowser** * **createBrowser** (const **cms::Queue** *queue, const std::string &selector)
Creates a new QueueBrowser to peek at Messages on the given Queue.
- virtual **cms::Queue** * **createQueue** (const std::string &queueName)
Creates a queue identity given a Queue name.
- virtual **cms::Topic** * **createTopic** (const std::string &topicName)
Creates a topic identity given a Queue name.
- virtual **cms::TemporaryQueue** * **createTemporaryQueue** ()
Creates a TemporaryQueue object.
- virtual **cms::TemporaryTopic** * **createTemporaryTopic** ()
Creates a TemporaryTopic object.
- virtual **cms::Message** * **createMessage** ()
Creates a new Message.
- virtual **cms::BytesMessage** * **createBytesMessage** ()
Creates a BytesMessage.
- virtual **cms::BytesMessage** * **createBytesMessage** (const unsigned char *bytes, int bytesSize)
Creates a BytesMessage and sets the payload to the passed value.
- virtual **cms::StreamMessage** * **createStreamMessage** ()
Creates a new StreamMessage.
- virtual **cms::TextMessage** * **createTextMessage** ()

Creates a new TextMessage.

- virtual **cms::TextMessage** * **createTextMessage** (const std::string &text)
Creates a new TextMessage and set the text to the value given.
- virtual **cms::MapMessage** * **createMapMessage** ()
Creates a new MapMessage.
- virtual **cms::Session::AcknowledgeMode** **getAcknowledgeMode** () const
Returns the acknowledgment mode of the session.
- virtual bool **isTransacted** () const
Gets if the Sessions is a Transacted Session.
- virtual void **unsubscribe** (const std::string &name)
Unsubscribes a durable subscription that has been created by a client.
- void **send** (**cms::Message** *message, **ActiveMQProducer** *producer, **util::Usage** *usage)
Sends a message from the Producer specified using this session's connection the message will be sent using the best available means depending on the configuration of the connection.
- **cms::ExceptionListener** * **getExceptionListener** ()
This method gets any registered exception listener of this sessions connection and returns it.
- const **commands::SessionInfo** & **getSessionInfo** () const
Gets the Session Information object for this session, if the session is closed than this method throws an exception.
- const **commands::SessionId** & **getSessionId** () const
Gets the Session Id object for this session, if the session is closed than this method throws an exception.
- **ActiveMQConnection** * **getConnection** () const
*Gets the **ActiveMQConnection** (p. 187) that is associated with this session.*
- **Pointer**< **threads::Scheduler** > **getScheduler** () const
Gets a Pointer to this Session's Scheduler instance.
- long long **getLastDeliveredSequenceId** () const
Gets the currently set Last Delivered Sequence Id.
- void **setLastDeliveredSequenceId** (long long value)
Sets the value of the Last Delivered Sequence Id.
- void **oneway** (**Pointer**< **commands::Command** > command)
Sends a Command to the broker without requesting any Response be returned.
- **Pointer**< **commands::Response** > **syncRequest** (**Pointer**< **commands::Command** > command, unsigned int timeout=0)
Sends a synchronous request and returns the response from the broker.
- void **addConsumer** (**ActiveMQConsumer** *consumer)
Adds a MessageConsumer to this session registering it with the Connection and store a reference to it so the session can ensure that all resources are closed when the session is closed.
- void **removeConsumer** (const **Pointer**< **commands::ConsumerId** > &consumerId)
Dispose of a MessageConsumer from this session.

- void **addProducer** (**ActiveMQProducer** *producer)
Adds a MessageProducer to this session registering it with the Connection and store a reference to it so the session can ensure that all resources are closed when the session is closed.
- void **removeProducer** (**ActiveMQProducer** *producer)
Dispose of a MessageProducer from this session.
- virtual void **doStartTransaction** ()
Starts if not already start a Transaction for this Session.
- **Pointer** < **ActiveMQTransactionContext** > **getTransactionContext** ()
Gets the Pointer to this Session's TransactionContext.
- void **acknowledge** ()
Request that the Session inform all its consumers to Acknowledge all Message's that have been received so far.
- void **deliverAcks** ()
Request that this Session inform all of its consumers to deliver their pending acks.
- void **clearMessagesInProgress** ()
Request that this Session inform all of its consumers to clear all messages that are currently in progress.
- void **wakeup** ()
Causes the Session to wakeup its executor and ensure all messages are dispatched.
- **Pointer**< **commands::ConsumerId** > **getNextConsumerId** ()
Get the Next available Consumer Id.
- **Pointer**< **commands::ProducerId** > **getNextProducerId** ()
Get the Next available Producer Id.
- void **doClose** ()
Performs the actual Session close operations.
- void **dispose** ()
*Cleans up the Session object's resources without attempting to send the Remove command to the broker, this can be called from **ActiveMQConnection** (p. 187) when it knows that the transport is down and the doClose method would throw an exception when it attempt to send the Remove Command.*

Protected Attributes

- SessionConfig * **config**
- **Pointer**< **commands::SessionInfo** > **sessionInfo**
SessionInfo for this Session.
- **Pointer** < **ActiveMQTransactionContext** > **transaction**
Transaction Management object.
- **ActiveMQConnection** * **connection**
Connection.
- **ConsumersMap** **consumers**
Map of consumers.
- **AtomicBoolean** **closed**

Indicates that this connection has been closed, it is no longer usable after this becomes true.

- `std::auto_ptr < ActiveMQSessionExecutor > executor`

Sends incoming messages to the registered consumers.

- `cms::Session::AcknowledgeMode ackMode`

This Sessions Acknowledgment mode.

- `util::LongSequenceGenerator producerIds`

Next available Producer Id.

- `util::LongSequenceGenerator producerSequenceIds`

Next available Producer Sequence Id.

- `util::LongSequenceGenerator consumerIds`

Next available Consumer Id.

- `long long lastDeliveredSequenceId`

Last Delivered Sequence Id.

Friends

- class **ActiveMQSessionExecutor**

6.37.1 Constructor & Destructor Documentation

6.37.1.1 `activemq::core::ActiveMQSession::ActiveMQSession (ActiveMQConnection * connection, const Pointer< commands::SessionId > & id, cms::Session::AcknowledgeMode ackMode, const decaf::util::Properties & properties)`

6.37.1.2 `virtual activemq::core::ActiveMQSession::~~ActiveMQSession () throw ()`
[virtual]

6.37.2 Member Function Documentation

6.37.2.1 `void activemq::core::ActiveMQSession::acknowledge ()`

Request that the Session inform all its consumers to Acknowledge all Message's that have been received so far.

6.37.2.2 `void activemq::core::ActiveMQSession::addConsumer (ActiveMQConsumer * consumer)`

Adds a MessageConsumer to this session registering it with the Connection and store a reference to it so the session can ensure that all resources are closed when the session is closed.

Parameters

<i>consumer</i>	The ActiveMQConsumer (p. 234) instance to add to this session.
-----------------	---

Exceptions

<i>ActiveMQException</i>	if an internal error occurs.
--------------------------	------------------------------

6.37.2.3 void activemq::core::ActiveMQSession::addProducer (ActiveMQProducer * producer)

Adds a MessageProducer to this session registering it with the Connection and store a reference to it so the session can ensure that all resources are closed when the session is closed.

Parameters

<i>consumer</i>	The ActiveMQProducer (p. 308) instance to add to this session.
-----------------	---

Exceptions

<i>ActiveMQException</i>	if an internal error occurs.
--------------------------	------------------------------

6.37.2.4 void activemq::core::ActiveMQSession::clearMessagesInProgress ()

Request that this Session inform all of its consumers to clear all messages that are currently in progress.

6.37.2.5 virtual void activemq::core::ActiveMQSession::close () [virtual]

Closes this session as well as any active child consumers or producers.

Exceptions

<i>CMSEException</i>	- If an internal error occurs.
----------------------	--------------------------------

Implements **cms::Session** (p. 2365).

6.37.2.6 virtual void activemq::core::ActiveMQSession::commit () [virtual]

Commits all messages done in this transaction and releases any locks currently held.

Exceptions

<i>CMSEException</i>	- If an internal error occurs.
<i>IllegalStateException</i>	- if the method is not called by a transacted session.

Implements **cms::Session** (p. 2366).

Reimplemented in **activemq::core::ActiveMQXASession** (p. 437).

6.37.2.7 `virtual cms::QueueBrowser* activemq::core::ActiveMQSession::createBrowser (const cms::Queue * queue)`
[virtual]

Creates a new QueueBrowser to peek at Messages on the given Queue.

Parameters

<i>queue</i>	the Queue to browse
--------------	---------------------

Returns

New QueueBrowser that is owned by the caller.

Exceptions

<i>CMSException</i>	- If an internal error occurs.
<i>InvalidDestination-Exception</i>	- if the destination given is invalid.

Implements **cms::Session** (p. 2366).

6.37.2.8 `virtual cms::QueueBrowser* activemq::core::ActiveMQSession::createBrowser (const cms::Queue * queue, const std::string & selector)`
[virtual]

Creates a new QueueBrowser to peek at Messages on the given Queue.

Parameters

<i>queue</i>	the Queue to browse
<i>selector</i>	the Message selector to filter which messages are browsed.

Returns

New QueueBrowser that is owned by the caller.

Exceptions

<i>CMSException</i>	- If an internal error occurs.
<i>InvalidDestination-Exception</i>	- if the destination given is invalid.

Implements **cms::Session** (p. 2367).

6.37.2.9 **virtual cms::BytesMessage* activemq::core::ActiveMQSession::createBytesMessage ()** [virtual]

Creates a BytesMessage.

Exceptions

<i>CMSException</i>	- If an internal error occurs.
---------------------	--------------------------------

Implements **cms::Session** (p. 2367).

6.37.2.10 **virtual cms::BytesMessage* activemq::core::ActiveMQSession::createBytesMessage (const unsigned char * bytes, int bytesSize)** [virtual]

Creates a BytesMessage and sets the payload to the passed value.

Parameters

<i>bytes</i>	an array of bytes to set in the message
<i>bytesSize</i>	the size of the bytes array, or number of bytes to use

Exceptions

<i>CMSException</i>	- If an internal error occurs.
---------------------	--------------------------------

Implements **cms::Session** (p. 2367).

6.37.2.11 **virtual cms::MessageConsumer* activemq::core::ActiveMQSession::createConsumer (const cms::Destination * destination)** [virtual]

Creates a MessageConsumer for the specified destination.

Parameters

<i>destination</i>	the Destination that this consumer receiving messages for.
--------------------	--

Returns

pointer to a new MessageConsumer that is owned by the caller (caller deletes)

Exceptions

<i>CMSException</i>	- If an internal error occurs.
<i>InvalidDestination-Exception</i>	- if an invalid destination is specified.

Implements **cms::Session** (p. 2368).

6.37.2.12 `virtual cms::MessageConsumer* activemq::core::ActiveMQSession-
::createConsumer (const cms::Destination * destination, const std::string &
selector) [virtual]`

Creates a MessageConsumer for the specified destination, using a message selector.

Parameters

<i>destination</i>	the Destination that this consumer receiving messages for.
<i>selector</i>	the Message Selector to use

Returns

pointer to a new MessageConsumer that is owned by the caller (caller deletes)

Exceptions

<i>CMSException</i>	- If an internal error occurs.
<i>InvalidDestination-Exception</i>	- if an invalid destination is specified.
<i>InvalidSelector-Exception</i>	- if the message selector is invalid.

Implements **cms::Session** (p. 2368).

6.37.2.13 `virtual cms::MessageConsumer* activemq::core::ActiveMQSession-
::createConsumer (const cms::Destination * destination, const std::string &
selector, bool noLocal) [virtual]`

Creates a MessageConsumer for the specified destination, using a message selector.

Parameters

<i>destination</i>	the Destination that this consumer receiving messages for.
<i>selector</i>	the Message Selector to use
<i>noLocal</i>	if true, and the destination is a topic, inhibits the delivery of messages published by its own connection. The behavior for NoLocal is not specified if the destination is a queue.

Returns

pointer to a new MessageConsumer that is owned by the caller (caller deletes)

Exceptions

<i>CMSException</i>	- If an internal error occurs.
<i>InvalidDestination-Exception</i>	- if an invalid destination is specified.
<i>InvalidSelector-Exception</i>	- if the message selector is invalid.

Implements **cms::Session** (p. 2369).

6.37.2.14 `virtual cms::MessageConsumer* activemq::core::ActiveMQSession::createDurableConsumer (const cms::Topic * destination, const std::string & name, const std::string & selector, bool noLocal = false) [virtual]`

Creates a durable subscriber to the specified topic, using a Message selector.

Sessions that create durable consumers must use the same client Id as was used the last time the subscription was created in order to receive all messages that were delivered while the client was offline.

Parameters

<i>destination</i>	the topic to subscribe to
<i>name</i>	The name used to identify the subscription
<i>selector</i>	the Message Selector to use
<i>noLocal</i>	if true, and the destination is a topic, inhibits the delivery of messages published by its own connection. The behavior for NoLocal is not specified if the destination is a queue.

Returns

pointer to a new durable MessageConsumer that is owned by the caller (caller deletes)

Exceptions

<i>CMSException</i>	- If an internal error occurs.
<i>InvalidDestination-Exception</i>	- if an invalid destination is specified.
<i>InvalidSelector-Exception</i>	- if the message selector is invalid.

Implements **cms::Session** (p. 2370).

6.37.2.15 `virtual cms::MapMessage* activemq::core::ActiveMQSession::createMapMessage () [virtual]`

Creates a new MapMessage.

Exceptions

<i>CMSEException</i>	- If an internal error occurs.
----------------------	--------------------------------

Implements **cms::Session** (p. 2371).

6.37.2.16 virtual **cms::Message*** **activemq::core::ActiveMQSession::createMessage ()** [virtual]

Creates a new Message.

Exceptions

<i>CMSEException</i>	- If an internal error occurs.
----------------------	--------------------------------

Implements **cms::Session** (p. 2371).

6.37.2.17 virtual **cms::MessageProducer*** **activemq::core::ActiveMQSession::createProducer (const cms::Destination * destination)** [virtual]

Creates a MessageProducer to send messages to the specified destination.

Parameters

<i>destination</i>	the Destination to send on
--------------------	----------------------------

Returns

New MessageProducer that is owned by the caller.

Exceptions

<i>CMSEException</i>	- If an internal error occurs.
<i>InvalidDestinationException</i>	- if an invalid destination is specified.

Implements **cms::Session** (p. 2371).

6.37.2.18 virtual **cms::Queue*** **activemq::core::ActiveMQSession::createQueue (const std::string & queueName)** [virtual]

Creates a queue identity given a Queue name.

Parameters

<i>queueName</i>	the name of the new Queue
------------------	---------------------------

Returns

new Queue pointer that is owned by the caller.

Exceptions

<i>CMSEException</i> - If an internal error occurs.

Implements **cms::Session** (p. 2372).

6.37.2.19 **virtual cms::StreamMessage* activemq::core::ActiveMQSession::createStreamMessage ()** [virtual]

Creates a new StreamMessage.

Exceptions

<i>CMSEException</i> - If an internal error occurs.

Implements **cms::Session** (p. 2372).

6.37.2.20 **virtual cms::TemporaryQueue* activemq::core::ActiveMQSession::createTemporaryQueue ()** [virtual]

Creates a TemporaryQueue object.

Returns

new TemporaryQueue pointer that is owned by the caller.

Exceptions

<i>CMSEException</i> - If an internal error occurs.

Implements **cms::Session** (p. 2373).

6.37.2.21 **virtual cms::TemporaryTopic* activemq::core::ActiveMQSession::createTemporaryTopic ()** [virtual]

Creates a TemporaryTopic object.

Exceptions

<i>CMSEException</i> - If an internal error occurs.

Implements **cms::Session** (p. 2373).

6.37.2.22 virtual **cms::TextMessage*** **activemq::core::ActiveMQSession::createTextMessage ()** [virtual]

Creates a new TextMessage.

Exceptions

<i>CMSEException</i>	- If an internal error occurs.
----------------------	--------------------------------

Implements **cms::Session** (p. 2373).

6.37.2.23 virtual **cms::TextMessage*** **activemq::core::ActiveMQSession::createTextMessage (const std::string & text)** [virtual]

Creates a new TextMessage and set the text to the value given.

Parameters

<i>text</i>	the initial text for the message
-------------	----------------------------------

Exceptions

<i>CMSEException</i>	- If an internal error occurs.
----------------------	--------------------------------

Implements **cms::Session** (p. 2374).

6.37.2.24 virtual **cms::Topic*** **activemq::core::ActiveMQSession::createTopic (const std::string & topicName)** [virtual]

Creates a topic identity given a Queue name.

Parameters

<i>topicName</i>	the name of the new Topic
------------------	---------------------------

Returns

new Topic pointer that is owned by the caller.

Exceptions

<i>CMSEException</i>	- If an internal error occurs.
----------------------	--------------------------------

Implements **cms::Session** (p. 2374).

6.37.2.25 void activemq::core::ActiveMQSession::deliverAcks ()

Request that this Session inform all of its consumers to deliver their pending acks.

6.37.2.26 virtual void activemq::core::ActiveMQSession::dispatch (const Pointer< MessageDispatch > & message) [virtual]

Dispatches a message to a particular consumer.

Parameters

<i>message</i>	- the message to be dispatched
----------------	--------------------------------

Implements **activemq::core::Dispatcher** (p. 1235).

6.37.2.27 void activemq::core::ActiveMQSession::dispose ()

Cleans up the Session object's resources without attempting to send the Remove command to the broker, this can be called from **ActiveMQConnection** (p. 187) when it knows that the transport is down and the doClose method would throw an exception when it attempt to send the Remove Command.

6.37.2.28 void activemq::core::ActiveMQSession::doClose ()

Performs the actual Session close operations.

This method is meant for use by **ActiveMQConnection** (p. 187), the connection object calls this when it has been closed to skip some of the extraneous processing done by the client level close method.

6.37.2.29 virtual void activemq::core::ActiveMQSession::doStartTransaction ()
[virtual]

Starts if not already start a Transaction for this Session.

If the session is not a Transacted Session then an exception is thrown. If a transaction is already in progress then this method has no effect.

Exceptions

<i>ActiveMQException</i>	if this is not a Transacted Session.
--------------------------	--------------------------------------

Reimplemented in **activemq::core::ActiveMQXASession** (p. 437).

6.37.2.30 `void activemq::core::ActiveMQSession::fire (const exceptions::ActiveMQException & ex)`

Fires the given exception to the exception listener of the connection.

6.37.2.31 `virtual cms::Session::AcknowledgeMode activemq::core::ActiveMQSession::getAcknowledgeMode () const [virtual]`

Returns the acknowledgment mode of the session.

Returns

the Sessions Acknowledge Mode

Exceptions

<i>CMSException</i>	- If an internal error occurs.
---------------------	--------------------------------

Implements **cms::Session** (p. 2374).

6.37.2.32 `ActiveMQConnection* activemq::core::ActiveMQSession::getConnection () const [inline]`

Gets the **ActiveMQConnection** (p. 187) that is associated with this session.

6.37.2.33 `cms::ExceptionListener* activemq::core::ActiveMQSession::getExceptionListener ()`

This method gets any registered exception listener of this sessions connection and returns it.

Mainly intended for use by the objects that this session creates so that they can notify the client of exceptions that occur in the context of another thread.

Returns

cms::ExceptionListener (p. 1286) pointer or NULL

6.37.2.34 `long long activemq::core::ActiveMQSession::getLastDeliveredSequenceId () const [inline]`

Gets the currently set Last Delivered Sequence Id.

Returns

long long containing the sequence id of the last delivered Message.

6.37.2.35 `Pointer<commands::ConsumerId> activemq-
::core::ActiveMQSession::getNextConsumerId ()`

Get the Next available Consumer Id.

Returns

the next id in the sequence.

6.37.2.36 `Pointer<commands::ProducerId> activemq::core::ActiveMQSession-
::getNextProducerId ()`

Get the Next available Producer Id.

Returns

the next id in the sequence.

6.37.2.37 `Pointer<threads::Scheduler> activemq::core::ActiveMQSession::get-
Scheduler () const`

Gets a Pointer to this Session's Scheduler instance.

6.37.2.38 `const commands::SessionId& activemq::core::ActiveMQSession::get-
SessionId () const [inline]`

Gets the Session Id object for this session, if the session is closed than this method throws an exception.

Returns

SessionId Reference

6.37.2.39 `const commands::SessionInfo& activemq::core::ActiveMQSession::get-
SessionInfo () const [inline]`

Gets the Session Information object for this session, if the session is closed than this method throws an exception.

Returns

SessionInfo Reference

6.37.2.40 **Pointer<ActiveMQTransactionContext> activemq-
::core::ActiveMQSession::getTransactionContext ()**
[inline]

Gets the Pointer to this Session's TransactionContext.

Returns

a Pointer to this Session's TransactionContext

6.37.2.41 **virtual bool activemq::core::ActiveMQSession::isAutoAcknowledge ()**
const [inline, virtual]

Reimplemented in **activemq::core::ActiveMQXASession** (p. 438).

References cms::Session::AUTO_ACKNOWLEDGE.

6.37.2.42 **virtual bool activemq::core::ActiveMQSession::isClientAcknowledge ()**
const [inline, virtual]

References cms::Session::CLIENT_ACKNOWLEDGE.

6.37.2.43 **virtual bool activemq::core::ActiveMQSession::isDupsOkAcknowledge ()**
const [inline, virtual]

References cms::Session::DUPS_OK_ACKNOWLEDGE.

6.37.2.44 **virtual bool activemq::core::ActiveMQSession::isIndividualAcknowledge ()**
const [inline, virtual]

References cms::Session::INDIVIDUAL_ACKNOWLEDGE.

6.37.2.45 **bool activemq::core::ActiveMQSession::isStarted ()** **const**

Indicates whether or not the session is currently in the started state.

6.37.2.46 **virtual bool activemq::core::ActiveMQSession::isTransacted ()** **const**
[virtual]

Gets if the Sessions is a Transacted Session.

Returns

transacted true - false.

Exceptions

<i>CMSException</i>	- If an internal error occurs.
---------------------	--------------------------------

Implements **cms::Session** (p. 2375).

Reimplemented in **activemq::core::ActiveMQXASession** (p. 438).

6.37.2.47 `void activemq::core::ActiveMQSession::oneway (Pointer< commands::Command > command)`

Sends a Command to the broker without requesting any Response be returned.

Parameters

<i>command</i>	The message to send to the Broker.
----------------	------------------------------------

Exceptions

<i>ActiveMQException</i>	if not currently connected, or if the operation fails for any reason.
--------------------------	---

6.37.2.48 `virtual void activemq::core::ActiveMQSession::recover ()` [virtual]

Stops message delivery in this session, and restarts message delivery with the oldest unacknowledged message.

All consumers deliver messages in a serial order. Acknowledging a received message automatically acknowledges all messages that have been delivered to the client.

Restarting a session causes it to take the following actions:

- Stop message delivery
- Mark all messages that might have been delivered but not acknowledged as "re-delivered"
- Restart the delivery sequence including all unacknowledged messages that had been previously delivered. Redelivered messages do not have to be delivered in exactly their original delivery order.

Exceptions

<i>CMSException</i>	- if the CMS provider fails to stop and restart message delivery due to some internal error.
<i>IllegalStateException</i>	- if the method is called by a transacted session.

Implements **cms::Session** (p. 2375).

6.37.2.49 **virtual void activemq::core::ActiveMQSession::redispatch (MessageDispatchChannel & *unconsumedMessages*)** [virtual]

Redispatches the given set of unconsumed messages to the consumers.

Parameters

<i>unconsumedMessages</i>	- unconsumed messages to be redelivered.
---------------------------	--

6.37.2.50 **void activemq::core::ActiveMQSession::removeConsumer (const Pointer< commands::ConsumerId > & *consumerId*)**

Dispose of a MessageConsumer from this session.

Removes it from the Connection and clean up any resources associated with it.

Parameters

<i>consumerId</i>	The ConsumerId of the MessageConsumer to remove from this Session.
-------------------	--

Exceptions

<i>ActiveMQException</i>	if an internal error occurs.
--------------------------	------------------------------

6.37.2.51 **void activemq::core::ActiveMQSession::removeProducer (ActiveMQProducer * *producer*)**

Dispose of a MessageProducer from this session.

Removes it from the Connection and clean up any resources associated with it.

Parameters

<i>producerId</i>	The ProducerId of the MessageProducer to remove from this session.
-------------------	--

Exceptions

<i>ActiveMQException</i>	if an internal error occurs.
--------------------------	------------------------------

6.37.2.52 **virtual void activemq::core::ActiveMQSession::rollback ()** [virtual]

Rolls back all messages done in this transaction and releases any locks currently held.

Exceptions

<i>CMSEException</i>	- If an internal error occurs.
<i>IllegalStateException</i>	- if the method is not called by a transacted session.

Implements **cms::Session** (p. 2376).

Reimplemented in **activemq::core::ActiveMQXASession** (p. 439).

6.37.2.53 `void activemq::core::ActiveMQSession::send (cms::Message * message, ActiveMQProducer * producer, util::Usage * usage)`

Sends a message from the Producer specified using this session's connection the message will be sent using the best available means depending on the configuration of the connection.

Asynchronous sends will be chosen if at all possible.

Parameters

<i>message</i>	The message to send to the broker.
<i>producer</i>	The sending Producer
<i>usage</i>	Pointer to a Usage tracker which if set will be increased by the size of the given message.

Exceptions

<i>CMSEException</i>

6.37.2.54 `void activemq::core::ActiveMQSession::setLastDeliveredSequenceId (long long value) [inline]`

Sets the value of the Last Delivered Sequence Id.

Parameters

<i>value</i>	The new value to assign to the Last Delivered Sequence Id property.
--------------	---

6.37.2.55 `virtual void activemq::core::ActiveMQSession::start () [virtual]`

Stops asynchronous message delivery.

Implements **cms::Startable** (p. 2534).

6.37.2.56 `virtual void activemq::core::ActiveMQSession::stop () [virtual]`

Starts asynchronous message delivery.

Implements **cms::Stoppable** (p. 2602).

6.37.2.57 `Pointer<commands::Response> activemq::core::ActiveMQSession::syncRequest (Pointer< commands::Command > command, unsigned int timeout = 0)`

Sends a synchronous request and returns the response from the broker.

Converts any error responses into an exception.

Parameters

<i>command</i>	The command to send to the broker.
<i>timeout</i>	The time to wait for a response, default is zero or infinite.

Returns

Pointer to a Response object that the broker has returned for the Command sent.

Exceptions

<i>ActiveMQException</i>	thrown if an error response was received from the broker, or if any other error occurred.
--------------------------	---

6.37.2.58 `virtual void activemq::core::ActiveMQSession::unsubscribe (const std::string & name) [virtual]`

Unsubscribes a durable subscription that has been created by a client.

This method deletes the state being maintained on behalf of the subscriber by its provider. It is erroneous for a client to delete a durable subscription while there is an active MessageConsumer or Subscriber for the subscription, or while a consumed message is part of a pending transaction or has not been acknowledged in the session.

Parameters

<i>name</i>	The name used to identify this subscription
-------------	---

Exceptions

<i>CMSEException</i>	- If an internal error occurs.
----------------------	--------------------------------

Implements **cms::Session** (p. 2376).

6.37.2.59 `void activemq::core::ActiveMQSession::wakeup ()`

Causes the Session to wakeup its executor and ensure all messages are dispatched.

6.37.3 Friends And Related Function Documentation

6.37.3.1 friend class **ActiveMQSessionExecutor** [friend]

6.37.4 Field Documentation

6.37.4.1 **cms::Session::AcknowledgeMode** **activemq::core::ActiveMQSession::ackMode** [protected]

This Sessions Acknowledgment mode.

6.37.4.2 **AtomicBoolean** **activemq::core::ActiveMQSession::closed** [protected]

Indicates that this connection has been closed, it is no longer usable after this becomes true.

6.37.4.3 **SessionConfig*** **activemq::core::ActiveMQSession::config** [protected]

6.37.4.4 **ActiveMQConnection*** **activemq::core::ActiveMQSession::connection** [protected]

Connection.

6.37.4.5 **util::LongSequenceGenerator** **activemq::core::ActiveMQSession::consumerIds** [protected]

Next available Consumer Id.

6.37.4.6 **ConsumersMap** **activemq::core::ActiveMQSession::consumers** [protected]

Map of consumers.

6.37.4.7 **std::auto_ptr<ActiveMQSessionExecutor>** **activemq::core::ActiveMQSession::executor** [protected]

Sends incoming messages to the registered consumers.

6.37.4.8 **long long** **activemq::core::ActiveMQSession::lastDeliveredSequenceId** [protected]

Last Delivered Sequence Id.

6.37.4.9 `util::LongSequenceGenerator` `activemq::core::ActiveMQSession-
::producerIds` [protected]

Next available Producer Id.

6.37.4.10 `util::LongSequenceGenerator` `activemq::core::ActiveMQSession-
::producerSequenceIds` [protected]

Next available Producer Sequence Id.

6.37.4.11 `Pointer<commands::SessionInfo>` `activemq::core::ActiveMQSession-
::sessionInfo` [protected]

SessionInfo for this Session.

6.37.4.12 `Pointer<ActiveMQTransactionContext>` `activemq::core::ActiveMQ-
Session::transaction` [protected]

Transaction Management object.

The documentation for this class was generated from the following file:

- `src/main/activemq/core/ActiveMQSession.h`

6.38 activemq::core::ActiveMQSessionExecutor Class Reference

Delegate dispatcher for a single session.

```
#include <src/main/activemq/core/ActiveMQSessionExecutor.h>
```

Inheritance diagram for `activemq::core::ActiveMQSessionExecutor`:

Public Member Functions

- **ActiveMQSessionExecutor** (**ActiveMQSession** *session)
Creates an un-started executor for the given session.
- virtual **~ActiveMQSessionExecutor** ()
*Calls **stop()** (p. 359) then **clear()** (p. 356).*
- virtual void **execute** (const **Pointer**< **MessageDispatch** > &data)
Executes the dispatch.
- virtual void **executeFirst** (const **Pointer**< **MessageDispatch** > &data)
Executes the dispatch.

- virtual void **clearMessagesInProgress** ()
Removes all messages in the Dispatch Channel so that non are delivered.
- virtual bool **hasUnconsumedMessages** () const
- virtual void **wakeup** ()
wakeup this executer and dispatch any pending messages.
- virtual void **start** ()
Starts the dispatching.
- virtual void **stop** ()
Stops dispatching.
- virtual void **close** ()
Terminates the dispatching thread.
- virtual bool **isRunning** () const
- virtual bool **isEmpty** ()
- virtual void **clear** ()
Removes all queued messages and destroys them.
- virtual bool **iterate** ()
*Iterates on the **MessageDispatchChannel** (p. 1886) sending all pending messages to the Consumers they are destined for.*
- std::vector< **Pointer** < **MessageDispatch** > > **getUnconsumedMessages** ()

6.38.1 Detailed Description

Delegate dispatcher for a single session.

Contains a thread to provide for asynchronous dispatching.

6.38.2 Constructor & Destructor Documentation

6.38.2.1 `activemq::core::ActiveMQSessionExecutor::ActiveMQSessionExecutor (ActiveMQSession * session)`

Creates an un-started executor for the given session.

6.38.2.2 `virtual activemq::core::ActiveMQSessionExecutor::~~ActiveMQSession-Executor () [virtual]`

Calls **stop()** (p. 359) then **clear()** (p. 356).

6.38.3 Member Function Documentation

6.38.3.1 `virtual void activemq::core::ActiveMQSessionExecutor::clear () [inline, virtual]`

Removes all queued messages and destroys them.

6.38.3.2 virtual void **activemq::core::ActiveMQSessionExecutor::clearMessagesInProgress** () [inline, virtual]

Removes all messages in the Dispatch Channel so that non are delivered.

6.38.3.3 virtual void **activemq::core::ActiveMQSessionExecutor::close** () [inline, virtual]

Terminates the dispatching thread.

Once this is called, the executor is no longer usable.

6.38.3.4 virtual void **activemq::core::ActiveMQSessionExecutor::execute** (const **Pointer< MessageDispatch > & data**) [virtual]

Executes the dispatch.

Adds the given data to the end of the queue.

Parameters

<i>data</i>	- the data to be dispatched.
-------------	------------------------------

6.38.3.5 virtual void **activemq::core::ActiveMQSessionExecutor::executeFirst** (const **Pointer< MessageDispatch > & data**) [virtual]

Executes the dispatch.

Adds the given data to the beginning of the queue.

Parameters

<i>data</i>	- the data to be dispatched.
-------------	------------------------------

6.38.3.6 std::vector< **Pointer< MessageDispatch >** > **activemq::core::ActiveMQSessionExecutor::getUnconsumedMessages** () [inline]

Returns

a vector containing all the unconsumed messages, this clears the Message - Dispatch Channel when called.

```
6.38.3.7 virtual bool activemq::core::ActiveMQSessionExecutor-  
::hasUnconsumedMessages ( ) const [inline,  
virtual]
```

Returns

true if there are any pending messages in the dispatch channel.

```
6.38.3.8 virtual bool activemq::core::ActiveMQSessionExecutor::isEmpty ( )  
[inline, virtual]
```

Returns

true if there are no messages in the Dispatch Channel.

```
6.38.3.9 virtual bool activemq::core::ActiveMQSessionExecutor::isRunning ( )  
const [inline, virtual]
```

Returns

true indicates if the executor is started

```
6.38.3.10 virtual bool activemq::core::ActiveMQSessionExecutor::iterate ( )  
[virtual]
```

Iterates on the **MessageDispatchChannel** (p. 1886) sending all pending messages to the Consumers they are destined for.

Returns

false if there are no more messages to dispatch.

Implements **activemq::threads::Task** (p. 2676).

```
6.38.3.11 virtual void activemq::core::ActiveMQSessionExecutor::start ( )  
[virtual]
```

Starts the dispatching.

6.38.3.12 virtual void **activemq::core::ActiveMQSessionExecutor::stop** ()
[virtual]

Stops dispatching.

6.38.3.13 virtual void **activemq::core::ActiveMQSessionExecutor::wakeup** ()
[virtual]

wakeup this executor and dispatch any pending messages.

The documentation for this class was generated from the following file:

- src/main/activemq/core/**ActiveMQSessionExecutor.h**

6.39 activemq::commands::ActiveMQStreamMessage Class - Reference

```
#include <src/main/activemq/commands/ActiveMQStreamMessage.h>
```

Inheritance diagram for **activemq::commands::ActiveMQStreamMessage**:

Public Member Functions

- **ActiveMQStreamMessage** ()
- virtual **~ActiveMQStreamMessage** () throw ()
- virtual unsigned char **getDataStructureType** () const
*Get the **DataStructure** (p. 1133) Type as defined in CommandTypes.h.*
- virtual **ActiveMQStreamMessage * cloneDataStructure** () const
Clone this object and return a new instance that the caller now owns, this will be an exact copy of this one.
- virtual void **copyDataStructure** (const **DataStructure** *src)
Copy the contents of the passed object into this objects members, overwriting any existing data.
- virtual std::string **toString** () const
*Returns a string containing the information for this **DataStructure** (p. 1133) such as its type and value of its elements.*
- virtual bool **equals** (const **DataStructure** *value) const
*Compares the **DataStructure** (p. 1133) passed in to this one, and returns if they are equivalent.*
- virtual void **onSend** ()
*Allows derived **Message** (p. 1821) classes to perform tasks before a message is sent.*
- virtual **cms::StreamMessage * clone** () const

Clone this message exactly, returns a new instance that the caller is required to delete.

- virtual void **clearBody** ()
Clears out the body of the message.
- virtual void **reset** ()
- virtual bool **readBoolean** () const
Reads a Boolean from the Stream message stream.
- virtual void **writeBoolean** (bool value)
Writes a boolean to the Stream message stream as a 1-byte value.
- virtual unsigned char **readByte** () const
Reads a Byte from the Stream message stream.
- virtual void **writeByte** (unsigned char value)
Writes a byte to the Stream message stream as a 1-byte value.
- virtual int **readBytes** (std::vector< unsigned char > &value) const
Reads a byte array from the Stream message stream.
- virtual void **writeBytes** (const std::vector< unsigned char > &value)
Writes a byte array to the Stream message stream using the vector size as the number of bytes to write.
- virtual int **readBytes** (unsigned char *buffer, int length) const
Reads a portion of the Stream message stream.
- virtual void **writeBytes** (const unsigned char *value, int offset, int length)
Writes a portion of a byte array to the Stream message stream.
- virtual char **readChar** () const
Reads a Char from the Stream message stream.
- virtual void **writeChar** (char value)
Writes a char to the Stream message stream as a 1-byte value.
- virtual float **readFloat** () const
Reads a 32 bit float from the Stream message stream.
- virtual void **writeFloat** (float value)
Writes a float to the Stream message stream as a 4 byte value.
- virtual double **readDouble** () const
Reads a 64 bit double from the Stream message stream.
- virtual void **writeDouble** (double value)
Writes a double to the Stream message stream as a 8 byte value.
- virtual short **readShort** () const
Reads a 16 bit signed short from the Stream message stream.
- virtual void **writeShort** (short value)
Writes a signed short to the Stream message stream as a 2 byte value.
- virtual unsigned short **readUnsignedShort** () const
Reads a 16 bit unsigned short from the Stream message stream.
- virtual void **writeUnsignedShort** (unsigned short value)
Writes a unsigned short to the Stream message stream as a 2 byte value.
- virtual int **readInt** () const
Reads a 32 bit signed integer from the Stream message stream.

- virtual void **writeInt** (int value)
Writes a signed int to the Stream message stream as a 4 byte value.
- virtual long long **readLong** () const
Reads a 64 bit long from the Stream message stream.
- virtual void **writeLong** (long long value)
Writes a long long to the Stream message stream as a 8 byte value.
- virtual std::string **readString** () const
Reads an ASCII String from the Stream message stream.
- virtual void **writeString** (const std::string &value)
Writes an ASCII String to the Stream message stream.

Static Public Attributes

- static const unsigned char **ID_ACTIVEMQSTREAMMESSAGE** = 27

6.39.1 Constructor & Destructor Documentation

6.39.1.1 **activemq::commands::ActiveMQStreamMessage::ActiveMQStreamMessage ()**

6.39.1.2 **virtual activemq::commands::ActiveMQStreamMessage::~ActiveMQStreamMessage () throw ()**
[virtual]

6.39.2 Member Function Documentation

6.39.2.1 **virtual void activemq::commands::ActiveMQStreamMessage::clearBody ()** [virtual]

Clears out the body of the message.

This does not clear the headers or properties.

Exceptions

<i>CMSEException</i>	- if an internal error occurs.
----------------------	--------------------------------

Reimplemented from **activemq::commands::ActiveMQMessageTemplate< cms::StreamMessage >** (p. 296).

6.39.2.2 **virtual cms::StreamMessage* activemq::commands::ActiveMQStreamMessage::clone () const** [inline, virtual]

Clone this message exactly, returns a new instance that the caller is required to delete.

Returns

new copy of this message

Implements **cms::Message** (p. 1844).

6.39.2.3 `virtual ActiveMQStreamMessage* activemq::commands-
::ActiveMQStreamMessage::cloneDataStructure () const
[virtual]`

Clone this object and return a new instance that the caller now owns, this will be an exact copy of this one.

Returns

new copy of this object.

Reimplemented from **activemq::commands::Message** (p. 1826).

6.39.2.4 `virtual void activemq::commands::ActiveMQStream-
Message::copyDataStructure (const DataStructure * src)
[virtual]`

Copy the contents of the passed object into this objects members, overwriting any existing data.

Returns

src - Source Object

Reimplemented from **activemq::commands::Message** (p. 1827).

6.39.2.5 `virtual bool activemq::commands::ActiveMQStreamMessage::equals (
const DataStructure * value) const [virtual]`

Compares the **DataStructure** (p. 1133) passed in to this one, and returns if they are equivalent.

Equivalent here means that they are of the same type, and that each element of the objects are the same.

Returns

true if **DataStructure** (p. 1133)'s are Equal.

Reimplemented from **activemq::commands::ActiveMQMessageTemplate< cms::StreamMessage >** (p. 297).

6.39.2.6 `virtual unsigned char activemq::commands::ActiveMQStreamMessage::getDataStructureType () const`
`[virtual]`

Get the **DataStream** (p. 1133) Type as defined in CommandTypes.h.

Returns

The type of the data structure

Reimplemented from **activemq::commands::Message** (p. 1829).

6.39.2.7 `virtual void activemq::commands::ActiveMQStreamMessage::onSend ()`
`[virtual]`

Allows derived **Message** (p. 1821) classes to perform tasks before a message is sent.

Reimplemented from **activemq::commands::ActiveMQMessageTemplate< cms::StreamMessage >** (p. 299).

6.39.2.8 `virtual bool activemq::commands::ActiveMQStreamMessage::read-Boolean () const` `[virtual]`

Reads a Boolean from the Stream message stream.

Returns

boolean value from stream

Exceptions

<i>CMSException</i>	- if the CMS provider fails to read the message due to some internal error.
<i>MessageEOF-Exception</i>	- if unexpected end of message stream has been reached.
<i>MessageFormat-Exception</i>	- if this type conversion is invalid.
<i>MessageNot-ReadableException</i>	- if the message is in write-only mode.

Implements **cms::StreamMessage** (p. 2608).

6.39.2.9 `virtual unsigned char activemq::commands::ActiveMQStreamMessage::readByte () const` `[virtual]`

Reads a Byte from the Stream message stream.

Returns

unsigned char value from stream

Exceptions

<i>CMSException</i>	- if the CMS provider fails to read the message due to some internal error.
<i>MessageEOF-Exception</i>	- if unexpected end of message stream has been reached.
<i>MessageFormat-Exception</i>	- if this type conversion is invalid.
<i>MessageNot-ReadableException</i>	- if the message is in write-only mode.

Implements **cms::StreamMessage** (p. 2609).

6.39.2.10 `virtual int activemq::commands::ActiveMQStreamMessage::readBytes (std::vector< unsigned char > & value) const [virtual]`

Reads a byte array from the Stream message stream.

If the length of vector value is less than the number of bytes remaining to be read from the stream, the vector should be filled. A subsequent call reads the next increment, and so on.

If the number of bytes remaining in the stream is less than the length of vector value, the bytes should be read into the vector. The return value of the total number of bytes read will be less than the length of the vector, indicating that there are no more bytes left to be read from the stream. The next read of the stream returns -1.

Parameters

<i>value</i>	buffer to place data in
--------------	-------------------------

Returns

the total number of bytes read into the buffer, or -1 if there is no more data because the end of the stream has been reached

Exceptions

<i>CMSException</i>	- if the CMS provider fails to read the message due to some internal error.
<i>MessageEOF-Exception</i>	- if unexpected end of message stream has been reached.
<i>MessageFormat-Exception</i>	- if this type conversion is invalid.
<i>MessageNot-ReadableException</i>	- if the message is in write-only mode.

Implements **cms::StreamMessage** (p. 2609).

6.39.2.11 `virtual int activemq::commands::ActiveMQStreamMessage::readBytes (unsigned char * buffer, int length) const` [virtual]

Reads a portion of the Stream message stream.

If the length of array value is less than the number of bytes remaining to be read from the stream, the array should be filled. A subsequent call reads the next increment, and so on.

If the number of bytes remaining in the stream is less than the length of array value, the bytes should be read into the array. The return value of the total number of bytes read will be less than the length of the array, indicating that there are no more bytes left to be read from the stream. The next read of the stream returns -1.

If length is negative, or length is greater than the length of the array value, then an `CMSException` is thrown. No bytes will be read from the stream for this exception case.

Parameters

<i>buffer</i>	the buffer into which the data is read
<i>length</i>	the number of bytes to read; must be less than or equal to value.length

Returns

the total number of bytes read into the buffer, or -1 if there is no more data because the end of the stream has been reached

Exceptions

<i>CMSException</i>	- if the CMS provider fails to read the message due to some internal error.
<i>MessageEOF-Exception</i>	- if unexpected end of message stream has been reached.
<i>MessageFormat-Exception</i>	- if this type conversion is invalid.
<i>MessageNot-ReadableException</i>	- if the message is in write-only mode.

Implements **cms::StreamMessage** (p. 2610).

6.39.2.12 `virtual char activemq::commands::ActiveMQStreamMessage::readChar () const` [virtual]

Reads a Char from the Stream message stream.

Returns

char value from stream

Exceptions

<i>CMSException</i>	- if the CMS provider fails to read the message due to some internal error.
<i>MessageEOF-Exception</i>	- if unexpected end of message stream has been reached.
<i>MessageFormat-Exception</i>	- if this type conversion is invalid.
<i>MessageNot-ReadableException</i>	- if the message is in write-only mode.

Implements **cms::StreamMessage** (p. 2611).

6.39.2.13 virtual double **activemq::commands::ActiveMQStreamMessage::readDouble** () const [virtual]

Reads a 64 bit double from the Stream message stream.

Returns

double value from stream

Exceptions

<i>CMSException</i>	- if the CMS provider fails to read the message due to some internal error.
<i>MessageEOF-Exception</i>	- if unexpected end of message stream has been reached.
<i>MessageFormat-Exception</i>	- if this type conversion is invalid.
<i>MessageNot-ReadableException</i>	- if the message is in write-only mode.

Implements **cms::StreamMessage** (p. 2612).

6.39.2.14 virtual float **activemq::commands::ActiveMQStreamMessage::readFloat** () const [virtual]

Reads a 32 bit float from the Stream message stream.

Returns

double value from stream

Exceptions

<i>CMSException</i>	- if the CMS provider fails to read the message due to some internal error.
<i>MessageEOF-Exception</i>	- if unexpected end of message stream has been reached.
<i>MessageFormat-Exception</i>	- if this type conversion is invalid.
<i>MessageNot-ReadableException</i>	- if the message is in write-only mode.

Implements **cms::StreamMessage** (p. 2612).

6.39.2.15 `virtual int activemq::commands::ActiveMQStreamMessage::readInt ()
const [virtual]`

Reads a 32 bit signed integer from the Stream message stream.

Returns

int value from stream

Exceptions

<i>CMSException</i>	- if the CMS provider fails to read the message due to some internal error.
<i>MessageEOF-Exception</i>	- if unexpected end of message stream has been reached.
<i>MessageFormat-Exception</i>	- if this type conversion is invalid.
<i>MessageNot-ReadableException</i>	- if the message is in write-only mode.

Implements **cms::StreamMessage** (p. 2613).

6.39.2.16 `virtual long long activemq::commands::ActiveMQStreamMessage::read-
Long () const [virtual]`

Reads a 64 bit long from the Stream message stream.

Returns

long long value from stream

Exceptions

<i>CMSException</i>	- if the CMS provider fails to read the message due to some internal error.
---------------------	---

<i>MessageEOF-Exception</i>	- if unexpected end of message stream has been reached.
<i>MessageFormat-Exception</i>	- if this type conversion is invalid.
<i>MessageNot-ReadableException</i>	- if the message is in write-only mode.

Implements **cms::StreamMessage** (p. 2613).

6.39.2.17 virtual short **activemq::commands::ActiveMQStreamMessage::readShort**
() const [virtual]

Reads a 16 bit signed short from the Stream message stream.

Returns

short value from stream

Exceptions

<i>CMSEException</i>	- if the CMS provider fails to read the message due to some internal error.
<i>MessageEOF-Exception</i>	- if unexpected end of message stream has been reached.
<i>MessageFormat-Exception</i>	- if this type conversion is invalid.
<i>MessageNot-ReadableException</i>	- if the message is in write-only mode.

Implements **cms::StreamMessage** (p. 2614).

6.39.2.18 virtual std::string **activemq::commands::ActiveMQStreamMessage::readString**
() const [virtual]

Reads an ASCII String from the Stream message stream.

Returns

String from stream

Exceptions

<i>CMSEException</i>	- if the CMS provider fails to read the message due to some internal error.
<i>MessageEOF-Exception</i>	- if unexpected end of message stream has been reached.

<i>MessageFormatException</i>	- if this type conversion is invalid.
<i>MessageNotReadableException</i>	- if the message is in write-only mode.

Implements **cms::StreamMessage** (p. 2614).

6.39.2.19 virtual unsigned short **activemq::commands::ActiveMQStreamMessage::readUnsignedShort** () const
[virtual]

Reads a 16 bit unsigned short from the Stream message stream.

Returns

unsigned short value from stream

Exceptions

<i>CMSException</i>	- if the CMS provider fails to read the message due to some internal error.
<i>MessageEOFException</i>	- if unexpected end of message stream has been reached.
<i>MessageFormatException</i>	- if this type conversion is invalid.
<i>MessageNotReadableException</i>	- if the message is in write-only mode.

Implements **cms::StreamMessage** (p. 2615).

6.39.2.20 virtual void **activemq::commands::ActiveMQStreamMessage::reset** ()
[virtual]

6.39.2.21 virtual std::string **activemq::commands::ActiveMQStreamMessage::toString** () const [virtual]

Returns a string containing the information for this **DataStructure** (p. 1133) such as its type and value of its elements.

Returns

formatted string useful for debugging.

Reimplemented from **activemq::commands::Message** (p. 1836).

6.39.2.22 `virtual void activemq::commands::ActiveMQStreamMessage::writeBoolean (bool value) [virtual]`

Writes a boolean to the Stream message stream as a 1-byte value.

The value true is written as the value (byte)1; the value false is written as the value (byte)0.

Parameters

<i>value</i>	boolean to write to the stream
--------------	--------------------------------

Exceptions

<i>CMSException</i>	- if the CMS provider fails to write the message due to some internal error.
<i>MessageNotWritableException</i>	- if the message is in read-only mode.

Implements **cms::StreamMessage** (p. 2615).

6.39.2.23 `virtual void activemq::commands::ActiveMQStreamMessage::writeByte (unsigned char value) [virtual]`

Writes a byte to the Stream message stream as a 1-byte value.

Parameters

<i>value</i>	byte to write to the stream
--------------	-----------------------------

Exceptions

<i>CMSException</i>	- if the CMS provider fails to write the message due to some internal error.
<i>MessageNotWritableException</i>	- if the message is in read-only mode.

Implements **cms::StreamMessage** (p. 2616).

6.39.2.24 `virtual void activemq::commands::ActiveMQStreamMessage::writeBytes (const std::vector< unsigned char > & value) [virtual]`

Writes a byte array to the Stream message stream using the vector size as the number of bytes to write.

Parameters

<i>value</i>	bytes to write to the stream
--------------	------------------------------

Exceptions

<i>CMSException</i>	- if the CMS provider fails to write the message due to some internal error.
<i>MessageNotWriteableException</i>	- if the message is in read-only mode.

Implements **cms::StreamMessage** (p. 2616).

6.39.2.25 virtual void **activemq::commands::ActiveMQStreamMessage::writeBytes**
(const unsigned char * *value*, int *offset*, int *length*) [virtual]

Writes a portion of a byte array to the Stream message stream.

size as the number of bytes to write.

Parameters

<i>value</i>	bytes to write to the stream
<i>offset</i>	the initial offset within the byte array
<i>length</i>	the number of bytes to use

Exceptions

<i>CMSException</i>	- if the CMS provider fails to write the message due to some internal error.
<i>MessageNotWriteableException</i>	- if the message is in read-only mode.

Implements **cms::StreamMessage** (p. 2616).

6.39.2.26 virtual void **activemq::commands::ActiveMQStreamMessage::writeChar** (
char *value*) [virtual]

Writes a char to the Stream message stream as a 1-byte value.

Parameters

<i>value</i>	char to write to the stream
--------------	-----------------------------

Exceptions

<i>CMSException</i>	- if the CMS provider fails to write the message due to some internal error.
<i>MessageNotWriteableException</i>	- if the message is in read-only mode.

Implements **cms::StreamMessage** (p. 2617).

6.39.2.27 **virtual void activemq::commands::ActiveMQStreamMessage::write-
Double (double *value*)** [virtual]

Writes a double to the Stream message stream as a 8 byte value.

Parameters

<i>value</i>	double to write to the stream
--------------	-------------------------------

Exceptions

<i>CMSEException</i>	- if the CMS provider fails to write the message due to some internal error.
<i>MessageNot-WriteableException</i>	- if the message is in read-only mode.

Implements **cms::StreamMessage** (p. 2617).

6.39.2.28 **virtual void activemq::commands::ActiveMQStreamMessage::writeFloat (float *value*)** [virtual]

Writes a float to the Stream message stream as a 4 byte value.

Parameters

<i>value</i>	float to write to the stream
--------------	------------------------------

Exceptions

<i>CMSEException</i>	- if the CMS provider fails to write the message due to some internal error.
<i>MessageNot-WriteableException</i>	- if the message is in read-only mode.

Implements **cms::StreamMessage** (p. 2618).

6.39.2.29 **virtual void activemq::commands::ActiveMQStreamMessage::writeInt (int *value*)** [virtual]

Writes a signed int to the Stream message stream as a 4 byte value.

Parameters

<i>value</i>	signed int to write to the stream
--------------	-----------------------------------

Exceptions

<i>CMSException</i>	- if the CMS provider fails to write the message due to some internal error.
<i>MessageNot-WriteableException</i>	- if the message is in read-only mode.

Implements **cms::StreamMessage** (p. 2618).

6.39.2.30 **virtual void activemq::commands::ActiveMQStreamMessage::writeLong (long long *value*)** [virtual]

Writes a long long to the Stream message stream as a 8 byte value.

Parameters

<i>value</i>	signed long long to write to the stream
--------------	---

Exceptions

<i>CMSException</i>	- if the CMS provider fails to write the message due to some internal error.
<i>MessageNot-WriteableException</i>	- if the message is in read-only mode.

Implements **cms::StreamMessage** (p. 2619).

6.39.2.31 **virtual void activemq::commands::ActiveMQStreamMessage::writeShort (short *value*)** [virtual]

Writes a signed short to the Stream message stream as a 2 byte value.

Parameters

<i>value</i>	signed short to write to the stream
--------------	-------------------------------------

Exceptions

<i>CMSException</i>	- if the CMS provider fails to write the message due to some internal error.
<i>MessageNot-WriteableException</i>	- if the message is in read-only mode.

Implements **cms::StreamMessage** (p. 2619).

6.39.2.32 `virtual void activemq::commands::ActiveMQStreamMessage::writeString
(const std::string & value) [virtual]`

Writes an ASCII String to the Stream message stream.

Parameters

<i>value</i>	String to write to the stream
--------------	-------------------------------

Exceptions

<i>CMSException</i>	- if the CMS provider fails to write the message due to some internal error.
<i>MessageNot-WriteableException</i>	- if the message is in read-only mode.

Implements **cms::StreamMessage** (p. 2619).

6.39.2.33 `virtual void activemq::commands::ActiveMQStream-
Message::writeUnsignedShort (unsigned short value)
[virtual]`

Writes a unsigned short to the Stream message stream as a 2 byte value.

Parameters

<i>value</i>	unsigned short to write to the stream
--------------	---------------------------------------

Exceptions

<i>CMSException</i>	- if the CMS provider fails to write the message due to some internal error.
<i>MessageNot-WriteableException</i>	- if the message is in read-only mode.

Implements **cms::StreamMessage** (p. 2620).

6.39.3 Field Documentation

6.39.3.1 `const unsigned char activemq::commands::ActiveMQ-
StreamMessage::ID_ACTIVEMQSTREAMMESSAGE = 27
[static]`

The documentation for this class was generated from the following file:

- src/main/activemq/commands/**ActiveMQStreamMessage.h**

6.40 activemq::wireformat::openwire::marshal::generated::ActiveMQStream-MessageMarshaller Class

Reference

375

6.40 activemq::wireformat::openwire::marshal::generated::ActiveMQStreamMessageMarshaller Class Reference

Marshaling code for Open Wire Format for **ActiveMQStreamMessageMarshaller** (p.375).

```
#include <src/main/activemq/wireformat/openwire/marshal/generated/-ActiveMQStreamMessageMarshaller.h>
```

Inheritance diagram for activemq::wireformat::openwire::marshal::generated::ActiveMQStreamMessageMarshaller:

Public Member Functions

- **ActiveMQStreamMessageMarshaller** ()
- virtual ~**ActiveMQStreamMessageMarshaller** ()
- virtual **commands::DataStructure** * **createObject** () const
Creates a new instance of the class that this class is a marshaling director for.
- virtual unsigned char **getDataStructureType** () const
Gets the DataStructureType that this class marshals/unmarshals.
- virtual void **tightUnmarshal** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataInputStream** *dataIn, **utils::BooleanStream** *bs)
Tight Un-marhsal to the given stream.
- virtual int **tightMarshal1** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **utils::BooleanStream** *bs)
Tight Marshal to the given stream.
- virtual void **tightMarshal2** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataOutputStream** *dataOut, **utils::BooleanStream** *bs)
Tight Marshal to the given stream.
- virtual void **looseUnmarshal** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataInputStream** *dataIn)
Loose Un-marhsal to the given stream.
- virtual void **looseMarshal** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataOutputStream** *dataOut)
Tight Marshal to the given stream.

6.40.1 Detailed Description

Marshaling code for Open Wire Format for **ActiveMQStreamMessageMarshaller** (p.375).

NOTE!: This file is auto generated - do not modify! if you need to make a change, please see the Java Classes in the activemq-openwire-generator module

6.40.2 Constructor & Destructor Documentation

6.40.2.1 **activemq::wireformat::openwire::marshal::generated::ActiveMQStreamMessageMarshaller::ActiveMQStreamMessageMarshaller ()**
[inline]

6.40.2.2 **virtual activemq::wireformat::openwire::marshal::generated::ActiveMQStreamMessageMarshaller::~~ActiveMQStreamMessageMarshaller ()**
[inline, virtual]

6.40.3 Member Function Documentation

6.40.3.1 **virtual commands::DataStructure* activemq::wireformat::openwire::marshal::generated::ActiveMQStreamMessageMarshaller::createObject () const** [virtual]

Creates a new instance of the class that this class is a marshaling director for.

Returns

newly allocated Command

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1120).

6.40.3.2 **virtual unsigned char activemq::wireformat::openwire::marshal::generated::ActiveMQStreamMessageMarshaller::getDataStructureType () const**
[virtual]

Gets the DataStructureType that this class marshals/unmarshals.

Returns

byte Id of this classes DataStructureType

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1122).

6.40.3.3 **virtual void activemq::wireformat::openwire::marshal::generated::ActiveMQStreamMessageMarshaller::looseMarshal (OpenWireFormat * format, commands::DataStructure * command, decaf::io::DataOutputStream * ds)** [virtual]

Tight Marshal to the given stream.

Parameters

<i>format</i>	- The OpenWireFormat properties
<i>command</i>	- the object to Marshal
<i>ds</i>	- DataOutputStream to marshal to

6.40 activemq::wireformat::openwire::marshal::generated::ActiveMQStream-MessageMarshaller Class

Reference

377

Exceptions

<i>IOException</i>	if an error occurs.
--------------------	---------------------

Reimplemented from **activemq::wireformat::openwire::marshal::generated::MessageMarshaller** (p. 1919).

6.40.3.4 virtual void **activemq::wireformat::openwire::marshal::generated::ActiveMQStreamMessageMarshaller::looseUnmarshal** (**OpenWireFormat** * *format*, **commands::DataStructure** * *command*, **decaf::io::DataInputStream** * *dis*) [virtual]

Loose Un-marhsal to the given stream.

Parameters

<i>format</i>	- The OpenWireFormat properties
<i>command</i>	- the object to Un-Marshal
<i>dis</i>	- the DataInputStream to Un-Marshal from

Exceptions

<i>IOException</i>	if an error occurs.
--------------------	---------------------

Reimplemented from **activemq::wireformat::openwire::marshal::generated::MessageMarshaller** (p. 1919).

6.40.3.5 virtual int **activemq::wireformat::openwire::marshal::generated::ActiveMQStreamMessageMarshaller::tightMarshal1** (**OpenWireFormat** * *format*, **commands::DataStructure** * *command*, **utils::BooleanStream** * *bs*) [virtual]

Tight Marhsal to the given stream.

Parameters

<i>format</i>	The OpenWireFormat properties
<i>command</i>	The object to Marshal
<i>bs</i>	The boolean stream to marshal to.

Exceptions

<i>IOException</i>	if an error occurs.
--------------------	---------------------

Reimplemented from **activemq::wireformat::openwire::marshal::generated::MessageMarshaller** (p. 1920).

6.40.3.6 virtual void `activemq::wireformat::openwire::marshal::generated::ActiveMQStreamMessageMarshaller::tightMarshal2` (`OpenWireFormat * format`, `commands::DataStructure * command`, `decaf::io::DataOutputStream * ds`, `utils::BooleanStream * bs`) `[virtual]`

Tight Marhsal to the given stream.

Parameters

<i>format</i>	- The OpenwireFormat properties
<i>command</i>	- the object to Marshal
<i>ds</i>	- the DataOutputStream to Marshal to
<i>bs</i>	- boolean stream to marshal to.

Exceptions

<i>IOException</i>	if an error occurs.
--------------------	---------------------

Reimplemented from `activemq::wireformat::openwire::marshal::generated::MessageMarshaller` (p. 1921).

6.40.3.7 virtual void `activemq::wireformat::openwire::marshal::generated::ActiveMQStreamMessageMarshaller::tightUnmarshal` (`OpenWireFormat * format`, `commands::DataStructure * command`, `decaf::io::DataInputStream * dis`, `utils::BooleanStream * bs`) `[virtual]`

Tight Un-marhsal to the given stream.

Parameters

<i>format</i>	- The OpenwireFormat properties
<i>command</i>	- the object to Un-Marshal
<i>dis</i>	- the DataInputStream to Un-Marshal from
<i>bs</i>	- boolean stream to unmarshal from.

Exceptions

<i>IOException</i>	if an error occurs.
--------------------	---------------------

Reimplemented from `activemq::wireformat::openwire::marshal::generated::MessageMarshaller` (p. 1921).

The documentation for this class was generated from the following file:

- `src/main/activemq/wireformat/openwire/marshal/generated/ActiveMQStreamMessageMarshaller.h`

6.41 activemq::commands::ActiveMQTempDestination Class - Reference

```
#include <src/main/activemq/commands/ActiveMQTempDestination.h>
```

Inheritance diagram for activemq::commands::ActiveMQTempDestination:

Public Member Functions

- **ActiveMQTempDestination** ()
- **ActiveMQTempDestination** (const std::string &name)
- virtual ~**ActiveMQTempDestination** () throw ()
- virtual unsigned char **getDataStructureType** () const
*Get the **DataStructure** (p. 1133) Type as defined in CommandTypes.h.*
- virtual **ActiveMQTempDestination** * **cloneDataStructure** () const
Clone this object and return a new instance that the caller now owns, this will be an exact copy of this one.
- virtual void **copyDataStructure** (const **DataStructure** *src)
Copy the contents of the passed object into this objects members, overwriting any existing data.
- virtual std::string **toString** () const
*Returns a string containing the information for this **DataStructure** (p. 1133) such as its type and value of its elements.*
- virtual bool **equals** (const **DataStructure** *value) const
*Compares the **DataStructure** (p. 1133) passed in to this one, and returns if they are equivalent.*
- virtual void **close** ()
Closes this object and deallocates the appropriate resources.
- void **setConnection** (core::ActiveMQConnection *connection)
Sets the Parent Connection that is notified when this destination is destroyed.

Static Public Attributes

- static const unsigned char **ID_ACTIVEMQTEMPDESTINATION** = 0

Protected Attributes

- core::ActiveMQConnection * **connection**
Connection that we call back on close to allow this resource to be cleaned up correctly at this end and at the Broker End.

6.41.1 Constructor & Destructor Documentation

- 6.41.1.1 `activemq::commands::ActiveMQTempDestination::ActiveMQTempDestination ()`
- 6.41.1.2 `activemq::commands::ActiveMQTempDestination::ActiveMQTempDestination (const std::string & name)`
- 6.41.1.3 `virtual activemq::commands::ActiveMQTempDestination::~~ActiveMQTempDestination () throw () [virtual]`

6.41.2 Member Function Documentation

- 6.41.2.1 `virtual ActiveMQTempDestination* activemq::commands::ActiveMQTempDestination::cloneDataStructure () const [inline, virtual]`

Clone this object and return a new instance that the caller now owns, this will be an exact copy of this one.

Returns

new copy of this object.

Reimplemented from `activemq::commands::ActiveMQDestination` (p. 249).

Reimplemented in `activemq::commands::ActiveMQTempQueue` (p. 388), and `activemq::commands::ActiveMQTempTopic` (p. 397).

- 6.41.2.2 `virtual void activemq::commands::ActiveMQTempDestination::close () [virtual]`

Closes this object and deallocates the appropriate resources.

The object is generally no longer usable after calling close.

Exceptions

<i>CMSException</i> - If an error occurs while the resource is being closed.
--

Implements `cms::Closeable` (p. 816).

- 6.41.2.3 `virtual void activemq::commands::ActiveMQTempDestination::copyDataStructure (const DataStructure * src) [inline, virtual]`

Copy the contents of the passed object into this objects members, overwriting any existing data.

Returns

src - Source Object

Reimplemented from **activemq::commands::ActiveMQDestination** (p. 249).

Reimplemented in **activemq::commands::ActiveMQTempQueue** (p. 388), and **activemq::commands::ActiveMQTempTopic** (p. 398).

References **activemq::commands::ActiveMQDestination::copyDataStructure()**.

6.41.2.4 **virtual bool activemq::commands::ActiveMQTempDestination::equals (const DataStructure * value) const** [inline, virtual]

Compares the **DataStructure** (p. 1133) passed in to this one, and returns if they are equivalent.

Equivalent here means that they are of the same type, and that each element of the objects are the same.

Returns

true if **DataStructure** (p. 1133)'s are Equal.

Reimplemented from **activemq::commands::ActiveMQDestination** (p. 250).

Reimplemented in **activemq::commands::ActiveMQTempQueue** (p. 389), and **activemq::commands::ActiveMQTempTopic** (p. 398).

References **activemq::commands::ActiveMQDestination::equals()**.

6.41.2.5 **virtual unsigned char activemq::commands::ActiveMQTempDestination::getDataStructureType () const** [virtual]

Get the **DataStructure** (p. 1133) Type as defined in CommandTypes.h.

Returns

The type of the data structure

Reimplemented from **activemq::commands::ActiveMQDestination** (p. 251).

Reimplemented in **activemq::commands::ActiveMQTempQueue** (p. 390), and **activemq::commands::ActiveMQTempTopic** (p. 399).

6.41.2.6 **void activemq::commands::ActiveMQTempDestination::setConnection (core::ActiveMQConnection * connection)** [inline]

Sets the Parent Connection that is notified when this destination is destroyed.

Parameters

<i>connection</i>	The parent connection to be used to destroy this destination if closed..
-------------------	--

6.41.2.7 `virtual std::string activemq::commands::ActiveMQTempDestination::toString () const` `[virtual]`

Returns a string containing the information for this **DataStructure** (p. 1133) such as its type and value of its elements.

Returns

formatted string useful for debugging.

Reimplemented from **activemq::commands::ActiveMQDestination** (p. 255).

Reimplemented in **activemq::commands::ActiveMQTempQueue** (p. 391), and **activemq::commands::ActiveMQTempTopic** (p. 400).

6.41.3 Field Documentation

6.41.3.1 `core::ActiveMQConnection* activemq::commands::ActiveMQTempDestination::connection` `[protected]`

Connection that we call back on close to allow this resource to be cleaned up correctly at this end and at the Broker End.

6.41.3.2 `const unsigned char activemq::commands::ActiveMQTempDestination::ID_ACTIVEMQTEMPDESTINATION = 0` `[static]`

The documentation for this class was generated from the following file:

- `src/main/activemq/commands/ActiveMQTempDestination.h`

6.42 **activemq::wireformat::openwire::marshal::generated::ActiveMQTempDestinationMarshaller** Class Reference

Marshaling code for Open Wire Format for **ActiveMQTempDestinationMarshaller** (p. 382).

```
#include <src/main/activemq/wireformat/openwire/marshal/generated/ActiveMQTempDestinationMarshaller.h>
```

Inheritance diagram for `activemq::wireformat::openwire::marshal::generated::ActiveMQTempDestinationMarshaller`:

- **ActiveMQTempDestinationMarshaller** ()
- virtual **~ActiveMQTempDestinationMarshaller** ()
- virtual void **tightUnmarshal** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataInputStream** *dataIn, **utils::BooleanStream** *bs)
Tight Un-marhsal to the given stream.
- virtual int **tightMarshal1** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **utils::BooleanStream** *bs)
Tight Marhsal to the given stream.
- virtual void **tightMarshal2** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataOutputStream** *dataOut, **utils::BooleanStream** *bs)
Tight Marhsal to the given stream.
- virtual void **looseUnmarshal** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataInputStream** *dataIn)
Loose Un-marhsal to the given stream.
- virtual void **looseMarshal** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataOutputStream** *dataOut)
Tight Marhsal to the given stream.

6.42.1 Detailed Description

Marshaling code for Open Wire Format for **ActiveMQTempDestinationMarshaller** (p.382).

NOTE!: This file is auto generated - do not modify! if you need to make a change, please see the Java Classes in the activemq-openwire-generator module

6.42.2 Constructor & Destructor Documentation

6.42.2.1 **activemq::wireformat::openwire::marshal::generated::ActiveMQTempDestinationMarshaller::ActiveMQTempDestinationMarshaller** ()
 [inline]

6.42.2.2 **virtual activemq::wireformat::openwire::marshal::generated::ActiveMQTempDestinationMarshaller::~~ActiveMQTempDestinationMarshaller** ()
 [inline, virtual]

6.42.3 Member Function Documentation

6.42.3.1 virtual void `activemq::wireformat::openwire::marshal::generated::ActiveMQTempDestinationMarshaller::looseMarshal (OpenWireFormat * format, commands::DataStructure * command, decaf::io::DataOutputStream * ds)` [virtual]

Tight Marshal to the given stream.

Parameters

<i>format</i>	- The OpenWireFormat properties
<i>command</i>	- the object to Marshal
<i>ds</i>	- DataOutputStream to marshal to

Exceptions

<i>IOException</i>	if an error occurs.
--------------------	---------------------

Reimplemented from `activemq::wireformat::openwire::marshal::generated::ActiveMQDestinationMarshaller` (p. 259).

Reimplemented in `activemq::wireformat::openwire::marshal::generated::ActiveMQTempQueueMarshaller` (p. 393), and `activemq::wireformat::openwire::marshal::generated::ActiveMQTempTopicMarshaller` (p. 403).

6.42.3.2 virtual void `activemq::wireformat::openwire::marshal::generated::ActiveMQTempDestinationMarshaller::looseUnmarshal (OpenWireFormat * format, commands::DataStructure * command, decaf::io::DataInputStream * dis)` [virtual]

Loose Un-marshal to the given stream.

Parameters

<i>format</i>	- The OpenWireFormat properties
<i>command</i>	- the object to Un-Marshal
<i>dis</i>	- the DataInputStream to Un-Marshal from

Exceptions

<i>IOException</i>	if an error occurs.
--------------------	---------------------

Reimplemented from `activemq::wireformat::openwire::marshal::generated::ActiveMQDestinationMarshaller` (p. 259).

Reimplemented in `activemq::wireformat::openwire::marshal::generated::ActiveMQTempQueueMarshaller` (p. 394), and `activemq::wireformat::openwire::marshal::generated::ActiveMQTempTopicMarshaller` (p. 403).

6.42 activemq::wireformat::openwire::marshal::generated::ActiveMQTempDestinationMarshaller Class

Reference

385

6.42.3.3 virtual int activemq::wireformat::openwire::marshal::generated::ActiveMQTempDestinationMarshaller::tightMarshal1 (OpenWireFormat * *format*, commands::DataStructure * *command*, utils::BooleanStream * *bs*)
[virtual]

Tight Marshal to the given stream.

Parameters

<i>format</i>	The OpenWireFormat properties
<i>command</i>	The object to Marshal
<i>bs</i>	The boolean stream to marshal to.

Exceptions

<i>IOException</i>	if an error occurs.
--------------------	---------------------

Reimplemented from **activemq::wireformat::openwire::marshal::generated::ActiveMQDestinationMarshaller** (p. 260).

Reimplemented in **activemq::wireformat::openwire::marshal::generated::ActiveMQTempQueueMarshaller** (p. 394), and **activemq::wireformat::openwire::marshal::generated::ActiveMQTempTopicMarshaller** (p. 403).

6.42.3.4 virtual void activemq::wireformat::openwire::marshal::generated::ActiveMQTempDestinationMarshaller::tightMarshal2 (OpenWireFormat * *format*, commands::DataStructure * *command*, decaf::io::DataOutputStream * *ds*, utils::BooleanStream * *bs*)
[virtual]

Tight Marshal to the given stream.

Parameters

<i>format</i>	- The OpenWireFormat properties
<i>command</i>	- the object to Marshal
<i>ds</i>	- the DataOutputStream to Marshal to
<i>bs</i>	- boolean stream to marshal to.

Exceptions

<i>IOException</i>	if an error occurs.
--------------------	---------------------

Reimplemented from **activemq::wireformat::openwire::marshal::generated::ActiveMQDestinationMarshaller** (p. 260).

Reimplemented in **activemq::wireformat::openwire::marshal::generated::ActiveMQTempQueueMarshaller** (p. 395), and **activemq::wireformat::openwire::marshal::generated::ActiveMQTempTopicMarshaller** (p. 404).

6.42.3.5 virtual void `activemq::wireformat::openwire::marshal::generated::ActiveMQTempDestinationMarshaller::tightUnmarshal (OpenWireFormat * format, commands::DataStructure * command, decaf::io::DataInputStream * dis, utils::BooleanStream * bs)` [virtual]

Tight Un-marhsal to the given stream.

Parameters

<i>format</i>	- The OpenWireFormat properties
<i>command</i>	- the object to Un-Marshal
<i>dis</i>	- the DataInputStream to Un-Marshal from
<i>bs</i>	- boolean stream to unmarshal from.

Exceptions

<i>IOException</i>	if an error occurs.
--------------------	---------------------

Reimplemented from `activemq::wireformat::openwire::marshal::generated::ActiveMQDestinationMarshaller` (p.261).

Reimplemented in `activemq::wireformat::openwire::marshal::generated::ActiveMQTempQueueMarshaller` (p.395), and `activemq::wireformat::openwire::marshal::generated::ActiveMQTempTopicMarshaller` (p.404).

The documentation for this class was generated from the following file:

- `src/main/activemq/wireformat/openwire/marshal/generated/ActiveMQTempDestinationMarshaller.h`

6.43 activemq::commands::ActiveMQTempQueue Class Reference

```
#include <src/main/activemq/commands/ActiveMQTempQueue.h>
```

Inheritance diagram for `activemq::commands::ActiveMQTempQueue`:

Public Member Functions

- `ActiveMQTempQueue ()`
- `ActiveMQTempQueue (const std::string &name)`
- `virtual ~ActiveMQTempQueue () throw ()`
- `virtual unsigned char getDataStructureType () const`
Get the *DataStructure* (p. 1133) Type as defined in *CommandTypes.h*.
- `virtual ActiveMQTempQueue * cloneDataStructure () const`

Clone this object and return a new instance that the caller now owns, this will be an exact copy of this one.

- virtual void **copyDataStructure** (const **DataStructure** *src)
Copy the contents of the passed object into this objects members, overwriting any existing data.
- virtual std::string **toString** () const
*Returns a string containing the information for this **DataStructure** (p. 1133) such as its type and value of its elements.*
- virtual bool **equals** (const **DataStructure** *value) const
*Compares the **DataStructure** (p. 1133) passed in to this one, and returns if they are equivalent.*
- virtual const **cms::Destination** * **getCMSDestination** () const
- virtual **cms::Destination::DestinationType** **getDestinationType** () const
Returns the Type of Destination that this object represents.
- virtual **cms::Destination** * **clone** () const
Creates a new instance of this destination type that is a copy of this one, and returns it.
- virtual void **copy** (const **cms::Destination** &source)
Copies the contents of the given Destination object to this one.
- virtual const **cms::CMSProperties** & **getCMSProperties** () const
Retrieve any properties that might be part of the destination that was specified.
- virtual bool **equals** (const **cms::Destination** &other) const
Compares two Destination instances to determine if they represent the same logic Destination.
- virtual std::string **getQueueName** () const
Gets the name of this queue.
- virtual void **destroy** ()
Destroy's the Temporary Destination at the Provider.

Static Public Attributes

- static const unsigned char **ID_ACTIVEMQTEMPQUEUE** = 102

6.43.1 Constructor & Destructor Documentation

6.43.1.1 **activemq::commands::ActiveMQTempQueue::ActiveMQTempQueue** ()

6.43.1.2 **activemq::commands::ActiveMQTempQueue::ActiveMQTempQueue** (const std::string & name)

6.43.1.3 **virtual activemq::commands::ActiveMQTempQueue::~~ActiveMQTempQueue** () throw () [virtual]

6.43.2 Member Function Documentation

6.43.2.1 `virtual cms::Destination* activemq::commands::-`
`ActiveMQTempQueue::clone () const [inline,`
`virtual]`

Creates a new instance of this destination type that is a copy of this one, and returns it.

Returns

cloned copy of this object

Implements **cms::Destination** (p. 1211).

6.43.2.2 `virtual ActiveMQTempQueue* activemq::commands::-`
`ActiveMQTempQueue::cloneDataStructure () const`
`[virtual]`

Clone this object and return a new instance that the caller now owns, this will be an exact copy of this one.

Returns

new copy of this object.

Reimplemented from **activemq::commands::ActiveMQTempDestination** (p. 380).

6.43.2.3 `virtual void activemq::commands::ActiveMQTempQueue::copy (const`
`cms::Destination & source) [inline, virtual]`

Copies the contents of the given Destination object to this one.

Parameters

<i>source</i>	The source Destination object.
---------------	--------------------------------

Implements **cms::Destination** (p. 1211).

6.43.2.4 `virtual void activemq::commands::ActiveMQTempQueue-`
`::copyDataStructure (const DataStructure * src)`
`[virtual]`

Copy the contents of the passed object into this objects members, overwriting any existing data.

Returns

src - Source Object

Reimplemented from **activemq::commands::ActiveMQTempDestination** (p. 380).

6.43.2.5 `virtual void activemq::commands::ActiveMQTempQueue::destroy ()`
[virtual]

Destroy's the Temporary Destination at the Provider.

Exceptions

<i>CMSException</i>	- if an internal error occurs.
---------------------	--------------------------------

Implements **cms::TemporaryQueue** (p. 2699).

6.43.2.6 `virtual bool activemq::commands::ActiveMQTempQueue::equals (const DataStructure * value) const` [virtual]

Compares the **DataStructure** (p. 1133) passed in to this one, and returns if they are equivalent.

Equivalent here means that they are of the same type, and that each element of the objects are the same.

Returns

true if **DataStructure** (p. 1133)'s are Equal.

Reimplemented from **activemq::commands::ActiveMQTempDestination** (p. 381).

6.43.2.7 `virtual bool activemq::commands::ActiveMQTempQueue::equals (const cms::Destination & other) const` [virtual]

Compares two Destination instances to determine if they represent the same logic - Destination.

Parameters

<i>other</i>	The other destination to compare this one to.
--------------	---

Returns

true if the two destinations are the same.

Implements **cms::Destination** (p. 1212).

6.43.2.8 `virtual const cms::Destination* activemq::commands::ActiveMQTempQueue::getCMSDestination () const` [inline, virtual]

Returns

the **cms::Destination** (p. 1210) interface pointer that the objects that derive from this class implement.

Reimplemented from **activemq::commands::ActiveMQDestination** (p. 251).

6.43.2.9 `virtual const cms::CMSProperties& activemq::commands::ActiveMQTempQueue::getCMSProperties () const [inline, virtual]`

Retrieve any properties that might be part of the destination that was specified.

This is a deviation from the JMS spec but necessary due to C++ restrictions.

Returns

A {const} reference to a CMSProperties object.

Implements **cms::Destination** (p. 1212).

6.43.2.10 `virtual unsigned char activemq::commands::ActiveMQTempQueue::getDataStructureType () const [virtual]`

Get the **DataStructure** (p. 1133) Type as defined in CommandTypes.h.

Returns

The type of the data structure

Reimplemented from **activemq::commands::ActiveMQTempDestination** (p. 381).

6.43.2.11 `virtual cms::Destination::DestinationType activemq::commands::ActiveMQTempQueue::getDestinationType () const [inline, virtual]`

Returns the Type of Destination that this object represents.

Returns

int type qualifier.

Implements **activemq::commands::ActiveMQDestination** (p. 251).

References cms::Destination::TEMPORARY_QUEUE.

6.44 activemq::wireformat::openwire::marshal::generated::ActiveMQTempQueueMarshaller Class

Reference

391

6.43.2.12 `virtual std::string activemq::commands::ActiveMQTempQueue::getQueueName () const [inline, virtual]`

Gets the name of this queue.

Returns

The queue name.

Exceptions

<i>CMSEException</i>	- if an internal error occurs.
----------------------	--------------------------------

Implements **cms::TemporaryQueue** (p. 2699).

6.43.2.13 `virtual std::string activemq::commands::ActiveMQTempQueue::toString () const [virtual]`

Returns a string containing the information for this **DataStructure** (p. 1133) such as its type and value of its elements.

Returns

formatted string useful for debugging.

Reimplemented from **activemq::commands::ActiveMQTempDestination** (p. 382).

6.43.3 Field Documentation

6.43.3.1 `const unsigned char activemq::commands::ActiveMQTempQueue::ID_ACTIVEMQTEMPQUEUE = 102 [static]`

The documentation for this class was generated from the following file:

- src/main/activemq/commands/**ActiveMQTempQueue.h**

6.44 activemq::wireformat::openwire::marshal::generated::ActiveMQTempQueueMarshaller Class Reference

Marshaling code for Open Wire Format for **ActiveMQTempQueueMarshaller** (p. 391).

```
#include <src/main/activemq/wireformat/openwire/marshal/generated/ActiveMQTempQueueMarshaller.h>
```

Inheritance diagram for activemq::wireformat::openwire::marshal::generated::ActiveMQTempQueueMarshaller:

Public Member Functions

- **ActiveMQTempQueueMarshaller** ()
- virtual **~ActiveMQTempQueueMarshaller** ()
- virtual **commands::DataStructure * createObject** () const
Creates a new instance of the class that this class is a marshaling director for.
- virtual unsigned char **getDataStructureType** () const
Gets the DataStructureType that this class marshals/unmarshals.
- virtual void **tightUnmarshal** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataInputStream** *dataIn, **utils::BooleanStream** *bs)
Tight Un-marhsal to the given stream.
- virtual int **tightMarshal1** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **utils::BooleanStream** *bs)
Tight Marhsal to the given stream.
- virtual void **tightMarshal2** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataOutputStream** *dataOut, **utils::BooleanStream** *bs)
Tight Marhsal to the given stream.
- virtual void **looseUnmarshal** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataInputStream** *dataIn)
Loose Un-marhsal to the given stream.
- virtual void **looseMarshal** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataOutputStream** *dataOut)
Tight Marhsal to the given stream.

6.44.1 Detailed Description

Marshaling code for Open Wire Format for **ActiveMQTempQueueMarshaller** (p. 391).

NOTE!: This file is auto generated - do not modify! if you need to make a change, please see the Java Classes in the activemq-openwire-generator module

6.44.2 Constructor & Destructor Documentation

- 6.44.2.1 **activemq::wireformat::openwire::marshal::generated::ActiveMQTempQueueMarshaller::ActiveMQTempQueueMarshaller** ()
[inline]

6.44 activemq::wireformat::openwire::marshal::generated::ActiveMQTempQueueMarshaller Class

Reference

393

6.44.2.2 `virtual activemq::wireformat::openwire::marshal::generated::ActiveMQTempQueueMarshaller::~~ActiveMQTempQueueMarshaller ()`
[inline, virtual]

6.44.3 Member Function Documentation

6.44.3.1 `virtual commands::DataStructure* activemq::wireformat::openwire::marshal::generated::ActiveMQTempQueueMarshaller::createObject ()`
`const` [virtual]

Creates a new instance of the class that this class is a marshaling director for.

Returns

newly allocated Command

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1120).

6.44.3.2 `virtual unsigned char activemq::wireformat::openwire::marshal::generated::ActiveMQTempQueueMarshaller::getDataStructureType () const`
[virtual]

Gets the DataStructureType that this class marshals/unmarshals.

Returns

byte Id of this classes DataStructureType

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1122).

6.44.3.3 `virtual void activemq::wireformat::openwire::marshal::generated::ActiveMQTempQueueMarshaller::looseMarshal (OpenWireFormat * format, commands::DataStructure * command, decaf::io::DataOutputStream * ds)` [virtual]

Tight Marshal to the given stream.

Parameters

<i>format</i>	- The OpenWireFormat properties
<i>command</i>	- the object to Marshal
<i>ds</i>	- DataOutputStream to marshal to

Exceptions

<i>IOException</i>	if an error occurs.
--------------------	---------------------

Reimplemented from **activemq::wireformat::openwire::marshal::generated::ActiveMQTempDestinationMarshaller** (p. 384).

6.44.3.4 `virtual void activemq::wireformat::openwire::marshal::generated::ActiveMQTempQueueMarshaller::looseUnmarshal (OpenWireFormat * format, commands::DataStructure * command, decaf::io::DataInputStream * dis)`
[virtual]

Loose Un-marhsal to the given stream.

Parameters

<i>format</i>	- The OpenWireFormat properties
<i>command</i>	- the object to Un-Marshal
<i>dis</i>	- the DataInputStream to Un-Marshal from

Exceptions

<i>IOException</i>	if an error occurs.
--------------------	---------------------

Reimplemented from **activemq::wireformat::openwire::marshal::generated::ActiveMQTempDestinationMarshaller** (p. 384).

6.44.3.5 `virtual int activemq::wireformat::openwire::marshal::generated::ActiveMQTempQueueMarshaller::tightMarshal1 (OpenWireFormat * format, commands::DataStructure * command, utils::BooleanStream * bs)`
[virtual]

Tight Marhsal to the given stream.

Parameters

<i>format</i>	The OpenWireFormat properties
<i>command</i>	The object to Marshal
<i>bs</i>	The boolean stream to marshal to.

Exceptions

<i>IOException</i>	if an error occurs.
--------------------	---------------------

Reimplemented from **activemq::wireformat::openwire::marshal::generated::ActiveMQTempDestinationMarshaller** (p. 385).

6.44 activemq::wireformat::openwire::marshal::generated::ActiveMQTempQueueMarshaller Class

Reference

395

6.44.3.6 virtual void activemq::wireformat::openwire::marshal::generated::ActiveMQTempQueueMarshaller::tightMarshal2 (OpenWireFormat * *format*, commands::DataStructure * *command*, decaf::io::DataOutputStream * *ds*, utils::BooleanStream * *bs*) [virtual]

Tight Marhsal to the given stream.

Parameters

<i>format</i>	- The OpenWireFormat properties
<i>command</i>	- the object to Marshal
<i>ds</i>	- the DataOutputStream to Marshal to
<i>bs</i>	- boolean stream to marshal to.

Exceptions

<i>IOException</i>	if an error occurs.
--------------------	---------------------

Reimplemented from **activemq::wireformat::openwire::marshal::generated::ActiveMQTempDestinationMarshaller** (p. 385).

6.44.3.7 virtual void activemq::wireformat::openwire::marshal::generated::ActiveMQTempQueueMarshaller::tightUnmarshal (OpenWireFormat * *format*, commands::DataStructure * *command*, decaf::io::DataInputStream * *dis*, utils::BooleanStream * *bs*) [virtual]

Tight Un-marhsal to the given stream.

Parameters

<i>format</i>	- The OpenWireFormat properties
<i>command</i>	- the object to Un-Marshal
<i>dis</i>	- the DataInputStream to Un-Marshal from
<i>bs</i>	- boolean stream to unmarshal from.

Exceptions

<i>IOException</i>	if an error occurs.
--------------------	---------------------

Reimplemented from **activemq::wireformat::openwire::marshal::generated::ActiveMQTempDestinationMarshaller** (p. 386).

The documentation for this class was generated from the following file:

- src/main/activemq/wireformat/openwire/marshal/generated/**ActiveMQTempQueueMarshaller.h**

6.45 activemq::commands::ActiveMQTempTopic Class Reference

```
#include <src/main/activemq/commands/ActiveMQTempTopic.h>
```

Inheritance diagram for activemq::commands::ActiveMQTempTopic:

Public Member Functions

- **ActiveMQTempTopic** ()
- **ActiveMQTempTopic** (const std::string &name)
- virtual ~**ActiveMQTempTopic** () throw ()
- virtual unsigned char **getDataStructureType** () const
*Get the **DataStructure** (p. 1133) Type as defined in CommandTypes.h.*
- virtual **ActiveMQTempTopic** * **cloneDataStructure** () const
Clone this object and return a new instance that the caller now owns, this will be an exact copy of this one.
- virtual void **copyDataStructure** (const **DataStructure** *src)
Copy the contents of the passed object into this objects members, overwriting any existing data.
- virtual std::string **toString** () const
*Returns a string containing the information for this **DataStructure** (p. 1133) such as its type and value of its elements.*
- virtual bool **equals** (const **DataStructure** *value) const
*Compares the **DataStructure** (p. 1133) passed in to this one, and returns if they are equivalent.*
- virtual const cms::Destination * **getCMSDestination** () const
- virtual cms::Destination::DestinationType **getDestinationType** () const
Returns the Type of Destination that this object represents.
- virtual cms::Destination * **clone** () const
Creates a new instance of this destination type that is a copy of this one, and returns it.
- virtual void **copy** (const cms::Destination &source)
Copies the contents of the given Destination object to this one.
- virtual const cms::CMSProperties & **getCMSProperties** () const
Retrieve any properties that might be part of the destination that was specified.
- virtual bool **equals** (const cms::Destination &other) const
Compares two Destination instances to determine if they represent the same logic Destination.
- virtual std::string **getTopicName** () const
Gets the name of this topic.
- virtual void **destroy** ()
Destroy's the Temporary Destination at the Provider.

Static Public Attributes

- static const unsigned char **ID_ACTIVEMQTEMPTOPIC** = 103

6.45.1 Constructor & Destructor Documentation

6.45.1.1 `activemq::commands::ActiveMQTempTopic::ActiveMQTempTopic ()`

6.45.1.2 `activemq::commands::ActiveMQTempTopic::ActiveMQTempTopic (const std::string & name)`

6.45.1.3 `virtual activemq::commands::ActiveMQTempTopic::~~ActiveMQTempTopic () throw () [virtual]`

6.45.2 Member Function Documentation

6.45.2.1 `virtual cms::Destination* activemq::commands::ActiveMQTempTopic::clone () const [inline, virtual]`

Creates a new instance of this destination type that is a copy of this one, and returns it.

Returns

cloned copy of this object

Implements **cms::Destination** (p. 1211).

6.45.2.2 `virtual ActiveMQTempTopic* activemq::commands::ActiveMQTempTopic::cloneDataStructure () const [virtual]`

Clone this object and return a new instance that the caller now owns, this will be an exact copy of this one.

Returns

new copy of this object.

Reimplemented from **activemq::commands::ActiveMQTempDestination** (p. 380).

6.45.2.3 `virtual void activemq::commands::ActiveMQTempTopic::copy (const cms::Destination & source) [inline, virtual]`

Copies the contents of the given Destination object to this one.

Parameters

<i>source</i>	The source Destination object.
---------------	--------------------------------

Implements **cms::Destination** (p. 1211).

6.45.2.4 `virtual void activemq::commands::ActiveMQTempTopic-
::copyDataStructure (const DataStructure * src)
[virtual]`

Copy the contents of the passed object into this objects members, overwriting any existing data.

Returns

src - Source Object

Reimplemented from **activemq::commands::ActiveMQTempDestination** (p. 380).

6.45.2.5 `virtual void activemq::commands::ActiveMQTempTopic::destroy ()
[virtual]`

Destroy's the Temporary Destination at the Provider.

Exceptions

<i>CMSException</i>

Implements **cms::TemporaryTopic** (p. 2701).

6.45.2.6 `virtual bool activemq::commands::ActiveMQTempTopic::equals (const
DataStructure * value) const [virtual]`

Compares the **DataStructure** (p. 1133) passed in to this one, and returns if they are equivalent.

Equivalent here means that they are of the same type, and that each element of the objects are the same.

Returns

true if **DataStructure** (p. 1133)'s are Equal.

Reimplemented from **activemq::commands::ActiveMQTempDestination** (p. 381).

6.45.2.7 `virtual bool activemq::commands::ActiveMQTempTopic::equals (const cms::Destination & other) const` [virtual]

Compares two Destination instances to determine if they represent the same logic - Destination.

Parameters

<i>other</i>	The other destination to compare this one to.
--------------	---

Returns

true if the two destinations are the same.

Implements **cms::Destination** (p. 1212).

6.45.2.8 `virtual const cms::Destination* activemq::commands::ActiveMQTempTopic::getCMSDestination () const` [inline, virtual]

Returns

the **cms::Destination** (p. 1210) interface pointer that the objects that derive from this class implement.

Reimplemented from **activemq::commands::ActiveMQDestination** (p. 251).

6.45.2.9 `virtual const cms::CMSProperties& activemq::commands::ActiveMQTempTopic::getCMSProperties () const` [inline, virtual]

Retrieve any properties that might be part of the destination that was specified.

This is a deviation from the JMS spec but necessary due to C++ restrictions.

Returns

A {const} reference to a CMSProperties object.

Implements **cms::Destination** (p. 1212).

6.45.2.10 `virtual unsigned char activemq::commands::ActiveMQTempTopic::getDataStructureType () const` [virtual]

Get the **DataStructure** (p. 1133) Type as defined in CommandTypes.h.

Returns

The type of the data structure

Reimplemented from **activemq::commands::ActiveMQTempDestination** (p. 381).

6.45.2.11 `virtual cms::Destination::DestinationType activemq::commands::ActiveMQTempTopic::getDestinationType () const [inline, virtual]`

Returns the Type of Destination that this object represents.

Returns

int type qualifier.

Implements **activemq::commands::ActiveMQDestination** (p. 251).

References cms::Destination::TEMPORARY_TOPIC.

6.45.2.12 `virtual std::string activemq::commands::ActiveMQTempTopic::getTopicName () const [inline, virtual]`

Gets the name of this topic.

Returns

The topic name.

Exceptions

<i>CMSEException</i> - if an internal error occurs.

Implements **cms::TemporaryTopic** (p. 2701).

6.45.2.13 `virtual std::string activemq::commands::ActiveMQTempTopic::toString () const [virtual]`

Returns a string containing the information for this **DataStructure** (p. 1133) such as its type and value of its elements.

Returns

formatted string useful for debugging.

Reimplemented from **activemq::commands::ActiveMQTempDestination** (p. 382).

6.45.3 Field Documentation

6.46 activemq::wireformat::openwire::marshal::generated::ActiveMQTempTopicMarshaller Class

Reference

401

```
6.45.3.1 const unsigned char activemq::commands::Active-
MQTempTopic::ID_ACTIVEMQTEMPTOPIC = 103
[static]
```

The documentation for this class was generated from the following file:

- src/main/activemq/commands/**ActiveMQTempTopic.h**

6.46 activemq::wireformat::openwire::marshal::generated::ActiveMQTempTopicMarshaller Class Reference

Marshaling code for Open Wire Format for **ActiveMQTempTopicMarshaller** (p. 401).

```
#include <src/main/activemq/wireformat/openwire/marshal/generated/-
ActiveMQTempTopicMarshaller.h>
```

Inheritance diagram for activemq::wireformat::openwire::marshal::generated::ActiveMQTempTopicMarshaller:

Public Member Functions

- **ActiveMQTempTopicMarshaller** ()
- virtual **~ActiveMQTempTopicMarshaller** ()
- virtual **commands::DataStructure * createObject** () const
Creates a new instance of the class that this class is a marshaling director for.
- virtual unsigned char **getDataStructureType** () const
Gets the DataStructureType that this class marshals/unmarshals.
- virtual void **tightUnmarshal** (**OpenWireFormat *wireFormat**, **commands::DataStructure *dataStructure**, **decaf::io::DataInputStream *dataIn**, **utils::BooleanStream *bs**)
Tight Un-marhsal to the given stream.
- virtual int **tightMarshal1** (**OpenWireFormat *wireFormat**, **commands::DataStructure *dataStructure**, **utils::BooleanStream *bs**)
Tight Marhsal to the given stream.
- virtual void **tightMarshal2** (**OpenWireFormat *wireFormat**, **commands::DataStructure *dataStructure**, **decaf::io::DataOutputStream *dataOut**, **utils::BooleanStream *bs**)
Tight Marhsal to the given stream.
- virtual void **looseUnmarshal** (**OpenWireFormat *wireFormat**, **commands::DataStructure *dataStructure**, **decaf::io::DataInputStream *dataIn**)
Loose Un-marhsal to the given stream.
- virtual void **looseMarshal** (**OpenWireFormat *wireFormat**, **commands::DataStructure *dataStructure**, **decaf::io::DataOutputStream *dataOut**)
Tight Marhsal to the given stream.

6.46.1 Detailed Description

Marshaling code for Open Wire Format for **ActiveMQTempTopicMarshaller** (p. 401).

NOTE!: This file is auto generated - do not modify! if you need to make a change, please see the Java Classes in the activemq-openwire-generator module

6.46.2 Constructor & Destructor Documentation

6.46.2.1 **activemq::wireformat::openwire::marshal::generated::ActiveMQTempTopicMarshaller::ActiveMQTempTopicMarshaller ()**
[inline]

6.46.2.2 **virtual activemq::wireformat::openwire::marshal::generated::ActiveMQTempTopicMarshaller::~~ActiveMQTempTopicMarshaller ()**
[inline, virtual]

6.46.3 Member Function Documentation

6.46.3.1 **virtual commands::DataStructure* activemq::wireformat::openwire::marshal::generated::ActiveMQTempTopicMarshaller::createObject ()**
const [virtual]

Creates a new instance of the class that this class is a marshaling director for.

Returns

newly allocated Command

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1120).

6.46.3.2 **virtual unsigned char activemq::wireformat::openwire::marshal::generated::ActiveMQTempTopicMarshaller::getDataStructureType ()** **const**
[virtual]

Gets the DataStructureType that this class marshals/unmarshals.

Returns

byte Id of this classes DataStructureType

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1122).

6.46 activemq::wireformat::openwire::marshal::generated::ActiveMQTempTopicMarshaller Class

Reference

403

6.46.3.3 virtual void activemq::wireformat::openwire::marshal::generated::ActiveMQTempTopicMarshaller::looseMarshal (OpenWireFormat * *format*, commands::DataStructure * *command*, decaf::io::DataOutputStream * *ds*) [virtual]

Tight Marshal to the given stream.

Parameters

<i>format</i>	- The OpenWireFormat properties
<i>command</i>	- the object to Marshal
<i>ds</i>	- DataOutputStream to marshal to

Exceptions

<i>IOException</i>	if an error occurs.
--------------------	---------------------

Reimplemented from **activemq::wireformat::openwire::marshal::generated::ActiveMQTempDestinationMarshaller** (p. 384).

6.46.3.4 virtual void activemq::wireformat::openwire::marshal::generated::ActiveMQTempTopicMarshaller::looseUnmarshal (OpenWireFormat * *format*, commands::DataStructure * *command*, decaf::io::DataInputStream * *dis*) [virtual]

Loose Un-marshal to the given stream.

Parameters

<i>format</i>	- The OpenWireFormat properties
<i>command</i>	- the object to Un-Marshal
<i>dis</i>	- the DataInputStream to Un-Marshal from

Exceptions

<i>IOException</i>	if an error occurs.
--------------------	---------------------

Reimplemented from **activemq::wireformat::openwire::marshal::generated::ActiveMQTempDestinationMarshaller** (p. 384).

6.46.3.5 virtual int activemq::wireformat::openwire::marshal::generated::ActiveMQTempTopicMarshaller::tightMarshal1 (OpenWireFormat * *format*, commands::DataStructure * *command*, utils::BooleanStream * *bs*) [virtual]

Tight Marshal to the given stream.

Parameters

<i>format</i>	The OpenwireFormat properties
<i>command</i>	The object to Marshal
<i>bs</i>	The boolean stream to marshal to.

Exceptions

<i>IOException</i>	if an error occurs.
--------------------	---------------------

Reimplemented from **activemq::wireformat::openwire::marshal::generated::ActiveMQTempDestinationMarshaller** (p.385).

6.46.3.6 virtual void activemq::wireformat::openwire::marshal::generated::ActiveMQTempTopicMarshaller::tightMarshal2 (OpenWireFormat * *format*, commands::DataStructure * *command*, decaf::io::DataOutputStream * *ds*, utils::BooleanStream * *bs*) [virtual]

Tight Marshal to the given stream.

Parameters

<i>format</i>	- The OpenwireFormat properties
<i>command</i>	- the object to Marshal
<i>ds</i>	- the DataOutputStream to Marshal to
<i>bs</i>	- boolean stream to marshal to.

Exceptions

<i>IOException</i>	if an error occurs.
--------------------	---------------------

Reimplemented from **activemq::wireformat::openwire::marshal::generated::ActiveMQTempDestinationMarshaller** (p.385).

6.46.3.7 virtual void activemq::wireformat::openwire::marshal::generated::ActiveMQTempTopicMarshaller::tightUnmarshal (OpenWireFormat * *format*, commands::DataStructure * *command*, decaf::io::DataInputStream * *dis*, utils::BooleanStream * *bs*) [virtual]

Tight Un-marshal to the given stream.

Parameters

<i>format</i>	- The OpenwireFormat properties
<i>command</i>	- the object to Un-Marshal
<i>dis</i>	- the DataInputStream to Un-Marshal from
<i>bs</i>	- boolean stream to unmarshal from.

Exceptions

<i>IOException</i>	if an error occurs.
--------------------	---------------------

Reimplemented from **activemq::wireformat::openwire::marshal::generated::ActiveMQTempDestinationMarshaller** (p. 386).

The documentation for this class was generated from the following file:

- src/main/activemq/wireformat/openwire/marshal/generated/**ActiveMQTempTopicMarshaller.h**

6.47 activemq::commands::ActiveMQTextMessage Class Reference

```
#include <src/main/activemq/commands/ActiveMQTextMessage.h>
```

Inheritance diagram for activemq::commands::ActiveMQTextMessage:

Public Member Functions

- **ActiveMQTextMessage** ()
- virtual **~ActiveMQTextMessage** () throw ()
- virtual unsigned char **getDataStructureType** () const
*Get the **DataSetructure** (p. 1133) Type as defined in CommandTypes.h.*
- virtual **ActiveMQTextMessage** * **cloneDataSetructure** () const
Clone this object and return a new instance that the caller now owns, this will be an exact copy of this one.
- virtual void **copyDataSetructure** (const **DataSetructure** *src)
Copy the contents of the passed object into this objects members, overwriting any existing data.
- virtual std::string **toString** () const
*Returns a string containing the information for this **DataSetructure** (p. 1133) such as its type and value of its elements.*
- virtual bool **equals** (const **DataSetructure** *value) const
*Compares the **DataSetructure** (p. 1133) passed in to this one, and returns if they are equivalent.*
- virtual void **clearBody** ()
Clears out the body of the message.
- virtual void **beforeMarshal** (**wireformat::WireFormat** *wireFormat)
Called before marshaling is started to prepare the object to be marshaled.
- virtual unsigned int **getSize** () const
Returns the Size of this message in Bytes.
- virtual **cms::TextMessage** * **clone** () const

Clone this message exactly, returns a new instance that the caller is required to delete.

- virtual std::string **getText** () const
Gets the message character buffer.
- virtual void **setText** (const char *msg)
Sets the message contents, does not take ownership of the passed char, but copies it instead.*
- virtual void **setText** (const std::string &msg)
Sets the message contents.

Data Fields

- std::auto_ptr< std::string > **text**

Static Public Attributes

- static const unsigned char **ID_ACTIVEMQTEXTMESSAGE** = 28

6.47.1 Constructor & Destructor Documentation

6.47.1.1 **activemq::commands::ActiveMQTextMessage::ActiveMQTextMessage** ()

6.47.1.2 **virtual activemq::commands::ActiveMQTextMessage::~~ActiveMQTextMessage** () throw () [virtual]

6.47.2 Member Function Documentation

6.47.2.1 **virtual void activemq::commands::ActiveMQTextMessage::beforeMarshal** (**wireformat::WireFormat * wireFormat**) [virtual]

Called before marshaling is started to prepare the object to be marshaled.

Parameters

<i>wireFormat</i>	The wireformat object to control marshaling
-------------------	---

Exceptions

<i>IOException</i>	if an I/O error occurs.
--------------------	-------------------------

Implements **activemq::wireformat::MarshalAware** (p. 1794).

6.47.2.2 **virtual void activemq::commands::ActiveMQTextMessage::clearBody** () [virtual]

Clears out the body of the message.

This does not clear the headers or properties.

Exceptions

<i>CMSException</i>	- if an internal error occurs.
---------------------	--------------------------------

Reimplemented from **activemq::commands::ActiveMQMessageTemplate**< **cms::TextMessage** > (p. 296).

6.47.2.3 `virtual cms::TextMessage* activemq::commands::ActiveMQTextMessage::clone () const [inline, virtual]`

Clone this message exactly, returns a new instance that the caller is required to delete.

Returns

new copy of this message

Implements **cms::Message** (p. 1844).

6.47.2.4 `virtual ActiveMQTextMessage* activemq::commands::ActiveMQTextMessage::cloneDataStructure () const [virtual]`

Clone this obbject and return a new instance that the caller now owns, this will be an exact copy of this one.

Returns

new copy of this object.

Reimplemented from **activemq::commands::Message** (p. 1826).

6.47.2.5 `virtual void activemq::commands::ActiveMQTextMessage::copyDataStructure (const DataStructure * src) [virtual]`

Copy the contents of the passed object into this objects members, overwriting any existing data.

Returns

src - Source Object

Reimplemented from **activemq::commands::Message** (p. 1827).

6.47.2.6 `virtual bool activemq::commands::ActiveMQTextMessage::equals (const
DataStructure * value) const [virtual]`

Compares the **DataStructure** (p. 1133) passed in to this one, and returns if they are equivalent.

Equivalent here means that they are of the same type, and that each element of the objects are the same.

Returns

true if **DataStructure** (p. 1133)'s are Equal.

Reimplemented from **activemq::commands::ActiveMQMessageTemplate**< **cms::TextMessage** > (p. 297).

6.47.2.7 `virtual unsigned char activemq::commands::Active-
MQTextMessage::getDataStructureType () const
[virtual]`

Get the **DataStructure** (p. 1133) Type as defined in CommandTypes.h.

Returns

The type of the data structure

Reimplemented from **activemq::commands::Message** (p. 1829).

6.47.2.8 `virtual unsigned int activemq::commands::ActiveMQTextMessage::getSize () const [virtual]`

Returns the Size of this message in Bytes.

Returns

number of bytes this message equates to.

Reimplemented from **activemq::commands::Message** (p. 1831).

6.47.2.9 `virtual std::string activemq::commands::ActiveMQTextMessage::getText () const [virtual]`

Gets the message character buffer.

Returns

The message character buffer.

Exceptions

<i>CMSException</i>	- if an internal error occurs.
---------------------	--------------------------------

Implements **cms::TextMessage** (p. 2702).

6.47.2.10 virtual void **activemq::commands::ActiveMQTextMessage::setText** (const char * *msg*) [virtual]

Sets the message contents, does not take ownership of the passed char*, but copies it instead.

Parameters

<i>msg</i>	The message buffer.
------------	---------------------

Exceptions

<i>CMSException</i>	- if an internal error occurs.
<i>MessageNotWriteableException</i>	- if the message is in read-only mode..

Implements **cms::TextMessage** (p. 2703).

6.47.2.11 virtual void **activemq::commands::ActiveMQTextMessage::setText** (const std::string & *msg*) [virtual]

Sets the message contents.

Parameters

<i>msg</i>	The message buffer.
------------	---------------------

Exceptions

<i>CMSException</i>	- if an internal error occurs.
<i>MessageNotWriteableException</i>	- if the message is in read-only mode..

Implements **cms::TextMessage** (p. 2703).

6.47.2.12 virtual std::string **activemq::commands::ActiveMQTextMessage::toString** () const [virtual]

Returns a string containing the information for this **DataStructure** (p. 1133) such as its type and value of its elements.

Returns

formatted string useful for debugging.

Reimplemented from **activemq::commands::Message** (p. 1836).

6.47.3 Field Documentation

6.47.3.1 `const unsigned char activemq::commands::ActiveMQTextMessage::ID_ACTIVEMQTEXTMESSAGE = 28`
[static]

6.47.3.2 `std::auto_ptr<std::string> activemq::commands::ActiveMQTextMessage::text` [mutable]

The documentation for this class was generated from the following file:

- `src/main/activemq/commands/ActiveMQTextMessage.h`

6.48 activemq::wireformat::openwire::marshal::generated::ActiveMQTextMessageMarshaller Class Reference

Marshaling code for Open Wire Format for **ActiveMQTextMessageMarshaller** (p. 410).

```
#include <src/main/activemq/wireformat/openwire/marshal/generated/ActiveMQTextMessageMarshaller.h>
```

Inheritance diagram for `activemq::wireformat::openwire::marshal::generated::ActiveMQTextMessageMarshaller`:

Public Member Functions

- **ActiveMQTextMessageMarshaller** ()
- virtual **~ActiveMQTextMessageMarshaller** ()
- virtual **commands::DataStructure * createObject** () const
Creates a new instance of the class that this class is a marshaling director for.
- virtual unsigned char **getDataStructureType** () const
Gets the DataStructureType that this class marshals/unmarshals.
- virtual void **tightUnmarshal** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataInputStream** *dataIn, **utils::BooleanStream** *bs)
Tight Un-marhsal to the given stream.
- virtual int **tightMarshal1** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **utils::BooleanStream** *bs)

6.48 activemq::wireformat::openwire::marshal::generated::ActiveMQTextMessageMarshaller Class

Reference

411

Tight Marshal to the given stream.

- virtual void **tightMarshal2** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataOutputStream** *dataOut, **utils::BooleanStream** *bs)

Tight Marshal to the given stream.

- virtual void **looseUnmarshal** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataInputStream** *dataIn)

Loose Un-marshal to the given stream.

- virtual void **looseMarshal** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataOutputStream** *dataOut)

Tight Marshal to the given stream.

6.48.1 Detailed Description

Marshaling code for Open Wire Format for **ActiveMQTextMessageMarshaller** (p. 410).

NOTE!: This file is auto generated - do not modify! if you need to make a change, please see the Java Classes in the activemq-openwire-generator module

6.48.2 Constructor & Destructor Documentation

6.48.2.1 **activemq::wireformat::openwire::marshal::generated::ActiveMQTextMessageMarshaller::ActiveMQTextMessageMarshaller** ()
[inline]

6.48.2.2 **virtual activemq::wireformat::openwire::marshal::generated::ActiveMQTextMessageMarshaller::~~ActiveMQTextMessageMarshaller** ()
[inline, virtual]

6.48.3 Member Function Documentation

6.48.3.1 **virtual commands::DataStructure* activemq::wireformat::openwire::marshal::generated::ActiveMQTextMessageMarshaller::createObject** () const [virtual]

Creates a new instance of the class that this class is a marshaling director for.

Returns

newly allocated Command

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1120).

6.48.3.2 virtual unsigned char **activemq::wireformat::openwire::marshal::generated::ActiveMQTextMessageMarshaller::getDataStructureType** () const
[virtual]

Gets the DataStructureType that this class marshals/unmarshals.

Returns

byte Id of this classes DataStructureType

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1122).

6.48.3.3 virtual void **activemq::wireformat::openwire::marshal::generated::ActiveMQTextMessageMarshaller::looseMarshal** (OpenWireFormat * *format*, commands::DataStructure * *command*, decaf::io::DataOutputStream * *ds*) [virtual]

Tight Marshal to the given stream.

Parameters

<i>format</i>	- The OpenWireFormat properties
<i>command</i>	- the object to Marshal
<i>ds</i>	- DataOutputStream to marshal to

Exceptions

<i>IOException</i>	if an error occurs.
--------------------	---------------------

Reimplemented from **activemq::wireformat::openwire::marshal::generated::MessageMarshaller** (p. 1919).

6.48.3.4 virtual void **activemq::wireformat::openwire::marshal::generated::ActiveMQTextMessageMarshaller::looseUnmarshal** (OpenWireFormat * *format*, commands::DataStructure * *command*, decaf::io::DataInputStream * *dis*) [virtual]

Loose Un-marshal to the given stream.

Parameters

<i>format</i>	- The OpenWireFormat properties
<i>command</i>	- the object to Un-Marshal
<i>dis</i>	- the DataInputStream to Un-Marshal from

6.48 activemq::wireformat::openwire::marshal::generated::ActiveMQTextMessageMarshaller Class

Reference

413

Exceptions

<i>IOException</i>	if an error occurs.
--------------------	---------------------

Reimplemented from **activemq::wireformat::openwire::marshal::generated::MessageMarshaller** (p. 1919).

6.48.3.5 virtual int **activemq::wireformat::openwire::marshal::generated::ActiveMQTextMessageMarshaller::tightMarshal1** (**OpenWireFormat** * *format*, **commands::DataStructure** * *command*, **utils::BooleanStream** * *bs*)
[virtual]

Tight Marshal to the given stream.

Parameters

<i>format</i>	The OpenWireFormat properties
<i>command</i>	The object to Marshal
<i>bs</i>	The boolean stream to marshal to.

Exceptions

<i>IOException</i>	if an error occurs.
--------------------	---------------------

Reimplemented from **activemq::wireformat::openwire::marshal::generated::MessageMarshaller** (p. 1920).

6.48.3.6 virtual void **activemq::wireformat::openwire::marshal::generated::ActiveMQTextMessageMarshaller::tightMarshal2** (**OpenWireFormat** * *format*, **commands::DataStructure** * *command*, **decaf::io::DataOutputStream** * *ds*, **utils::BooleanStream** * *bs*) [virtual]

Tight Marshal to the given stream.

Parameters

<i>format</i>	- The OpenWireFormat properties
<i>command</i>	- the object to Marshal
<i>ds</i>	- the DataOutputStream to Marshal to
<i>bs</i>	- boolean stream to marshal to.

Exceptions

<i>IOException</i>	if an error occurs.
--------------------	---------------------

Reimplemented from **activemq::wireformat::openwire::marshal::generated::MessageMarshaller** (p. 1921).

6.48.3.7 virtual void `activemq::wireformat::openwire::marshal::generated::ActiveMQTextMessageMarshaller::tightUnmarshal (OpenWireFormat * format, commands::DataStructure * command, decaf::io::DataInputStream * dis, utils::BooleanStream * bs)` [virtual]

Tight Un-marhsal to the given stream.

Parameters

<i>format</i>	- The OpenwireFormat properties
<i>command</i>	- the object to Un-Marshal
<i>dis</i>	- the DataInputStream to Un-Marshal from
<i>bs</i>	- boolean stream to unmarshal from.

Exceptions

<i>IOException</i>	if an error occurs.
--------------------	---------------------

Reimplemented from `activemq::wireformat::openwire::marshal::generated::MessageMarshaller` (p. 1921).

The documentation for this class was generated from the following file:

- `src/main/activemq/wireformat/openwire/marshal/generated/ActiveMQTextMessageMarshaller.h`

6.49 activemq::commands::ActiveMQTopic Class Reference

```
#include <src/main/activemq/commands/ActiveMQTopic.h>
```

Inheritance diagram for `activemq::commands::ActiveMQTopic`:

Public Member Functions

- **ActiveMQTopic** ()
- **ActiveMQTopic** (const std::string &name)
- virtual ~**ActiveMQTopic** () throw ()
- virtual unsigned char **getDataStructureType** () const
*Get the **DataStructure** (p. 1133) Type as defined in CommandTypes.h.*
- virtual **ActiveMQTopic** * **cloneDataStructure** () const
Clone this obbject and return a new instance that the caller now owns, this will be an exact copy of this one.
- virtual void **copyDataStructure** (const **DataStructure** *src)
Copy the contents of the passed object into this objects members, overwriting any existing data.

- virtual std::string **toString** () const
*Returns a string containing the information for this **DataStructure** (p. 1133) such as its type and value of its elements.*
- virtual bool **equals** (const **DataStructure** *value) const
*Compares the **DataStructure** (p. 1133) passed in to this one, and returns if they are equivalent.*
- virtual const cms::Destination * **getCMSDestination** () const
- virtual cms::Destination::DestinationType **getDestinationType** () const
Returns the Type of Destination that this object represents.
- virtual cms::Destination * **clone** () const
Creates a new instance of this destination type that is a copy of this one, and returns it.
- virtual void **copy** (const cms::Destination &source)
Copies the contents of the given Destination object to this one.
- virtual const cms::CMSProperties & **getCMSProperties** () const
Retrieve any properties that might be part of the destination that was specified.
- virtual bool **equals** (const cms::Destination &other) const
Compares two Destination instances to determine if they represent the same logic Destination.
- virtual std::string **getTopicName** () const
Gets the name of this topic.

Static Public Attributes

- static const unsigned char **ID_ACTIVEMQTOPIC** = 101

6.49.1 Constructor & Destructor Documentation

6.49.1.1 **activemq::commands::ActiveMQTopic::ActiveMQTopic** ()

6.49.1.2 **activemq::commands::ActiveMQTopic::ActiveMQTopic** (const std::string & name)

6.49.1.3 **virtual activemq::commands::ActiveMQTopic::~~ActiveMQTopic** () throw
() [virtual]

6.49.2 Member Function Documentation

6.49.2.1 **virtual cms::Destination* activemq::commands::ActiveMQTopic::clone** () const [inline, virtual]

Creates a new instance of this destination type that is a copy of this one, and returns it.

Returns

cloned copy of this object

Implements **cms::Destination** (p. 1211).

6.49.2.2 `virtual ActiveMQTopic* activemq::commands::ActiveMQTopic::clone-
DataStructure () const [virtual]`

Clone this object and return a new instance that the caller now owns, this will be an exact copy of this one.

Returns

new copy of this object.

Reimplemented from `activemq::commands::ActiveMQDestination` (p. 249).

6.49.2.3 `virtual void activemq::commands::ActiveMQTopic::copy (const
cms::Destination & source) [inline, virtual]`

Copies the contents of the given Destination object to this one.

Parameters

<i>source</i>	The source Destination object.
---------------	--------------------------------

Implements `cms::Destination` (p. 1211).

6.49.2.4 `virtual void activemq::commands::ActiveMQTopic::copyDataStructure (const DataStructure * src) [virtual]`

Copy the contents of the passed object into this objects members, overwriting any existing data.

Returns

src - Source Object

Reimplemented from `activemq::commands::ActiveMQDestination` (p. 249).

6.49.2.5 `virtual bool activemq::commands::ActiveMQTopic::equals (const
DataStructure * value) const [virtual]`

Compares the `DataStructure` (p. 1133) passed in to this one, and returns if they are equivalent.

Equivalent here means that they are of the same type, and that each element of the objects are the same.

Returns

true if `DataStructure` (p. 1133)'s are Equal.

Reimplemented from `activemq::commands::ActiveMQDestination` (p. 250).

6.49.2.6 `virtual bool activemq::commands::ActiveMQTopic::equals (const cms::Destination & other) const` [virtual]

Compares two Destination instances to determine if they represent the same logic - Destination.

Parameters

<i>other</i>	The other destination to compare this one to.
--------------	---

Returns

true if the two destinations are the same.

Implements **cms::Destination** (p. 1212).

6.49.2.7 `virtual const cms::Destination* activemq::commands::ActiveMQTopic::getCMSDestination () const` [inline, virtual]

Returns

the **cms::Destination** (p. 1210) interface pointer that the objects that derive from this class implement.

Reimplemented from **activemq::commands::ActiveMQDestination** (p. 251).

6.49.2.8 `virtual const cms::CMSProperties& activemq::commands::ActiveMQTopic::getCMSProperties () const` [inline, virtual]

Retrieve any properties that might be part of the destination that was specified.

This is a deviation from the JMS spec but necessary due to C++ restrictions.

Returns

A {const} reference to a CMSProperties object.

Implements **cms::Destination** (p. 1212).

6.49.2.9 `virtual unsigned char activemq::commands::ActiveMQTopic::getDataStructureType () const` [virtual]

Get the **DataStructure** (p. 1133) Type as defined in CommandTypes.h.

Returns

The type of the data structure

Reimplemented from **activemq::commands::ActiveMQDestination** (p. 251).

6.49.2.10 `virtual cms::Destination::DestinationType activemq::commands-
::ActiveMQTopic::getDestinationType () const [inline,
virtual]`

Returns the Type of Destination that this object represents.

Returns

int type qualifier.

Implements **activemq::commands::ActiveMQDestination** (p. 251).

References cms::Destination::TOPIC.

6.49.2.11 `virtual std::string activemq::commands::ActiveMQTopic::getTopicName () const [inline, virtual]`

Gets the name of this topic.

Returns

The topic name.

Exceptions

<i>CMSException</i> - If an internal error occurs.
--

Implements **cms::Topic** (p. 2765).

6.49.2.12 `virtual std::string activemq::commands::ActiveMQTopic::toString () const [virtual]`

Returns a string containing the information for this **DataStructure** (p. 1133) such as its type and value of its elements.

Returns

formatted string useful for debugging.

Reimplemented from **activemq::commands::ActiveMQDestination** (p. 255).

6.49.3 Field Documentation

6.49.3.1 `const unsigned char activemq::commands::ActiveMQTopic::ID_ACTIVEMQ-
TOPIC = 101 [static]`

The documentation for this class was generated from the following file:

6.50 activemq::wireformat::openwire::marshal::generated::ActiveMQTopicMarshaller Class Reference

Marshaling code for Open Wire Format for **ActiveMQTopicMarshaller** (p. 419).

```
#include <src/main/activemq/wireformat/openwire/marshal/generated/-  
ActiveMQTopicMarshaller.h>
```

Inheritance diagram for activemq::wireformat::openwire::marshal::generated::ActiveMQTopicMarshaller:

Public Member Functions

- **ActiveMQTopicMarshaller** ()
- virtual **~ActiveMQTopicMarshaller** ()
- virtual **commands::DataStructure * createObject** () const
Creates a new instance of the class that this class is a marshaling director for.
- virtual unsigned char **getDataStructureType** () const
Gets the DataStructureType that this class marshals/unmarshals.
- virtual void **tightUnmarshal** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataInputStream** *dataIn, **utils::BooleanStream** *bs)
Tight Un-marhsal to the given stream.
- virtual int **tightMarshal1** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **utils::BooleanStream** *bs)
Tight Marhsal to the given stream.
- virtual void **tightMarshal2** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataOutputStream** *dataOut, **utils::BooleanStream** *bs)
Tight Marhsal to the given stream.
- virtual void **looseUnmarshal** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataInputStream** *dataIn)
Loose Un-marhsal to the given stream.
- virtual void **looseMarshal** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataOutputStream** *dataOut)
Tight Marhsal to the given stream.

6.50.1 Detailed Description

Marshaling code for Open Wire Format for **ActiveMQTopicMarshaller** (p. 419).

NOTE!: This file is auto generated - do not modify! if you need to make a change, please see the Java Classes in the activemq-openwire-generator module

6.50.2 Constructor & Destructor Documentation

6.50.2.1 **activemq::wireformat::openwire::marshal::generated::ActiveMQTopicMarshaller::ActiveMQTopicMarshaller ()**
[inline]

6.50.2.2 **virtual activemq::wireformat::openwire::marshal::generated::ActiveMQTopicMarshaller::~~ActiveMQTopicMarshaller ()** [inline, virtual]

6.50.3 Member Function Documentation

6.50.3.1 **virtual commands::DataStructure* activemq::wireformat::openwire::marshal::generated::ActiveMQTopicMarshaller::createObject ()** const
[virtual]

Creates a new instance of the class that this class is a marshaling director for.

Returns

newly allocated Command

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1120).

6.50.3.2 **virtual unsigned char activemq::wireformat::openwire::marshal::generated::ActiveMQTopicMarshaller::getDataStructureType ()** const
[virtual]

Gets the DataStructureType that this class marshals/unmarshals.

Returns

byte Id of this classes DataStructureType

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1122).

6.50

activemq::wireformat::openwire::marshal::generated::ActiveMQTopicMarshaller Class Reference 421

6.50.3.3 virtual void activemq::wireformat::openwire::marshal::generated::ActiveMQTopicMarshaller::looseMarshal (OpenWireFormat * *format*, commands::DataStructure * *command*, decaf::io::DataOutputStream * *ds*) [virtual]

Tight Marshal to the given stream.

Parameters

<i>format</i>	- The OpenWireFormat properties
<i>command</i>	- the object to Marshal
<i>ds</i>	- DataOutputStream to marshal to

Exceptions

<i>IOException</i>	if an error occurs.
--------------------	---------------------

Reimplemented from **activemq::wireformat::openwire::marshal::generated::ActiveMQDestinationMarshaller** (p. 259).

6.50.3.4 virtual void activemq::wireformat::openwire::marshal::generated::ActiveMQTopicMarshaller::looseUnmarshal (OpenWireFormat * *format*, commands::DataStructure * *command*, decaf::io::DataInputStream * *dis*) [virtual]

Loose Un-marshal to the given stream.

Parameters

<i>format</i>	- The OpenWireFormat properties
<i>command</i>	- the object to Un-Marshal
<i>dis</i>	- the DataInputStream to Un-Marshal from

Exceptions

<i>IOException</i>	if an error occurs.
--------------------	---------------------

Reimplemented from **activemq::wireformat::openwire::marshal::generated::ActiveMQDestinationMarshaller** (p. 259).

6.50.3.5 virtual int activemq::wireformat::openwire::marshal::generated::ActiveMQTopicMarshaller::tightMarshal1 (OpenWireFormat * *format*, commands::DataStructure * *command*, utils::BooleanStream * *bs*) [virtual]

Tight Marshal to the given stream.

Parameters

<i>format</i>	The OpenwireFormat properties
<i>command</i>	The object to Marshal
<i>bs</i>	The boolean stream to marshal to.

Exceptions

<i>IOException</i>	if an error occurs.
--------------------	---------------------

Reimplemented from **activemq::wireformat::openwire::marshal::generated::ActiveMQDestinationMarshaller** (p. 260).

6.50.3.6 virtual void activemq::wireformat::openwire::marshal::generated::ActiveMQTopicMarshaller::tightMarshal2 (OpenWireFormat * *format*, commands::DataStructure * *command*, decaf::io::DataOutputStream * *ds*, utils::BooleanStream * *bs*) [virtual]

Tight Marshal to the given stream.

Parameters

<i>format</i>	- The OpenwireFormat properties
<i>command</i>	- the object to Marshal
<i>ds</i>	- the DataOutputStream to Marshal to
<i>bs</i>	- boolean stream to marshal to.

Exceptions

<i>IOException</i>	if an error occurs.
--------------------	---------------------

Reimplemented from **activemq::wireformat::openwire::marshal::generated::ActiveMQDestinationMarshaller** (p. 260).

6.50.3.7 virtual void activemq::wireformat::openwire::marshal::generated::ActiveMQTopicMarshaller::tightUnmarshal (OpenWireFormat * *format*, commands::DataStructure * *command*, decaf::io::DataInputStream * *dis*, utils::BooleanStream * *bs*) [virtual]

Tight Un-marshal to the given stream.

Parameters

<i>format</i>	- The OpenwireFormat properties
<i>command</i>	- the object to Un-Marshal
<i>dis</i>	- the DataInputStream to Un-Marshal from
<i>bs</i>	- boolean stream to unmarshal from.

Exceptions

<i>IOException</i>	if an error occurs.
--------------------	---------------------

Reimplemented from **activemq::wireformat::openwire::marshal::generated::ActiveMQDestinationMarshaller** (p. 261).

The documentation for this class was generated from the following file:

- src/main/activemq/wireformat/openwire/marshal/generated/**ActiveMQTopicMarshaller.h**

6.51 activemq::core::ActiveMQTransactionContext Class Reference

Transaction Management class, hold messages that are to be redelivered upon a request to roll-back.

```
#include <src/main/activemq/core/ActiveMQTransaction-Context.h>
```

Inheritance diagram for **activemq::core::ActiveMQTransactionContext**:

Public Member Functions

- **ActiveMQTransactionContext** (**ActiveMQSession** *session, const **decaf::util::Properties** &properties)
Constructor.
- virtual ~**ActiveMQTransactionContext** ()
- virtual void **addSynchronization** (const **Pointer**< **Synchronization** > &sync)
*Adds a **Synchronization** (p. 2654) to this Transaction.*
- virtual void **removeSynchronization** (const **Pointer**< **Synchronization** > &sync)
*Removes a **Synchronization** (p. 2654) to this Transaction.*
- virtual void **begin** ()
Begins a new transaction if one is not currently in progress.
- virtual void **commit** ()
Commit the current Transaction.
- virtual void **rollback** ()
Rollback the current Transaction.
- virtual const **decaf::lang::Pointer** < **commands::TransactionId** > & **getTransactionId** () const
Get the Transaction Id object for the current Transaction, returns NULL if no transaction is running.
- virtual bool **isInTransaction** () const

Checks to see if there is currently a Transaction in progress returns false if not, true otherwise.

- virtual bool **isInLocalTransaction** () const

Checks to see if there is currently an Local Transaction in progress, returns false if not, true otherwise.

- virtual bool **isInXATransaction** () const

Checks to see if there is currently an XA Transaction in progress, returns false if not, true otherwise.

- virtual void **commit** (const **cms::Xid** *xid, bool onePhase)

Commits a global transaction.

- virtual void **end** (const **cms::Xid** *xid, int flags)

Ends the work done for a transaction branch.

- virtual void **forget** (const **cms::Xid** *xid)

Informs the Resource Manager that it can forget about a specified transaction branch.

- virtual int **getTransactionTimeout** () const

Gets the transaction timeout value for this XAResource.

- virtual bool **isSameRM** (const **cms::XAResource** *theXAResource)

Returns true if the ResourceManager for this XAResource is the same as the Resource Manager for a supplied XAResource.

- virtual int **prepare** (const **cms::Xid** *xid)

Requests the Resource manager to prepare to commit a specified transaction.

- virtual int **recover** (int flag, **cms::Xid** **recovered)

Get a list of prepared transaction branches.

- virtual void **rollback** (const **cms::Xid** *xid)

Requests the Resource Manager to rollback a specified transaction branch.

- virtual bool **setTransactionTimeout** (int seconds)

Sets the transaction timeout value for this XAResource.

- virtual void **start** (const **cms::Xid** *xid, int flags)

Starts work for a specified transaction branch.

6.51.1 Detailed Description

Transaction Management class, hold messages that are to be redelivered upon a request to roll-back.

The Transaction represents an always running transaction, when it is committed or rolled back it silently creates a new transaction for the next set of messages. The only way to permanently end this transaction is to delete it.

Since

2.0

6.51.2 Constructor & Destructor Documentation

6.51.2.1 `activemq::core::ActiveMQTransactionContext::ActiveMQTransactionContext (
 ActiveMQSession * session, const decaf::util::Properties & properties)`

Constructor.

Parameters

<i>session</i>	The session that contains this transaction
<i>properties</i>	Configuration parameters for this object

6.51.2.2 `virtual activemq::core::ActiveMQTransactionContext::~~ActiveMQ-
 TransactionContext () [virtual]`

6.51.3 Member Function Documentation

6.51.3.1 `virtual void activemq::core::ActiveMQTransactionContext::add-
 Synchronization (const Pointer< Synchronization > & sync)
 [virtual]`

Adds a **Synchronization** (p. 2654) to this Transaction.

Parameters

<i>sync</i>	- The Synchronization (p. 2654) instance to add.
-------------	---

6.51.3.2 `virtual void activemq::core::ActiveMQTransactionContext::begin ()
 [virtual]`

Begins a new transaction if one is not currently in progress.

Exceptions

<i>ActiveMQException</i>	
--------------------------	--

6.51.3.3 `virtual void activemq::core::ActiveMQTransactionContext::commit ()
 [virtual]`

Commit the current Transaction.

Exceptions

<i>ActiveMQException</i>	
--------------------------	--

6.51.3.4 `virtual void activemq::core::ActiveMQTransactionContext::commit (const cms::Xid * xid, bool onePhase) [virtual]`

Commits a global transaction.

Parameters

<i>xid</i>	the XID which identifies the global transaction.
<i>onePhase</i>	true if the resource manager should use a one-phase commit protocol to commit the transaction.

Exceptions

<i>XAException</i>	if an error occurred.
--------------------	-----------------------

Possible errors are identified by the errorcode in the XAException and include: XA_HEURHAZ, XA_HEURCOM, XA_HEURRB, XA_HEURMIX, XAER_RMERR, XAER_RMFAIL, XAER_NOTA, XAER_INVALID, or XAER_PROTO. In addition, one of the XA_RB* errors can occur if the transaction was not committed and onePhase was set to true. On completion of this method, the Resource Manager has rolled back the transaction and released resources held by the transaction.

Implements **cms::XAResource** (p. 2928).

6.51.3.5 `virtual void activemq::core::ActiveMQTransactionContext::end (const cms::Xid * xid, int flags) [virtual]`

Ends the work done for a transaction branch.

The Resource Manager disconnects the XA resource from the transaction branch and allows the transaction to complete.

Parameters

<i>xid</i>	the XID which identifies the global transaction. Should have previously been used as the parameter to a start. method.
<i>flags</i>	a flags integer - one of: XAResource::TMSUCCESS, XAResource::TMFAIL, or XAResource::TMSUSPEND.

TMSUCCESS means that this section of work completed successfully.

TMFAIL means that this section of work failed. The Resource Manager can mark the transaction for rollback only.

TMSUSPEND means that this section of work is suspended and not yet complete. The associated transaction context is also suspended and must be restarted with a call to **start(Xid, int)** (p. ??) with the TMRESUME flag.

Exceptions

<i>XAException</i>	if an error occurs. Possible error identified in the errorcode include: XAER_RMERR, XAER_RMFAIL, XAER_NOTA, XAER_INVAL, XAER_PROTO, or XA_RB*.
--------------------	--

Implements **cms::XAResource** (p.2929).

6.51.3.6 virtual void **activemq::core::ActiveMQTransactionContext::forget** (const **cms::Xid** * *xid*) [virtual]

Informs the Resource Manager that it can forget about a specified transaction branch.

Parameters

<i>xid</i>	the XID which identifies the global transaction.
------------	--

Exceptions

<i>XAException</i>	if an error occurs. Possible error identified in the errorcode include: XAER_RMERR, XAER_RMFAIL, XAER_NOTA, XAER_INVAL, or XAER_PROTO.
--------------------	--

Implements **cms::XAResource** (p.2930).

6.51.3.7 virtual const **decaf::lang::Pointer**<**commands::TransactionId**>& **activemq::core::ActiveMQTransactionContext::getTransactionId** () const [virtual]

Get the Transaction Id object for the current Transaction, returns NULL if no transaction is running.

Returns

TransactionInfo

Exceptions

<i>InvalidStateException</i>	if a Transaction is not in progress.
------------------------------	--------------------------------------

6.51.3.8 virtual int **activemq::core::ActiveMQTransactionContext::getTransactionTimeout** () const [virtual]

Gets the transaction timeout value for this XAResource.

The default timeout value is the default timeout value set for the Resource Manager.

Returns

the transaction timeout value for this XAResource in seconds.

Exceptions

<i>XAException</i>	if an error occurs. Possible error identified in the errorcode include: XAER_RMERR and XAER_RMFAIL.
--------------------	---

Implements **cms::XAResource** (p. 2930).

6.51.3.9 `virtual bool activemq::core::ActiveMQTransactionContext::isInLocalTransaction () const [virtual]`

Checks to see if there is currently an Local Transaction in progress, returns false if not, true otherwise.

Returns

true if an Local Transaction is in progress.

6.51.3.10 `virtual bool activemq::core::ActiveMQTransactionContext::isInTransaction () const [virtual]`

Checks to see if there is currently a Transaction in progress returns false if not, true otherwise.

Returns

true if a transaction is in progress.

6.51.3.11 `virtual bool activemq::core::ActiveMQTransactionContext::isInXATransaction () const [virtual]`

Checks to see if there is currently an XA Transaction in progress, returns false if not, true otherwise.

Returns

true if an XA Transaction is in progress.

6.51.3.12 `virtual bool activemq::core::ActiveMQTransactionContext::isSameRM (const cms::XAResource * theXAResource) [virtual]`

Returns true if the ResourceManager for this XAResource is the same as the Resource Manager for a supplied XAResource.

Parameters

<i>theXA-Resource</i>	an XAResource object
-----------------------	----------------------

Returns

true if the Resource Manager for this XAResource is the same as the Resource Manager for *theXAResource*.

Exceptions

<i>XAException</i>	if an error occurs. Possible error identified in the errorcode include: XAER_RMERR and XAER_RMFAIL.
--------------------	---

Implements **cms::XAResource** (p. 2930).

6.51.3.13 `virtual int activemq::core::ActiveMQTransactionContext::prepare (const cms::Xid * xid)` [virtual]

Requests the Resource manager to prepare to commit a specified transaction.

Parameters

<i>xid</i>	the XID which identifies the global transaction.
------------	--

Returns

an integer: XA_RDONLY or XA_OK. XA_OK implies that the transaction work has been prepared normally, XA_RDONLY implies that the transaction branch is read only and has been committed. If there is a failure which requires a rollback, an XAException is raised.

Exceptions

<i>XAException</i>	if an error occurs. Possible error identified in the errorcode include: XA_RB*, XAER_RMERR, XAER_RMFAIL, XAER_NOTA, XAER_INVALID, or XAER_PROTO.
--------------------	--

Implements **cms::XAResource** (p. 2931).

6.51.3.14 `virtual int activemq::core::ActiveMQTransactionContext::recover (int flag, cms::Xid ** recovered)` [virtual]

Get a list of prepared transaction branches.

Typically used by a transaction manager during recovery to find transaction branches that are in prepared or heuristically completed states.

Parameters

<i>flag</i>	an integer. Must be one of: XAResource::TMSTARTRSCAN, XAResource::TMENDRSCAN, XAResource::TMNOFLAGS.
-------------	--

Returns

an array of zero or more XIDs identifying the transaction branches in the prepared or heuristically completed states.

Exceptions

<i>XAException</i>	if an error occurs. Possible error identified in the errorcode include: XAER_RMERR, XAER_RMFAIL, XAER_INVALID, and XAER_PROTO.
--------------------	--

Implements **cms::XAResource** (p. 2931).

6.51.3.15 **virtual void activemq::core::ActiveMQTransactionContext::removeSynchronization (const Pointer< Synchronization > & sync)**
[virtual]

Removes a **Synchronization** (p. 2654) to this Transaction.

Parameters

<i>sync</i>	- The Synchronization (p. 2654) instance to add.
-------------	---

6.51.3.16 **virtual void activemq::core::ActiveMQTransactionContext::rollback ()**
[virtual]

Rollback the current Transaction.

Exceptions

<i>ActiveMQException</i>	
--------------------------	--

6.51.3.17 **virtual void activemq::core::ActiveMQTransactionContext::rollback (const cms::Xid * xid)** [virtual]

Requests the Resource Manager to rollback a specified transaction branch.

Parameters

<i>xid</i>	the XID which identifies the transaction branch.
------------	--

Exceptions

<i>XAException</i>	if an error occurs.
--------------------	---------------------

Implements **cms::XAResource** (p.2932).

6.51.3.18 `virtual bool activemq::core::ActiveMQTransactionContext::setTransactionTimeout (int seconds)`
[virtual]

Sets the transaction timeout value for this XAResource.

If the value is set to 0, the default timeout value for the Resource Manager is used.

Parameters

<i>seconds</i>	the new Timeout value in seconds.
----------------	-----------------------------------

Returns

true if the transaction timeout value has been updated, false otherwise.

Exceptions

<i>XAException</i>	if an error occurs. Possible error identified in the errorcode include: XAER_RMERR, XAER_RMFAIL, or XAER_INVALID.
--------------------	---

Implements **cms::XAResource** (p.2932).

6.51.3.19 `virtual void activemq::core::ActiveMQTransactionContext::start (const cms::Xid * xid, int flags)` [virtual]

Starts work for a specified transaction branch.

Parameters

<i>xid</i>	the XID which identifies the transaction branch.
<i>flags</i>	an integer. Must be one of XAResource::TMNOFLAGS, XAResource::TMJOIN, or XAResource::TMRESUME.

TMJOIN implies that the start applies to joining a transaction previously passed to the Resource Manager.

TMRESUME implies that the start applies to a suspended transaction that should be restarted.

If TMNOFLAGS is specified, then if the transaction has been previously seen by the Resource Manager, an XAException is raised with the code XAER_DUPID.

Exceptions

<i>XAException</i>	if an error occurs. Possible error identified in the errorcode include: XA_RB*, XAER_RMERR, XAER_RMFAIL, XAER_DUPID, XAER_OUTSIDE, XAER_NOTA, XAER_INVAL, or XAER_PROTO.
--------------------	--

Implements **cms::XAResource** (p. 2933).

The documentation for this class was generated from the following file:

- src/main/activemq/core/**ActiveMQTransactionContext.h**

6.52 activemq::core::ActiveMQXAConnection Class Reference

```
#include <src/main/activemq/core/ActiveMQXAConnection.h>
```

Inheritance diagram for activemq::core::ActiveMQXAConnection:

Public Member Functions

- **ActiveMQXAConnection** (const **Pointer**< **transport::Transport** > &transport, const **Pointer**< **decaf::util::Properties** > &properties)
- virtual ~**ActiveMQXAConnection** () throw ()
- virtual **cms::XASession** * **createXASession** ()
Creates an XASession object.
- virtual **cms::Session** * **createSession** (**cms::Session::AcknowledgeMode** ackMode)

*Creates a new **Session** (p. 2361) to work for this **Connection** (p. 933) using the specified acknowledgment mode.*

Parameters

ackMode	<i>the Acknowledgment Mode to use.</i>
---------	--

Exceptions

CMSEException (p. 826)

6.52.1 Constructor & Destructor Documentation

6.52.1.1 **activemq::core::ActiveMQXAConnection::ActiveMQXAConnection** (const **Pointer**< **transport::Transport** > & *transport*, const **Pointer**< **decaf::util::Properties** > & *properties*)

6.52.1.2 virtual **activemq::core::ActiveMQXAConnection::~~ActiveMQXAConnection** () throw () [virtual]

6.52.2 Member Function Documentation

6.52.2.1 `virtual cms::Session* activemq::core::ActiveMQXAConnectionFactory::createSession (cms::Session::AcknowledgeMode ackMode)`
`[virtual]`

Creates a new **Session** (p. 2361) to work for this **Connection** (p. 933) using the specified acknowledgment mode.

Parameters

<i>ackMode</i>	the Acknowledgment Mode to use.
----------------	---------------------------------

Exceptions

CMSException (p. 826)	
---------------------------------	--

Reimplemented from **activemq::core::ActiveMQConnection** (p. 196).

6.52.2.2 `virtual cms::XASession* activemq::core::ActiveMQXAConnectionFactory::createXASession ()` `[virtual]`

Creates an XASession object.

Returns

a newly created XASession instance, caller owns the pointer.

Exceptions

CMSException	If the XAConnection object fails to create the XASession instance due to an internal error.
---------------------	---

Implements **cms::XAConnection** (p. 2918).

The documentation for this class was generated from the following file:

- `src/main/activemq/core/ActiveMQXAConnectionFactory.h`

6.53 activemq::core::ActiveMQXAConnectionFactory Class Reference

```
#include <src/main/activemq/core/ActiveMQXAConnectionFactory.h>
```

Inheritance diagram for `activemq::core::ActiveMQXAConnectionFactory`:

Public Member Functions

- **ActiveMQXAConnectionFactory** ()
- **ActiveMQXAConnectionFactory** (const std::string &uri, const std::string &username="", const std::string &password="")
Constructor.
- **ActiveMQXAConnectionFactory** (const **decaf::net::URI** &uri, const std::string &username="", const std::string &password="")
Constructor.
- virtual ~**ActiveMQXAConnectionFactory** () throw ()
- virtual **cms::XAConnection** * **createXAConnection** ()
Creates an XAConnection with the default user name and password.
- virtual **cms::XAConnection** * **createXAConnection** (const std::string &username, const std::string &password)
Creates an XA connection with the specified user name and password.

Protected Member Functions

- virtual **ActiveMQConnection** * **createActiveMQConnection** (const **Pointer**<**transport::Transport**> &transport, const **Pointer**<**decaf::util::Properties**> &properties)
*Create a new **ActiveMQConnection** (p. 187) instance using the provided Transport and Properties.*

6.53.1 Constructor & Destructor Documentation

6.53.1.1 **activemq::core::ActiveMQXAConnectionFactory::ActiveMQXAConnectionFactory** ()

6.53.1.2 **activemq::core::ActiveMQXAConnectionFactory::ActiveMQXAConnectionFactory** (const std::string & uri, const std::string & username = " ", const std::string & password = " ")

Constructor.

Parameters

<i>uri</i>	the URI of the Broker we are connecting to.
<i>username</i>	to authenticate with, defaults to ""
<i>password</i>	to authenticate with, defaults to ""

6.53.1.3 **activemq::core::ActiveMQXAConnectionFactory::ActiveMQXAConnectionFactory** (*const decaf::net::URI & uri*, *const std::string & username* = " ", *const std::string & password* = " ")

Constructor.

Parameters

<i>uri</i>	the URI of the Broker we are connecting to.
<i>username</i>	to authenticate with, defaults to ""
<i>password</i>	to authenticate with, defaults to ""

6.53.1.4 **virtual activemq::core::ActiveMQXAConnectionFactory::~~ActiveMQXAConnectionFactory** () throw ()
[virtual]

6.53.2 Member Function Documentation

6.53.2.1 **virtual ActiveMQConnection* activemq::core::ActiveMQXAConnectionFactory::createActiveMQConnection** (*const Pointer< transport::Transport > & transport*, *const Pointer< decaf::util::Properties > & properties*) [protected, virtual]

Create a new **ActiveMQConnection** (p. 187) instance using the provided Transport and Properties.

Subclasses can override this to control the actual type of **ActiveMQConnection** (p. 187) that is created.

Parameters

<i>transport</i>	The Transport that the Connection should use to communicate with the Broker.
<i>properties</i>	The Properties that are assigned to the new Connection instance.

Returns

a new **ActiveMQConnection** (p. 187) pointer instance.

Reimplemented from **activemq::core::ActiveMQConnectionFactory** (p. 217).

6.53.2.2 **virtual cms::XAConnection* activemq::core::ActiveMQXAConnectionFactory::createXAConnection** ()
[virtual]

Creates an XAConnection with the default user name and password.

The connection is created in stopped mode just as the standard Connection object is created from the ConnectionFactory. No messages will be delivered until the Connection.start method is explicitly called.

Returns

a new `XAConnectionFactory` instance, the caller owns the returned pointer.

Exceptions

<i>CMSException</i>	if an internal error occurs while creating the Connection.
<i>CMSSecurity-Exception</i>	if the client authentication fails because the user name or password are invalid.

Implements **cms::XAConnectionFactory** (p. 2920).

6.53.2.3 virtual `cms::XAConnection* activemq::core::ActiveMQXAConnectionFactory::createXAConnection (const std::string & userName, const std::string & password)` [virtual]

Creates an XA connection with the specified user name and password.

The connection is created in stopped mode just as the standard `ConnectionFactory` creates a new `Connection`. No messages will be delivered until the `Connection.start` method is explicitly called.

Returns

a new `XAConnectionFactory` instance, the caller owns the returned pointer.

Exceptions

<i>CMSException</i>	if an internal error occurs while creating the Connection.
<i>CMSSecurity-Exception</i>	if the client authentication fails because the user name or password are invalid.

Implements **cms::XAConnectionFactory** (p. 2920).

The documentation for this class was generated from the following file:

- `src/main/activemq/core/ActiveMQXAConnectionFactory.h`

6.54 activemq::core::ActiveMQXASession Class Reference

```
#include <src/main/activemq/core/ActiveMQXASession.h>
```

Inheritance diagram for `activemq::core::ActiveMQXASession`:

Public Member Functions

- **ActiveMQXASession** (**ActiveMQConnection** ***connection**, const **Pointer**<**commands::SessionId** > &**sessionId**, const **decaf::util::Properties** &**properties**)
- virtual **~ActiveMQXASession** () throw ()
- virtual bool **isTransacted** () const
Gets if the Sessions is a Transacted Session.
- virtual bool **isAutoAcknowledge** () const
- virtual void **doStartTransaction** ()
Starts if not already start a Transaction for this Session.
- virtual void **commit** ()
Commits all messages done in this transaction and releases any locks currently held.
- virtual void **rollback** ()
Rolls back all messages done in this transaction and releases any locks currently held.
- virtual **cms::XAResource** * **getXAResource** () const
Returns the XA resource associated with this Session to the caller.

6.54.1 Constructor & Destructor Documentation

- 6.54.1.1 **activemq::core::ActiveMQXASession::ActiveMQXASession** (**ActiveMQConnection** * **connection**, const **Pointer**< **commands::SessionId** > & **sessionId**, const **decaf::util::Properties** & **properties**)
- 6.54.1.2 virtual **activemq::core::ActiveMQXASession::~~ActiveMQXASession** () throw () [virtual]

6.54.2 Member Function Documentation

- 6.54.2.1 virtual void **activemq::core::ActiveMQXASession::commit** () [virtual]

Commits all messages done in this transaction and releases any locks currently held.

Exceptions

<i>CMSException</i>	- If an internal error occurs.
<i>IllegalStateException</i>	- if the method is not called by a transacted session.

Reimplemented from **activemq::core::ActiveMQSession** (p. 338).

- 6.54.2.2 virtual void **activemq::core::ActiveMQXASession::doStartTransaction** () [virtual]

Starts if not already start a Transaction for this Session.

If the session is not a Transacted Session then an exception is thrown. If a transaction is already in progress then this method has no effect.

Exceptions

<i>ActiveMQException</i>	if this is not a Transacted Session.
--------------------------	--------------------------------------

Reimplemented from **activemq::core::ActiveMQSession** (p. 346).

6.54.2.3 `virtual cms::XAResource* activemq::core::ActiveMQXASession::getXA-Resource () const [virtual]`

Returns the XA resource associated with this Session to the caller.

The client can use the provided XA resource to interact with the XA Transaction Manager in use in the client application.

Returns

an XAResource instance to the caller, the caller does not own this pointer and should not delete it.

Implements **cms::XASession** (p. 2936).

6.54.2.4 `virtual bool activemq::core::ActiveMQXASession::isAutoAcknowledge () const [virtual]`

Reimplemented from **activemq::core::ActiveMQSession** (p. 349).

6.54.2.5 `virtual bool activemq::core::ActiveMQXASession::isTransacted () const [virtual]`

Gets if the Sessions is a Transacted Session.

Returns

transacted true - false.

Exceptions

<i>CMSEException</i>	- If an internal error occurs.
----------------------	--------------------------------

Reimplemented from **activemq::core::ActiveMQSession** (p. 349).

6.54.2.6 virtual void **activemq::core::ActiveMQXASession::rollback** ()
[virtual]

Rolls back all messages done in this transaction and releases any locks currently held.

Exceptions

<i>CMSEException</i>	- If an internal error occurs.
<i>IllegalStateException</i>	- if the method is not called by a transacted session.

Reimplemented from **activemq::core::ActiveMQSession** (p. 351).

The documentation for this class was generated from the following file:

- src/main/activemq/core/**ActiveMQXASession.h**

6.55 decaf::util::zip::Adler32 Class Reference

Clas that can be used to compute an Adler-32 **Checksum** (p. 810) for a data stream.

```
#include <src/main/decaf/util/zip/Adler32.h>
```

Inheritance diagram for decaf::util::zip::Adler32:

Public Member Functions

- **Adler32** ()
- virtual \sim **Adler32** ()
- virtual long long **getValue** () const
- virtual void **reset** ()
Reset the checksum to its initial value.
- virtual void **update** (const std::vector< unsigned char > &buffer)
Updates the current checksum with the specified vector of bytes.
- virtual void **update** (const std::vector< unsigned char > &buffer, int offset, int length)
Updates the current checksum with the specified array of bytes.
- virtual void **update** (const unsigned char *buffer, int size, int offset, int length)
Updates the current checksum with the specified array of bytes.
- virtual void **update** (int byte)
Updates the current checksum with the specified byte value.

6.55.1 Detailed Description

Clas that can be used to compute an Adler-32 **Checksum** (p. 810) for a data stream.
The Alder-32 checksum trades reliability for speed over the CRC-32 algorithm.

Since

1.0

6.55.2 Constructor & Destructor Documentation

6.55.2.1 `decaf::util::zip::Adler32::Adler32 ()`

6.55.2.2 `virtual decaf::util::zip::Adler32::~~Adler32 () [virtual]`

6.55.3 Member Function Documentation

6.55.3.1 `virtual long long decaf::util::zip::Adler32::getValue () const [virtual]`

Returns

the current checksum value.

Implements **decaf::util::zip::Checksum** (p. 811).

6.55.3.2 `virtual void decaf::util::zip::Adler32::reset () [virtual]`

Reset the checksum to its initial value.

Implements **decaf::util::zip::Checksum** (p. 811).

6.55.3.3 `virtual void decaf::util::zip::Adler32::update (const std::vector< unsigned char > & buffer) [virtual]`

Updates the current checksum with the specified vector of bytes.

Parameters

<i>buffer</i>	The buffer to read the updated bytes from.
---------------	--

Implements **decaf::util::zip::Checksum** (p. 811).

6.55.3.4 `virtual void decaf::util::zip::Adler32::update (const std::vector< unsigned char > & buffer, int offset, int length) [virtual]`

Updates the current checksum with the specified array of bytes.

Parameters

<i>buffer</i>	The buffer to read the updated bytes from.
<i>offset</i>	The position in the buffer to start reading.
<i>length</i>	The amount of data to read from the byte buffer.

Exceptions

<i>IndexOutOfBoundsException</i>	if $\text{offset} + \text{length} > \text{size of the buffer}$.
----------------------------------	--

Implements **decaf::util::zip::Checksum** (p. 811).

6.55.3.5 virtual void **decaf::util::zip::Adler32::update** (const unsigned char * *buffer*, int *size*, int *offset*, int *length*) [virtual]

Updates the current checksum with the specified array of bytes.

Parameters

<i>buffer</i>	The buffer to read the updated bytes from.
<i>size</i>	The size of the passed buffer.
<i>offset</i>	The position in the buffer to start reading.
<i>length</i>	The amount of data to read from the byte buffer.

Exceptions

<i>NullPointerException</i>	if the passed buffer is NULL.
<i>IndexOutOfBoundsException</i>	if $\text{offset} + \text{length} > \text{size of the buffer}$.

Implements **decaf::util::zip::Checksum** (p. 812).

6.55.3.6 virtual void **decaf::util::zip::Adler32::update** (int *byte*) [virtual]

Updates the current checksum with the specified byte value.

Parameters

<i>byte</i>	The byte value to update the current Checksum (p. 810) with (0..255).
-------------	--

Implements **decaf::util::zip::Checksum** (p. 812).

The documentation for this class was generated from the following file:

- src/main/decaf/util/zip/**Adler32.h**

6.56 decaf::lang::Appendable Class Reference

An object to which char sequences and values can be appended.

```
#include <src/main/decaf/lang/Appendable.h>
```

Inheritance diagram for decaf::lang::Appendable:

Public Member Functions

- virtual **~Appendable** ()
- virtual **Appendable & append** (char value)=0
*Appends the specified character to this **Appendable** (p. 442).*
- virtual **Appendable & append** (const **CharSequence** *csq)=0
*Appends the specified character sequence to this **Appendable** (p. 442).*
- virtual **Appendable & append** (const **CharSequence** *csq, int start, int end)=0
*Appends a subsequence of the specified character sequence to this **Appendable** (p. 442).*

6.56.1 Detailed Description

An object to which char sequences and values can be appended.

The **Appendable** (p. 442) interface must be implemented by any class whose instances are intended to receive formatted output from a Formatter.

TODO The characters to be appended should be valid Unicode characters as described in Unicode **Character** (p. 765) Representation. Note that supplementary characters may be composed of multiple 16-bit char values.

Appendables are not necessarily safe for multithreaded access. **Thread** (p. 2703) safety is the responsibility of classes that extend and implement this interface.

Since this interface may be implemented by existing classes with different styles of error handling there is no guarantee that errors will be propagated to the invoker.

Since

1.0

6.56.2 Constructor & Destructor Documentation

6.56.2.1 virtual decaf::lang::Appendable::~~Appendable () [inline, virtual]

6.56.3 Member Function Documentation

6.56.3.1 `virtual Appendable& decaf::lang::Appendable::append (char value)`
[pure virtual]

Appends the specified character to this **Appendable** (p. 442).

Parameters

<i>value</i>	The character to append.
--------------	--------------------------

Returns

a Reference to this **Appendable** (p. 442)

Exceptions

Exception (p. 1279)	if an error occurs.
----------------------------	---------------------

Implemented in **decaf::io::Writer** (p. 2910), and **decaf::nio::CharBuffer** (p. 789).

6.56.3.2 `virtual Appendable& decaf::lang::Appendable::append (const CharSequence * csq)` [pure virtual]

Appends the specified character sequence to this **Appendable** (p. 442).

Parameters

<i>csq</i>	The character sequence from which a subsequence will be appended. If <i>csq</i> is NULL, then characters will be appended as if <i>csq</i> contained the string "null".
------------	---

Returns

a Reference to this **Appendable** (p. 442).

Exceptions

Exception (p. 1279)	if an error occurs.
----------------------------	---------------------

Implemented in **decaf::io::Writer** (p. 2910), and **decaf::nio::CharBuffer** (p. 789).

6.56.3.3 `virtual Appendable& decaf::lang::Appendable::append (const CharSequence * csq, int start, int end)` [pure virtual]

Appends a subsequence of the specified character sequence to this **Appendable** (p. 442).

Parameters

<i>csq</i>	- The character sequence from which a subsequence will be appended. If <i>csq</i> is NULL, then characters will be appended as if <i>csq</i> contained the string "null".
<i>start</i>	The index of the first character in the subsequence.
<i>end</i>	The index of the character following the last character in the subsequence.

Returns

a Reference to this **Appendable** (p. 442)

Exceptions

Exception (p. 1279)	if an error occurs.
<i>IndexOutOfBoundsException</i>	<i>start</i> is greater than <i>end</i> , or <i>end</i> is greater than <i>csq.length()</i>

Implemented in **decaf::nio::CharBuffer** (p. 790), and **decaf::io::Writer** (p. 2911).

The documentation for this class was generated from the following file:

- `src/main/decaf/lang/Appendable.h`

6.57 decaf::internal::AprPool Class Reference

Wraps an APR pool object so that classes in decaf can create a static member for use in static methods where apr function calls that need a pool are made.

```
#include <src/main/decaf/internal/AprPool.h>
```

Public Member Functions

- **AprPool** ()
- virtual **~AprPool** ()
- `apr_pool_t * getAprPool () const`
Gets the internal APR Pool.
- void **cleanup** ()
Clears data that was allocated by this pool.

Static Public Member Functions

- static `apr_pool_t * getGlobalPool ()`
Gets a pointer to the Global APR Pool used for the Application.

6.57.1 Detailed Description

Wraps an APR pool object so that classes in decaf can create a static member for use in static methods where apr function calls that need a pool are made.

6.57.2 Constructor & Destructor Documentation

6.57.2.1 `decaf::internal::AprPool::AprPool ()`

6.57.2.2 `virtual decaf::internal::AprPool::~~AprPool () [virtual]`

6.57.3 Member Function Documentation

6.57.3.1 `void decaf::internal::AprPool::cleanup ()`

Clears data that was allocated by this pool.

Users should call this after getting the data from the APR functions and copying it to someplace safe.

6.57.3.2 `apr_pool_t* decaf::internal::AprPool::getAprPool () const`

Gets the internal APR Pool.

Returns

the internal APR pool

6.57.3.3 `static apr_pool_t* decaf::internal::AprPool::getGlobalPool () [static]`

Gets a pointer to the Global APR Pool used for the Application.

Returns

pointer to the global APR Pool

The documentation for this class was generated from the following file:

- `src/main/decaf/internal/AprPool.h`

6.58 decaf::util::ArrayList< E > Class Template Reference

```
#include <src/main/decaf/util/ArrayList.h>
```

Inheritance diagram for `decaf::util::ArrayList< E >`:

Public Member Functions

- **ArrayList** ()
- **ArrayList** (const **Collection**< E > &collection)
- **ArrayList** (const **ArrayList**< E > &arrayList)
- **ArrayList** (int initialCapacity)
- virtual ~**ArrayList** ()
- **ArrayList**< E > & **operator=** (const **ArrayList**< E > &list)
- **ArrayList**< E > & **operator=** (const **Collection**< E > &collection)
- void **ensureCapacity** (int minimumCapacity)
*Increases the capacity of this **ArrayList** (p. 445) instance, if necessary, to ensure that it can hold at least the number of elements specified by the minimum capacity argument.*
- void **trimToSize** ()
*Trims the internal array to match the current size of the **ArrayList** (p. 445).*
- virtual void **clear** ()
Removes all of the elements from this collection (optional operation).
- virtual bool **isEmpty** () const
Returns true if this collection contains no elements.
- virtual int **size** () const
Returns the number of elements in this collection.
- virtual E **set** (int index, const E &element)
Replaces the element at the specified position in this list with the specified element.
- virtual E **get** (int index) const
Gets the element contained at position passed.
- virtual bool **add** (const E &value)
Returns true if this collection changed as a result of the call.
- virtual void **add** (int index, const E &element)
Inserts the specified element at the specified position in this list.
- virtual bool **addAll** (const **Collection**< E > &collection)
*Adds all of the elements in the specified collection to this collection.
The behavior of this operation is undefined if the specified collection is modified while the operation is in progress. (This implies that the behavior of this call is undefined if the specified collection is this collection, and this collection is nonempty.)*

Parameters

collection	The Collection (p. 851) whose elements are added to this one.
------------	--

Returns

true if this collection changed as a result of the call

Exceptions

UnsupportedOperationException	<i>if this is an unmodifiable collection.</i>
NullPointerException	<i>if the Collection (p. 851) is a container of pointers and does not allow NULL values.</i>
IllegalArgumentException	<i>if some property of an element prevents it from being added to this collection</i>
IllegalStateException	<i>if an element cannot be added at this time due to insertion restrictions.</i>

This implementation iterates over the specified collection, and adds each object returned by the iterator to this collection, in turn.

Note that this implementation will throw an `UnsupportedOperationException` unless `add` is overridden (assuming the specified collection is non-empty).

- virtual bool **addAll** (int index, const **Collection**< E > &collection)

Inserts all of the elements in the specified collection into this list at the specified position (optional operation).

- virtual bool **remove** (const E &value)

Removes a single instance of the specified element from the collection.

More formally, removes an element `e` such that `(value == NULL ? e == NULL : value == e)`, if this collection contains one or more such elements. Returns true if this collection contained the specified element (or equivalently, if this collection changed as a result of the call).

Parameters

value	The reference to the element to remove from this Collection (p. 851).
-------	--

Returns

true if the collection was changed, false otherwise.

Exceptions

UnsupportedOperationException	if this is an unmodifiable collection.
NullPointerException	if the Collection (p. 851) is a container of pointers and does not allow NULL values.

This implementation iterates over the collection looking for the specified element. - If it finds the element, it removes the element from the collection using the iterator's `remove` method.

Note that this implementation throws an `UnsupportedOperationException` if the iterator returned by this collection's `iterator` method does not implement the `remove` method and this collection contains the specified object.

- virtual E **removeAt** (int index)

Removes the element at the specified position in this list.

- virtual bool **contains** (const E &value) const

Returns true if this collection contains the specified element.

More formally, returns true if and only if this collection contains at least one element `e` such that `(value == NULL ? e == NULL : value == e)`.

Parameters

value	The value to check for presence in the collection.
-------	--

Returns

true if there is at least one of the elements in the collection

Exceptions

NullPointerException	if the Collection (p. 851) contains pointers and the Collection (p. 851) does not allow for NULL elements (optional check).
----------------------	---

This implementation iterates over the elements in the collection, checking each element in turn for equality with the specified element.

- virtual int **indexOf** (const E &value) const

Returns the index of the first occurrence of the specified element in this list, or -1 if this list does not contain the element.

- virtual int **lastIndexOf** (const E &value) const

Returns the index of the last occurrence of the specified element in this list, or -1 if this list does not contain the element.

- virtual std::vector< E > **toArray** () const

*Answers an STL vector containing copies of all elements contained in this **Collection** (p. 851).*

- virtual std::string **toString** () const

```
template<typename E> class decaf::util::ArrayList< E >
```

6.58.1 Constructor & Destructor Documentation

6.58.1.1 `template<typename E> decaf::util::ArrayList< E >::ArrayList ()`
[inline]

References decaf::util::ArrayList< E >::ensureCapacity().

6.58.1.2 `template<typename E> decaf::util::ArrayList< E >::ArrayList (const Collection< E > & collection)` [inline]

References decaf::lang::Iterable< E >::iterator(), and decaf::util::Collection< E >::size().

6.58.1.3 `template<typename E> decaf::util::ArrayList< E >::ArrayList (const ArrayList< E > & arrayList)` [inline]

References decaf::util::AbstractList< E >::iterator(), and decaf::util::ArrayList< E >::size().

6.58.1.4 `template<typename E> decaf::util::ArrayList< E >::ArrayList (int initialCapacity)` [inline]

6.58.1.5 `template<typename E> virtual decaf::util::ArrayList< E >::~~ArrayList ()`
[inline, virtual]

References DECAF_CATCHALL_NOTHROW.

6.58.2 Member Function Documentation

6.58.2.1 `template<typename E> virtual bool decaf::util::ArrayList< E >::add (const E & value)` [inline, virtual]

Returns true if this collection changed as a result of the call.

(Returns false if this collection does not permit duplicates and already contains the specified element.)

Collections that support this operation may place limitations on what elements may be added to this collection. In particular, some collections will refuse to add null elements, and others will impose restrictions on the type of elements that may be added. **Collection** (p. 851) classes should clearly specify in their documentation any restrictions on what elements may be added.

If a collection refuses to add a particular element for any reason other than that it already contains the element, it must throw an exception (rather than returning false). This preserves the invariant that a collection always contains the specified element after this call returns.

For non-pointer values, i.e. class instances or string's the object will be copied into the collection, thus the object must support being copied, must not hide the copy constructor and assignment operator.

Parameters

<i>value</i>	The reference to the element to add to this Collection (p. 851).
--------------	---

Returns

true if the element was added to this **Collection** (p. 851).

Exceptions

<i>UnsupportedOperationException</i>	if this is an unmodifiable collection.
<i>NullPointerException</i>	if the Collection (p. 851) is a container of pointers and does not allow NULL values.
<i>IllegalArgumentException</i>	if some property of the element prevents it from being added to this collection
<i>IllegalStateException</i>	if the element cannot be added at this time due to insertion restrictions.

Reimplemented from **decaf::util::AbstractList< E >** (p. 125).

6.58.2.2 `template<typename E> virtual void decaf::util::ArrayList< E >::add (int index, const E & element) [inline, virtual]`

Inserts the specified element at the specified position in this list.

Shifts the element currently at that position (if any) and any subsequent elements to the right (adds one to their indices).

Parameters

<i>index</i>	The index at which the specified element is to be inserted.
<i>element</i>	The element to be inserted in this List (p. 1658).

Exceptions

<i>IndexOutOfBoundsException</i>	if the index is greater than size of the List (p. 1658).
<i>UnsupportedOperationException</i>	if this is an unmodifiable collection.
<i>NullPointerException</i>	if the Collection (p. 851) is a container of pointers and does not allow NULL values.
<i>IllegalArgumentException</i>	if some property of the element prevents it from being added to this collection
<i>IllegalStateException</i>	if the element cannot be added at this time due to insertion restrictions.

Implements **decaf::util::List< E >** (p. 1660).

6.58.2.3 `template<typename E> virtual bool decaf::util::ArrayList< E >::addAll (const Collection< E > & collection) [inline, virtual]`

Adds all of the elements in the specified collection to this collection.

The behavior of this operation is undefined if the specified collection is modified while the operation is in progress. (This implies that the behavior of this call is undefined if the specified collection is this collection, and this collection is nonempty.)

Parameters

<i>collection</i>	The Collection (p. 851) whose elements are added to this one.
-------------------	--

Returns

true if this collection changed as a result of the call

Exceptions

<i>UnsupportedOperationException</i>	if this is an unmodifiable collection.
<i>NullPointerException</i>	if the Collection (p. 851) is a container of pointers and does not allow NULL values.
<i>IllegalArgumentException</i>	if some property of an element prevents it from being added to this collection
<i>IllegalStateException</i>	if an element cannot be added at this time due to insertion restrictions.

This implementation iterates over the specified collection, and adds each object returned by the iterator to this collection, in turn.

Note that this implementation will throw an `UnsupportedOperationException` unless `add` is overridden (assuming the specified collection is non-empty).

This implementation iterates over the specified collection, and adds each object re-

turned by the iterator to this collection, in turn.

Note that this implementation will throw an UnsupportedOperationException unless add is overridden (assuming the specified collection is non-empty).

Reimplemented from **decaf::util::AbstractCollection< E >** (p. 110).

References decaf::util::Collection< E >::size(), and decaf::util::Collection< E >::toArray().

Referenced by decaf::util::ArrayList< E >::operator=().

6.58.2.4 `template<typename E> virtual bool decaf::util::ArrayList< E >::addAll (int index, const Collection< E > & source) [inline, virtual]`

Inserts all of the elements in the specified collection into this list at the specified position (optional operation).

Shifts the element currently at that position (if any) and any subsequent elements to the right (increases their indices). The new elements will appear in this list in the order that they are returned by the specified collection's iterator. The behavior of this operation is undefined if the specified collection is modified while the operation is in progress. (Note that this will occur if the specified collection is this list, and it's nonempty.)

Parameters

<i>index</i>	The index at which to insert the first element from the specified collection
<i>source</i>	The Collection (p. 851) containing elements to be added to this list

Returns

true if this list changed as a result of the call

Exceptions

<i>IndexOutOfBoundsException</i>	if the index given is less than zero or greater than the List (p. 1658) size.
<i>UnsupportedOperationException</i>	if this is an unmodifiable collection.
<i>NullPointerException</i>	if the Collection (p. 851) is a container of pointers and does not allow NULL values.
<i>IllegalArgumentException</i>	if some property of the element prevents it from being added to this collection
<i>IllegalStateException</i>	if the element cannot be added at this time due to insertion restrictions.

Reimplemented from **decaf::util::AbstractList< E >** (p. 126).

References decaf::util::Collection< E >::size(), and decaf::util::Collection< E >::toArray().

6.58.2.5 `template<typename E> virtual void decaf::util::ArrayList< E >::clear ()`
`[inline, virtual]`

Removes all of the elements from this collection (optional operation).

The collection will be empty after this method returns.

This implementation iterates over this collection, removing each element using the **Iterator.remove** (p. 1560) operation. Most implementations will probably choose to override this method for efficiency.

Note that this implementation will throw an `UnsupportedOperationException` if the iterator returned by this collection's iterator method does not implement the remove method and this collection is non-empty.

Exceptions

<i>Unsupported- OperationException</i>	if the clear operation is not supported by this collection
--	--

Reimplemented from **decaf::util::AbstractList< E >** (p. 127).

Referenced by `decaf::util::ArrayList< E >::operator=()`.

6.58.2.6 `template<typename E> virtual bool decaf::util::ArrayList< E >::contains (`
`const E & value) const [inline, virtual]`

Returns true if this collection contains the specified element.

More formally, returns true if and only if this collection contains at least one element *e* such that `(value == NULL ? e == NULL : value == e)`.

Parameters

<i>value</i>	The value to check for presence in the collection.
--------------	--

Returns

true if there is at least one of the elements in the collection

Exceptions

<i>NullPointerException</i>	if the Collection (p. 851) contains pointers and the Collection (p. 851) does not allow for NULL elements (optional check).
-----------------------------	---

This implementation iterates over the elements in the collection, checking each element in turn for equality with the specified element.

This implementation iterates over the elements in the collection, checking each element in turn for equality with the specified element.

Reimplemented from **decaf::util::AbstractCollection< E >** (p. 112).

References decaf::util::ArrayList< E >::indexOf().

6.58.2.7 `template<typename E> void decaf::util::ArrayList< E >::ensureCapacity (int minimumCapacity) [inline]`

Increases the capacity of this **ArrayList** (p. 445) instance, if necessary, to ensure that it can hold at least the number of elements specified by the minimum capacity argument.

If the capacity is already greater than or equal to the minimum capacity argument then the array is left unchanged.

Parameters

<i>minimum-Capacity</i>	The desired minimum capacity for this ArrayList (p. 445).
-------------------------	--

References decaf::lang::System::arraycopy().

Referenced by decaf::util::ArrayList< E >::ArrayList().

6.58.2.8 `template<typename E> virtual E decaf::util::ArrayList< E >::get (int index) const [inline, virtual]`

Gets the element contained at position passed.

Parameters

<i>index</i>	The position to get.
--------------	----------------------

Returns

value at index specified.

Exceptions

<i>IndexOutOfBoundsException</i>	if the index given is less than zero or greater than the List (p. 1658) size.
----------------------------------	--

Implements **decaf::util::List< E >** (p. 1662).

6.58.2.9 `template<typename E> virtual int decaf::util::ArrayList< E >::indexOf (const E & value) const [inline, virtual]`

Returns the index of the first occurrence of the specified element in this list, or -1 if this list does not contain the element.

More formally, returns the lowest index *i* such that `get(i) == value`, or -1 if there is no such index.

Parameters

<i>value</i>	The element to search for in this List (p. 1658).
--------------	--

Returns

the index of the first occurrence of the specified element in this list,

Exceptions

<i>NullPointerException</i>	if the Collection (p. 851) is a container of pointers and does not allow NULL values.
-----------------------------	--

Reimplemented from **decaf::util::AbstractList**< **E** > (p. 128).

Referenced by **decaf::util::ArrayList**< **E** >::contains(), and **decaf::util::ArrayList**< **E** >::remove().

```
6.58.2.10  template<typename E> virtual bool decaf::util::ArrayList< E >::isEmpty ( )
           const [inline, virtual]
```

Returns true if this collection contains no elements.

This implementation returns **size()** (p. 457) == 0.

Returns

true if the size method return 0.

Reimplemented from **decaf::util::AbstractCollection**< **E** > (p. 114).

```
6.58.2.11  template<typename E> virtual int decaf::util::ArrayList< E >::lastIndexOf (
           const E & value ) const [inline, virtual]
```

Returns the index of the last occurrence of the specified element in this list, or -1 if this list does not contain the element.

More formally, returns the highest index *i* such that **get**(*i*) == **value** or -1 if there is no such index.

Parameters

<i>value</i>	The element to search for in this List (p. 1658).
--------------	--

Returns

the index of the last occurrence of the specified element in this list.

Exceptions

<i>NullPointerException</i>	if the Collection (p. 851) is a container of pointers and does not allow NULL values.
-----------------------------	--

Reimplemented from **decaf::util::AbstractList< E >** (p. 129).

6.58.2.12 `template<typename E> ArrayList<E>& decaf::util::ArrayList< E >::operator= (const ArrayList< E > & list) [inline]`

References `decaf::util::ArrayList< E >::addAll()`, and `decaf::util::ArrayList< E >::clear()`.

6.58.2.13 `template<typename E> ArrayList<E>& decaf::util::ArrayList< E >::operator= (const Collection< E > & collection) [inline]`

References `decaf::util::ArrayList< E >::addAll()`, and `decaf::util::ArrayList< E >::clear()`.

6.58.2.14 `template<typename E> virtual bool decaf::util::ArrayList< E >::remove (const E & value) [inline, virtual]`

Removes a single instance of the specified element from the collection.

More formally, removes an element *e* such that (`value == NULL ? e == NULL : value == e`), if this collection contains one or more such elements. Returns true if this collection contained the specified element (or equivalently, if this collection changed as a result of the call).

Parameters

<i>value</i>	The reference to the element to remove from this Collection (p. 851).
--------------	--

Returns

true if the collection was changed, false otherwise.

Exceptions

<i>UnsupportedOperationException</i>	if this is an unmodifiable collection.
<i>NullPointerException</i>	if the Collection (p. 851) is a container of pointers and does not allow NULL values.

This implementation iterates over the collection looking for the specified element. If it finds the element, it removes the element from the collection using the iterator's `remove` method.

Note that this implementation throws an `UnsupportedOperationException` if the iterator returned by this collection's iterator method does not implement the `remove` method and this collection contains the specified object.

This implementation iterates over the collection looking for the specified element. If it finds the element, it removes the element from the collection using the iterator's `remove` method.

Note that this implementation throws an `UnsupportedOperationException` if the iterator returned by this collection's iterator method does not implement the `remove` method and this collection contains the specified object.

Reimplemented from **decaf::util::AbstractCollection**< **E** > (p. 116).

References `decaf::util::ArrayList< E >::indexOf()`, and `decaf::util::ArrayList< E >::removeAt()`.

6.58.2.15 `template<typename E> virtual E decaf::util::ArrayList< E >::removeAt (int index) [inline, virtual]`

Removes the element at the specified position in this list.

Shifts any subsequent elements to the left (subtracts one from their indices). Returns the element that was removed from the list.

Parameters

<i>index</i>	- the index of the element to be removed.
--------------	---

Returns

the element previously at the specified position.

Exceptions

<i>IndexOutOfBoundsException</i>	if the index given is less than zero or greater than the List (p. 1658) size.
<i>UnsupportedOperationException</i>	if this is an unmodifiable collection.

Reimplemented from **decaf::util::AbstractList**< **E** > (p. 133).

References `decaf::lang::System::arraycopy()`.

Referenced by `decaf::util::ArrayList< E >::remove()`.

6.58.2.16 `template<typename E> virtual E decaf::util::ArrayList< E >::set (int index, const E & element) [inline, virtual]`

Replaces the element at the specified position in this list with the specified element.

Parameters

<i>index</i>	The index of the element to replace.
<i>element</i>	The element to be stored at the specified position.

Returns

the element previously at the specified position.

Exceptions

<i>IndexOutOfBoundsException</i>	if the index given is less than zero or greater than the List (p. 1658) size.
<i>UnsupportedOperationException</i>	if this is an unmodifiable collection.
<i>NullPointerException</i>	if the Collection (p. 851) is a container of pointers and does not allow NULL values.
<i>IllegalArgumentException</i>	if some property of the element prevents it from being added to this collection
<i>IllegalStateException</i>	if the element cannot be added at this time due to insertion restrictions.

Implements **decaf::util::List< E >** (p. 1669).

```
6.58.2.17  template<typename E> virtual int decaf::util::ArrayList< E >::size ( ) const
           [inline, virtual]
```

Returns the number of elements in this collection.

If this collection contains more than Integer::MAX_VALUE elements, returns Integer::MAX_VALUE.

Returns

the number of elements in this collection

Implements **decaf::util::Collection< E >** (p. 864).

Referenced by decaf::util::ArrayList< E >::ArrayList().

```
6.58.2.18  template<typename E> virtual std::vector<E> decaf::util::ArrayList< E
           >::toArray ( ) const [inline, virtual]
```

Answers an STL vector containing copies of all elements contained in this **Collection** (p. 851).

All the elements in the array will not be referenced by the collection. The elements in the returned array will be sorted to the same order as those returned by the iterator of this collection itself if the collection guarantees the order.

Returns

an vector of copies of all the elements from this **Collection** (p. 851)

Reimplemented from **decaf::util::AbstractCollection**< **E** > (p. 119).

```
6.58.2.19  template<typename E> virtual std::string decaf::util::ArrayList< E >::toString
           ( ) const [inline, virtual]
```

References decaf::lang::Integer::toString().

```
6.58.2.20  template<typename E> void decaf::util::ArrayList< E >::trimToSize ( )
           [inline]
```

Trims the internal array to match the current size of the **ArrayList** (p. 445).

This compacts the internal array to save storage space used by this **ArrayList** (p. 445).

References decaf::lang::System::arraycopy().

The documentation for this class was generated from the following file:

- src/main/decaf/util/**ArrayList.h**

6.59 decaf::util::concurrent::CopyOnWriteArrayList< E >::ArrayListIterator Class Reference

```
#include <src/main/decaf/util/concurrent/CopyOnWrite-
ArrayList.h>
```

Inheritance diagram for decaf::util::concurrent::CopyOnWriteArrayList< E >::ArrayListIterator:

Public Member Functions

- **ArrayListIterator** (const **ArrayPointer**< E > &array, int index)
- virtual **~ArrayListIterator** ()
- virtual E **next** ()
Returns the next element in the iteration.
- virtual bool **hasNext** () const
Returns true if the iteration has more elements.
- virtual void **remove** ()
Removes from the underlying collection the last element returned by the iterator (optional operation).
- virtual void **add** (const E &e **DECAF_UNUSED**)

- virtual void **set** (const E &e **DECAF_UNUSED**)
- virtual bool **hasPrevious** () const
Returns true if this list iterator has more elements when traversing the list in the reverse direction.
- virtual E **previous** ()
Returns the previous element in the list.
- virtual int **nextIndex** () const
Returns the index of the element that would be returned by a subsequent call to next.
- virtual int **previousIndex** () const
Returns the index of the element that would be returned by a subsequent call to previous.

template<typename E> class decaf::util::concurrent::CopyOnWriteArrayList< E >::ArrayListIterator

6.59.1 Constructor & Destructor Documentation

6.59.1.1 template<typename E> decaf::util::concurrent::CopyOnWriteArrayList< E >::ArrayListIterator (const ArrayPointer< E > &array, int index) [inline]

References decaf::lang::ArrayPointer< T, REFCOUNTER >::length().

6.59.1.2 template<typename E> virtual decaf::util::concurrent::CopyOnWriteArrayList< E >::ArrayListIterator::~~ArrayListIterator () [inline, virtual]

6.59.2 Member Function Documentation

6.59.2.1 template<typename E> virtual void decaf::util::concurrent::CopyOnWriteArrayList< E >::ArrayListIterator::add (const E &e **DECAF_UNUSED**) [inline, virtual]

6.59.2.2 template<typename E> virtual bool decaf::util::concurrent::CopyOnWriteArrayList< E >::ArrayListIterator::hasNext () const [inline, virtual]

Returns true if the iteration has more elements.

Returns false if the next call to next would result in an **NoSuchElementException** (p. 1984) to be thrown.

Returns

true if there are more elements available for iteration.

Implements **decaf::util::Iterator< E >** (p. 1559).

6.59.2.3 `template<typename E> virtual bool decaf::util::concurrent::CopyOnWrite-
ArrayList< E >::ArrayListIterator::hasPrevious () const [inline,
virtual]`

Returns true if this list iterator has more elements when traversing the list in the reverse direction.

(In other words, returns true if previous would return an element rather than throwing an exception.)

Returns

true if the list iterator has more elements when traversing the list in the reverse direction.

Implements **decaf::util::ListIterator< E >** (p. 1672).

6.59.2.4 `template<typename E> virtual E decaf::util::concurrent::CopyOnWrite-
ArrayList< E >::ArrayListIterator::next () [inline,
virtual]`

Returns the next element in the iteration.

Calling this method repeatedly until the **hasNext()** (p. 459) method returns false will return each element in the underlying collection exactly once.

Returns

the next element in the iteration of elements.

Exceptions

NoSuchElement- Exception (p. 1984)	if the iteration has no more elements.
--	--

Implements **decaf::util::Iterator< E >** (p. 1560).

6.59.2.5 `template<typename E> virtual int decaf::util::concurrent::CopyOnWrite-
ArrayList< E >::ArrayListIterator::nextIndex () const [inline,
virtual]`

Returns the index of the element that would be returned by a subsequent call to next.

(Returns list size if the list iterator is at the end of the list.)

Returns

the index of the element that would be returned by a subsequent call to next, or list size if list iterator is at end of list.

Implements **decaf::util::ListIterator< E >** (p. 1673).

6.59.2.6 `template<typename E> virtual E decaf::util::concurrent::CopyOnWriteArrayList< E >::ArrayListIterator::previous () [inline, virtual]`

Returns the previous element in the list.

This method may be called repeatedly to iterate through the list backwards, or inter-mixed with calls to next to go back and forth. (Note that alternating calls to next and previous will return the same element repeatedly.)

Returns

the previous element in the list.

Exceptions

<i>NoSuchElementException</i> (p. 1984)	if the iteration has no previous element.
---	---

Implements **decaf::util::ListIterator< E >** (p. 1673).

6.59.2.7 `template<typename E> virtual int decaf::util::concurrent::CopyOnWriteArrayList< E >::ArrayListIterator::previousIndex () const [inline, virtual]`

Returns the index of the element that would be returned by a subsequent call to previous.

(Returns -1 if the list iterator is at the beginning of the list.)

Returns

the index of the element that would be returned by a subsequent call to previous, or -1 if list iterator is at beginning of list.

Implements **decaf::util::ListIterator< E >** (p. 1673).

6.59.2.8 `template<typename E> virtual void decaf::util::concurrent::CopyOnWriteArrayList< E >::ArrayListIterator::remove () [inline, virtual]`

Removes from the underlying collection the last element returned by the iterator (optional operation).

This method can be called only once per call to next. The behavior of an iterator is unspecified if the underlying collection is modified while the iteration is in progress in any way other than by calling this method.

Exceptions

<i>Unsupported-OperationException</i>	if the remove operation is not supported by this Iterator (p. 1559).
<i>IllegalStateException</i>	if the next method has not yet been called, or the remove method has already been called after the last call to the next method.

Implements **decaf::util::Iterator**< **E** > (p. 1560).

6.59.2.9 `template<typename E> virtual void decaf::util::concurrent::CopyOnWriteArrayList< E >::ArrayListIterator::set (const E &e DECAF_UNUSED)`
`[inline, virtual]`

The documentation for this class was generated from the following file:

- `src/main/decaf/util/concurrent/CopyOnWriteArrayList.h`

6.60 **decaf::lang::ArrayPointer**< **T**, **REFCOUNTER** > **Class** - Template Reference

Decaf's implementation of a Smart **Pointer** (p. 2083) that is a template on a Type and is **Thread** (p. 2703) Safe if the default Reference Counter is used.

```
#include <src/main/decaf/lang/ArrayPointer.h>
```

Data Structures

- struct **ArrayData**

Public Types

- typedef **T** * **PointerType**
- typedef **T** & **ReferenceType**
- typedef const **T** & **ConstReferenceType**
- typedef **REFCOUNTER** **CounterType**

Public Member Functions

- **ArrayPointer** ()
Default Constructor.
- **ArrayPointer** (int size)
*Create a new **ArrayPointer** (p. 462) instance and allocates an internal array that is sized using the passed in size value.*
- **ArrayPointer** (int size, const **T** &fillWith)

6.60 decaf::lang::ArrayPointer< T, REFCOUNTER > Class Template Reference 463

Create a new **ArrayPointer** (p. 462) instance and allocates an internal array that is sized using the passed in size value.

- **ArrayPointer** (const **PointerType** value, int size)

Explicit Constructor, creates an **ArrayPointer** (p. 462) that contains value with a single reference.

- **ArrayPointer** (const **ArrayPointer** &value) throw ()

Copy constructor.

- virtual ~**ArrayPointer** () throw ()

- void **reset** (T *value, int size=0)

Resets the **ArrayPointer** (p. 462) to hold the new value.

- T * **release** ()

Releases the **Pointer** (p. 2083) held and resets the internal pointer value to Null.

- **PointerType** **get** () const

Gets the real array pointer that is contained within this **Pointer** (p. 2083).

- int **length** () const

Returns the current size of the contained array or zero if the array is NULL.

- void **swap** (**ArrayPointer** &value) throw ()

Exception (p. 1279) Safe Swap Function.

- **ArrayPointer** **clone** () const

Creates a new **ArrayPointer** (p. 462) instance that is a clone of the value contained in this **ArrayPointer** (p. 462).

- **ArrayPointer** & **operator=** (const **ArrayPointer** &right) throw ()

Assigns the value of right to this **Pointer** (p. 2083) and increments the reference Count.

- template<typename T1 , typename R1 >

ArrayPointer & **operator=** (const **ArrayPointer**< T1, R1 > &right) throw ()

- **ReferenceType** **operator[]** (int index)

Dereference Operator, returns a reference to the Contained value.

- **ConstReferenceType** **operator[]** (int index) const

- bool **operator!** () const

- template<typename T1 , typename R1 >

bool **operator==** (const **ArrayPointer**< T1, R1 > &right) const

- template<typename T1 , typename R1 >

bool **operator!=** (const **ArrayPointer**< T1, R1 > &right) const

Friends

- bool **operator==** (const **ArrayPointer** &left, const T *right)
- bool **operator==** (const T *left, const **ArrayPointer** &right)
- bool **operator!=** (const **ArrayPointer** &left, const T *right)
- bool **operator!=** (const T *left, const **ArrayPointer** &right)

6.60.1 Detailed Description

```
template<typename T, typename REFCOUNTER = decaf::util::concurrent::atomic::AtomicRef-
Counter>class decaf::lang::ArrayPointer< T, REFCOUNTER >
```

Decaf's implementation of a Smart **Pointer** (p. 2083) that is a template on a Type and is **Thread** (p. 2703) Safe if the default Reference Counter is used.

This **Pointer** (p. 2083) type allows for the substitution of different Reference Counter implementations which provide a means of using invasive reference counting if desired using a custom implementation of `ReferenceCounter`.

The Decaf smart pointer provide comparison operators for comparing **Pointer** (p. 2083) instances in the same manner as normal pointer, except that it does not provide an overload of operators (`<`, `<=`, `>`, `>=`). To allow use of a **Pointer** (p. 2083) in a STL container that requires it, **Pointer** (p. 2083) provides an implementation of `std::less`.

Since

1.0

6.60.2 Member Typedef Documentation

6.60.2.1 `template<typename T, typename REFCOUNTER = decaf::util::concurrent::atomic::AtomicRefCounter> typedef const T& decaf::lang::ArrayPointer< T, REFCOUNTER >::ConstReferenceType`

6.60.2.2 `template<typename T, typename REFCOUNTER = decaf::util::concurrent::atomic::AtomicRefCounter> typedef REFCOUNTER decaf::lang::ArrayPointer< T, REFCOUNTER >::CounterType`

6.60.2.3 `template<typename T, typename REFCOUNTER = decaf::util::concurrent::atomic::AtomicRefCounter> typedef T* decaf::lang::ArrayPointer< T, REFCOUNTER >::PointerType`

6.60.2.4 `template<typename T, typename REFCOUNTER = decaf::util::concurrent::atomic::AtomicRefCounter> typedef T& decaf::lang::ArrayPointer< T, REFCOUNTER >::ReferenceType`

6.60.3 Constructor & Destructor Documentation

6.60.3.1 `template<typename T, typename REFCOUNTER = decaf::util::concurrent::atomic::AtomicRefCounter> decaf::lang::ArrayPointer< T, REFCOUNTER >::ArrayPointer () [inline]`

Default Constructor.

Initialized the contained array pointer to NULL, using the subscript operator results in an exception unless reset to contain a real value.

6.60 decaf::lang::ArrayPointer< T, REFCOUNTER > Class Template Reference 465

Referenced by decaf::lang::ArrayPointer< decaf::util::LinkedList< Pointer< MessageDispatch > > >::clone(), and decaf::lang::ArrayPointer< decaf::util::LinkedList< Pointer< MessageDispatch > > >::reset().

```
6.60.3.2  template<typename T, typename REFCOUNTER = decaf::util::concurrent-
           ::atomic::AtomicRefCount> decaf::lang::ArrayPointer< T, REFCOUNTER
           >::ArrayPointer ( int size ) [inline]
```

Create a new **ArrayPointer** (p. 462) instance and allocates an internal array that is sized using the passed in size value.

Parameters

<i>size</i>	The size of the array to allocate for this ArrayPointer (p. 462) instance.
-------------	---

```
6.60.3.3  template<typename T, typename REFCOUNTER = decaf::util::concurrent-
           ::atomic::AtomicRefCount> decaf::lang::ArrayPointer< T, REFCOUNTER
           >::ArrayPointer ( int size, const T & fillWith ) [inline]
```

Create a new **ArrayPointer** (p. 462) instance and allocates an internal array that is sized using the passed in size value.

The array elements are initialized with the given value.

Parameters

<i>size</i>	The size of the array to allocate for this ArrayPointer (p. 462) instance.
<i>fillWith</i>	The value to initialize each element of the newly allocated array with.

```
6.60.3.4  template<typename T, typename REFCOUNTER = decaf::util::concurrent-
           ::atomic::AtomicRefCount> decaf::lang::ArrayPointer< T, REFCOUNTER
           >::ArrayPointer ( const PointerType value, int size ) [inline,
           explicit]
```

Explicit Constructor, creates an **ArrayPointer** (p. 462) that contains value with a single reference.

This object now has ownership until a call to release.

Parameters

<i>value</i>	The pointer to the instance of the array we are taking ownership of.
<i>size</i>	The size of the array this object is taking ownership of.

```
6.60.3.5 template<typename T, typename REFCOUNTER = decaf::util::concurrent-
::atomic::AtomicRefCount> decaf::lang::ArrayPointer< T, REFCOUNTER
>::ArrayPointer ( const ArrayPointer< T, REFCOUNTER > & value ) throw ()
[inline]
```

Copy constructor.

Copies the value contained in the **ArrayPointer** (p. 462) to the new instance and increments the reference counter.

```
6.60.3.6 template<typename T, typename REFCOUNTER = decaf::util::concurrent::atomic::-
AtomicRefCount> virtual decaf::lang::ArrayPointer< T, REFCOUNTER
>::~~ArrayPointer ( ) throw () [inline, virtual]
```

6.60.4 Member Function Documentation

```
6.60.4.1 template<typename T, typename REFCOUNTER = decaf::util::concurrent::atomic::-
AtomicRefCount> ArrayPointer decaf::lang::ArrayPointer< T, REFCOUNTER
>::clone ( ) const [inline]
```

Creates a new **ArrayPointer** (p. 462) instance that is a clone of the value contained in this **ArrayPointer** (p. 462).

Returns

an **ArrayPointer** (p. 462) that contains a copy of the data in this **ArrayPointer** (p. 462).

```
6.60.4.2 template<typename T, typename REFCOUNTER = decaf::util::concurrent::atomic::-
AtomicRefCount> PointerType decaf::lang::ArrayPointer< T, REFCOUNTER
>::get ( ) const [inline]
```

Gets the real array pointer that is contained within this **Pointer** (p. 2083).

This is not really safe since the caller could delete or alter the pointer but it mimics the STL `auto_ptr` and gives access in cases where the caller absolutely needs the real **Pointer** (p. 2083). Use at your own risk.

Returns

the contained pointer.

Referenced by `decaf::util::concurrent::CopyOnWriteArrayList< ServiceListener * >::add()`, `decaf::util::concurrent::CopyOnWriteArrayList< ServiceListener * >::addAll()`, `decaf::util::concurrent::CopyOnWriteArrayList< ServiceListener * >::addAllAbsent()`, `decaf::util::concurrent::CopyOnWriteArrayList< ServiceListener * >::addIfAbsent()`, `decaf::lang::ArrayPointer< decaf::util::LinkedList< Pointer< MessageDispatch > > >::clone()`, `decaf::lang::ArrayPointerComparator< T, R >::compare()`, `decaf::lang::ArrayPointer< decaf::util::LinkedList< Pointer< MessageDispatch > > >::operator!=()`,

6.60 decaf::lang::ArrayPointer< T, REFCOUNTER > Class Template Reference 467

decaf::lang::operator!=(), decaf::lang::ArrayPointerComparator< T, R >::operator>(), std::less< decaf::lang::ArrayPointer< T > >::operator>(), decaf::lang::ArrayPointer< decaf::util::LinkedList< Pointer< MessageDispatch > > >::operator==(), decaf::lang::operator==(), decaf::util::concurrent::CopyOnWriteArrayList< ServiceListener * >::removeAll(), decaf::util::concurrent::CopyOnWriteArrayList< ServiceListener * >::removeAt(), decaf::util::concurrent::CopyOnWriteArrayList< ServiceListener * >::retainAll(), and decaf::util::concurrent::CopyOnWriteArrayList< ServiceListener * >::set().

6.60.4.3 `template<typename T, typename REFCOUNTER = decaf::util::concurrent::atomic::AtomicRefCounter> int decaf::lang::ArrayPointer< T, REFCOUNTER >::length () const [inline]`

Returns the current size of the contained array or zero if the array is NULL.

Returns

the size of the array or zero if the array is NULL

Referenced by decaf::util::concurrent::CopyOnWriteArrayList< ServiceListener * >::add(), decaf::util::concurrent::CopyOnWriteArrayList< ServiceListener * >::addAll(), decaf::util::concurrent::CopyOnWriteArrayList< ServiceListener * >::addAllAbsent(), decaf::util::concurrent::CopyOnWriteArrayList< ServiceListener * >::addIfAbsent(), decaf::util::concurrent::CopyOnWriteArrayList< E >::ArrayListIterator::ArrayListIterator(), decaf::util::concurrent::CopyOnWriteArrayList< ServiceListener * >::contains(), decaf::util::concurrent::CopyOnWriteArrayList< ServiceListener * >::get(), decaf::util::concurrent::CopyOnWriteArrayList< ServiceListener * >::indexOf(), decaf::util::concurrent::CopyOnWriteArrayList< ServiceListener * >::lastIndexOf(), decaf::util::concurrent::CopyOnWriteArrayList< ServiceListener * >::removeAll(), decaf::util::concurrent::CopyOnWriteArrayList< ServiceListener * >::removeAt(), decaf::util::concurrent::CopyOnWriteArrayList< ServiceListener * >::retainAll(), decaf::util::concurrent::CopyOnWriteArrayList< ServiceListener * >::set(), and decaf::util::concurrent::CopyOnWriteArrayList< ServiceListener * >::toArray().

6.60.4.4 `template<typename T, typename REFCOUNTER = decaf::util::concurrent::atomic::AtomicRefCounter> bool decaf::lang::ArrayPointer< T, REFCOUNTER >::operator! () const [inline]`

6.60.4.5 `template<typename T, typename REFCOUNTER = decaf::util::concurrent::atomic::AtomicRefCounter> template<typename T1, typename R1> bool decaf::lang::ArrayPointer< T, REFCOUNTER >::operator!= (const ArrayPointer< T1, R1 > & right) const [inline]`

6.60.4.6 `template<typename T, typename REFCOUNTER = decaf::util::concurrent::atomic::AtomicRefCounter> ArrayPointer& decaf::lang::ArrayPointer< T, REFCOUNTER >::operator= (const ArrayPointer< T, REFCOUNTER > & right) throw () [inline]`

Assigns the value of right to this **Pointer** (p. 2083) and increments the reference Count.

Parameters

<i>right</i>	- Pointer (p. 2083) on the right hand side of an operator= call to this.
--------------	---

6.60.4.7 `template<typename T, typename REFCOUNTER = decaf::util::concurrent::atomic::AtomicRefCount> template<typename T1, typename R1 > ArrayPointer& decaf::lang::ArrayPointer< T, REFCOUNTER >::operator= (const ArrayPointer< T1, R1 > & right) throw ()` [inline]

6.60.4.8 `template<typename T, typename REFCOUNTER = decaf::util::concurrent::atomic::AtomicRefCount> template<typename T1, typename R1 > bool decaf::lang::ArrayPointer< T, REFCOUNTER >::operator== (const ArrayPointer< T1, R1 > & right) const` [inline]

6.60.4.9 `template<typename T, typename REFCOUNTER = decaf::util::concurrent::atomic::AtomicRefCount> ReferenceType decaf::lang::ArrayPointer< T, REFCOUNTER >::operator[] (int index)` [inline]

Dereference Operator, returns a reference to the Contained value.

This method throws an `NullPointerException` if the contained value is NULL.

Returns

reference to the contained pointer.

Exceptions

<i>NullPointerException</i>	if the contained value is Null
-----------------------------	--------------------------------

6.60.4.10 `template<typename T, typename REFCOUNTER = decaf::util::concurrent::atomic::AtomicRefCount> ConstReferenceType decaf::lang::ArrayPointer< T, REFCOUNTER >::operator[] (int index) const` [inline]

6.60.4.11 `template<typename T, typename REFCOUNTER = decaf::util::concurrent::atomic::AtomicRefCount> T* decaf::lang::ArrayPointer< T, REFCOUNTER >::release ()` [inline]

Releases the **Pointer** (p. 2083) held and resets the internal pointer value to Null.

This method is not guaranteed to be safe if the **Pointer** (p. 2083) is held by more than one object or this method is called from more than one thread.

Parameters

<i>value</i>	- The new value to contain.
--------------	-----------------------------

6.60 decaf::lang::ArrayPointer< T, REFCOUNTER > Class Template Reference 469

Returns

The pointer instance that was held by this **Pointer** (p. 2083) object, the pointer is no longer owned by this **Pointer** (p. 2083) and won't be freed when this **Pointer** (p. 2083) goes out of scope.

Referenced by decaf::lang::ArrayPointer< decaf::util::LinkedList< Pointer< MessageDispatch > > >::ArrayPointer().

6.60.4.12 `template<typename T, typename REFCOUNTER = decaf::util::concurrent::atomic::AtomicRefCounter> void decaf::lang::ArrayPointer< T, REFCOUNTER >::reset (T * value, int size = 0) [inline]`

Resets the **ArrayPointer** (p. 462) to hold the new value.

Before the new value is stored reset checks if the old value should be destroyed and if so calls delete. Call reset with a value of NULL is supported and acts to set this **Pointer** (p. 2083) to a NULL pointer.

Parameters

<i>value</i>	The new array pointer value to contain.
<i>size</i>	The size of the new array value this object now contains.

6.60.4.13 `template<typename T, typename REFCOUNTER = decaf::util::concurrent::atomic::AtomicRefCounter> void decaf::lang::ArrayPointer< T, REFCOUNTER >::swap (ArrayPointer< T, REFCOUNTER > & value) throw () [inline]`

Exception (p. 1279) Safe Swap Function.

Parameters

<i>value</i>	- the value to swap with this.
--------------	--------------------------------

Referenced by decaf::lang::ArrayPointer< decaf::util::LinkedList< Pointer< MessageDispatch > > >::operator=(), and decaf::lang::ArrayPointer< decaf::util::LinkedList< Pointer< MessageDispatch > > >::swap().

6.60.5 Friends And Related Function Documentation

6.60.5.1 `template<typename T, typename REFCOUNTER = decaf::util::concurrent::atomic::AtomicRefCounter> bool operator!= (const ArrayPointer< T, REFCOUNTER > & left, const T * right) [friend]`

6.60.5.2 `template<typename T, typename REFCOUNTER = decaf::util::concurrent::atomic::AtomicRefCounter> bool operator!= (const T * left, const ArrayPointer< T, REFCOUNTER > & right) [friend]`

```
6.60.5.3  template<typename T, typename REFCOUNTER = decaf::util::concurrent::atomic::-
AtomicRefCount> bool operator==( const ArrayPointer< T, REFCOUNTER > &
left, const T * right )  [friend]
```

```
6.60.5.4  template<typename T, typename REFCOUNTER = decaf::util::concurrent::atomic::-
AtomicRefCount> bool operator==( const T * left, const ArrayPointer< T,
REFCOUNTER > & right )  [friend]
```

The documentation for this class was generated from the following file:

- src/main/decaf/lang/**ArrayPointer.h**

6.61 decaf::lang::ArrayPointerComparator< T, R > Class - Template Reference

This implementation of Comparator is designed to allows objects in a Collection to be sorted or tested for equality based on the value of the value of the actual pointer to the array being contained in this **ArrayPointer** (p. 462).

```
#include <src/main/decaf/lang/ArrayPointer.h>
```

Inheritance diagram for decaf::lang::ArrayPointerComparator< T, R >:

Public Member Functions

- virtual **~ArrayPointerComparator** ()
- virtual bool **operator()** (const **ArrayPointer**< T, R > &left, const **ArrayPointer**< T, R > &right) const
- virtual int **compare** (const **ArrayPointer**< T, R > &left, const **ArrayPointer**< T, R > &right) const

6.61.1 Detailed Description

```
template<typename T, typename R = decaf::util::concurrent::atomic::AtomicRefCount>class
decaf::lang::ArrayPointerComparator< T, R >
```

This implementation of Comparator is designed to allows objects in a Collection to be sorted or tested for equality based on the value of the value of the actual pointer to the array being contained in this **ArrayPointer** (p. 462).

This allows for a basic ordering to be acheived in Decaf containers.

Custom implementations are possible where an array of some type has a logical natural ordering such as array of integers where the sum of all ints in the array is used.

6.61.2 Constructor & Destructor Documentation

6.61.2.1 `template<typename T , typename R = decaf::util::concurrent::atomic::AtomicRefCounter> virtual decaf::lang::ArrayPointerComparator< T, R >::~~ArrayPointerComparator () [inline, virtual]`

6.61.3 Member Function Documentation

6.61.3.1 `template<typename T , typename R = decaf::util::concurrent::atomic::AtomicRefCounter> virtual int decaf::lang::ArrayPointerComparator< T, R >::compare (const ArrayPointer< T, R > & left, const ArrayPointer< T, R > & right) const [inline, virtual]`

References `decaf::lang::ArrayPointer< T, REFCOUNTER >::get()`.

6.61.3.2 `template<typename T , typename R = decaf::util::concurrent::atomic::AtomicRefCounter> virtual bool decaf::lang::ArrayPointerComparator< T, R >::operator() (const ArrayPointer< T, R > & left, const ArrayPointer< T, R > & right) const [inline, virtual]`

References `decaf::lang::ArrayPointer< T, REFCOUNTER >::get()`.

The documentation for this class was generated from the following file:

- `src/main/decaf/lang/ArrayPointer.h`

6.62 decaf::util::Arrays Class Reference

```
#include <src/main/decaf/util/Arrays.h>
```

Public Member Functions

- `virtual ~Arrays ()`

Static Public Member Functions

- `template<typename E > static void fill (E *array, int size, const E &value)`
Fills an array with the specified element.
- `template<typename E > static void fill (E *array, int size, int start, int end, const E &value)`
Fills an array with the specified element within the range given.

6.62.1 Constructor & Destructor Documentation

6.62.1.1 virtual `decaf::util::Arrays::~~Arrays ()` [virtual]

6.62.2 Member Function Documentation

6.62.2.1 `template<typename E > static void decaf::util::Arrays::fill (E * array, int size, const E & value)` [inline, static]

Fills an array with the specified element.

Parameters

<i>array</i>	The Array to fill.
<i>size</i>	The actual size of the array passed.
<i>value</i>	The value to fill the array with.

Exceptions

<i>NullPointerException</i>	if array is Null.
<i>IllegalArgumentException</i>	if the size parameter is negative, or the start index is greater than the end index.

Referenced by `decaf::lang::ArrayPointer< decaf::util::LinkedList< Pointer< Message-Dispatch > > >::ArrayPointer()`.

6.62.2.2 `template<typename E > static void decaf::util::Arrays::fill (E * array, int size, int start, int end, const E & value)` [inline, static]

Fills an array with the specified element within the range given.

Parameters

<i>array</i>	The Array to fill.
<i>size</i>	The actual size of the array passed.
<i>start</i>	The index to start the fill from (inclusive).
<i>end</i>	The index to fill to (exclusive).
<i>value</i>	The value to fill the array with.

Exceptions

<i>NullPointerException</i>	if array is Null.
<i>IllegalArgumentException</i>	if the size parameter is negative, or the start index is greater than the end index.
<i>IndexOutOfBoundsException</i>	if the start index is negative or the end index is greater than the size parameter.

The documentation for this class was generated from the following file:

- src/main/decaf/util/**Arrays.h**

6.63 decaf::util::concurrent::atomic::AtomicBoolean Class - Reference

A boolean value that may be updated atomically.

```
#include <src/main/decaf/util/concurrent/atomic/AtomicBoolean.h>
```

Public Member Functions

- **AtomicBoolean** ()
*Creates a new **AtomicBoolean** (p. 473) whose initial value is false.*
- **AtomicBoolean** (bool initialValue)
*Creates a new **AtomicBoolean** (p. 473) with the initial value.*
- virtual ~**AtomicBoolean** ()
- bool **get** () const
*Gets the current value of this **AtomicBoolean** (p. 473).*
- void **set** (bool newValue)
Unconditionally sets to the given value.
- bool **compareAndSet** (bool expect, bool update)
Atomically sets the value to the given updated value if the current value == the expected value.
- bool **getAndSet** (bool newValue)
Atomically sets to the given value and returns the previous value.
- std::string **toString** () const
Returns the String representation of the current value.

6.63.1 Detailed Description

A boolean value that may be updated atomically.

An **AtomicBoolean** (p. 473) is used in applications such as atomically updated flags, and cannot be used as a replacement for a Boolean.

6.63.2 Constructor & Destructor Documentation

6.63.2.1 decaf::util::concurrent::atomic::AtomicBoolean::AtomicBoolean ()

Creates a new **AtomicBoolean** (p. 473) whose initial value is false.

6.63.2.2 **decaf::util::concurrent::atomic::AtomicBoolean::AtomicBoolean** (*bool initialValue*)

Creates a new **AtomicBoolean** (p. 473) with the initial value.

Parameters

<i>initialValue</i>	- The initial value of this boolean.
---------------------	--------------------------------------

6.63.2.3 **virtual decaf::util::concurrent::atomic::AtomicBoolean::~~AtomicBoolean** () [*inline, virtual*]

6.63.3 Member Function Documentation

6.63.3.1 **bool decaf::util::concurrent::atomic::AtomicBoolean::compareAndSet** (*bool expect, bool update*)

Atomically sets the value to the given updated value if the current value == the expected value.

Parameters

<i>expect</i>	- the expected value
<i>update</i>	- the new value

Returns

true if successful. False return indicates that the actual value was not equal to the expected value.

6.63.3.2 **bool decaf::util::concurrent::atomic::AtomicBoolean::get** () *const* [*inline*]

Gets the current value of this **AtomicBoolean** (p. 473).

Returns

the currently set value.

6.63.3.3 **bool decaf::util::concurrent::atomic::AtomicBoolean::getAndSet** (*bool newValue*)

Atomically sets to the given value and returns the previous value.

Parameters

<i>newValue</i>	- the new value
-----------------	-----------------

Returns

the previous value

6.63.3.4 `void decaf::util::concurrent::atomic::AtomicBoolean::set (bool newValue)`
`[inline]`

Unconditionally sets to the given value.

Parameters

<i>newValue</i>	- the new value
-----------------	-----------------

6.63.3.5 `std::string decaf::util::concurrent::atomic::AtomicBoolean::toString ()`
`const`

Returns the String representation of the current value.

Returns

the String representation of the current value.

The documentation for this class was generated from the following file:

- `src/main/decaf/util/concurrent/atomic/AtomicBoolean.h`

6.64 decaf::util::concurrent::atomic::AtomicInteger Class Reference

An int value that may be updated atomically.

```
#include <src/main/decaf/util/concurrent/atomic/AtomicInteger.h>
```

Inheritance diagram for `decaf::util::concurrent::atomic::AtomicInteger`:

Public Member Functions

- **AtomicInteger ()**
*Create a new **AtomicInteger** (p. 475) with an initial value of 0.*
- **AtomicInteger (int initialValue)**
*Create a new **AtomicInteger** (p. 475) with the given initial value.*
- `virtual ~AtomicInteger ()`
- `int get () const`

Gets the current value.

- void **set** (int newValue)

Sets to the given value.

- int **getAndSet** (int newValue)

Atomically sets to the given value and returns the old value.

- bool **compareAndSet** (int expect, int update)

Atomically sets the value to the given updated value if the current value == the expected value.

- int **getAndIncrement** ()

Atomically increments by one the current value.

- int **getAndDecrement** ()

Atomically decrements by one the current value.

- int **getAndAdd** (int delta)

Atomically adds the given value to the current value.

- int **incrementAndGet** ()

Atomically increments by one the current value.

- int **decrementAndGet** ()

Atomically decrements by one the current value.

- int **addAndGet** (int delta)

Atomically adds the given value to the current value.

- std::string **toString** () const

Returns the String representation of the current value.

- int **intValue** () const

Description copied from class: Number Returns the value of the specified number as an int.

- long long **longValue** () const

Description copied from class: Number Returns the value of the specified number as a long.

- float **floatValue** () const

Description copied from class: Number Returns the value of the specified number as a float.

- double **doubleValue** () const

Description copied from class: Number Returns the value of the specified number as a double.

6.64.1 Detailed Description

An int value that may be updated atomically.

An **AtomicInteger** (p. 475) is used in applications such as atomically incremented counters, and cannot be used as a replacement for an Integer. However, this class does extend Number to allow uniform access by tools and utilities that deal with numerically-based classes.

6.64.2 Constructor & Destructor Documentation

6.64.2.1 decaf::util::concurrent::atomic::AtomicInteger::AtomicInteger ()

Create a new **AtomicInteger** (p. 475) with an initial value of 0.

6.64.2.2 decaf::util::concurrent::atomic::AtomicInteger::AtomicInteger (int *initialValue*)

Create a new **AtomicInteger** (p. 475) with the given initial value.

Parameters

<i>initialValue</i>	- The initial value of this object.
---------------------	-------------------------------------

6.64.2.3 virtual decaf::util::concurrent::atomic::AtomicInteger::~~AtomicInteger () [inline, virtual]

6.64.3 Member Function Documentation

6.64.3.1 int decaf::util::concurrent::atomic::AtomicInteger::addAndGet (int *delta*)

Atomically adds the given value to the current value.

Parameters

<i>delta</i>	- the value to add.
--------------	---------------------

Returns

the updated value.

6.64.3.2 bool decaf::util::concurrent::atomic::AtomicInteger::compareAndSet (int *expect*, int *update*)

Atomically sets the value to the given updated value if the current value == the expected value.

Parameters

<i>expect</i>	- the expected value
<i>update</i>	- the new value

Returns

true if successful. False return indicates that the actual value was not equal to the expected value.

6.64.3.3 int decaf::util::concurrent::atomic::AtomicInteger::decrementAndGet ()

Atomically decrements by one the current value.

Returns

the updated value.

Referenced by `decaf::util::concurrent::atomic::AtomicRefCounter::release()`.

6.64.3.4 double decaf::util::concurrent::atomic::AtomicInteger::doubleValue () const [virtual]

Description copied from class: Number Returns the value of the specified number as a double.

This may involve rounding.

Returns

the numeric value represented by this object after conversion to type double.

Implements **decaf::lang::Number** (p. 1993).

6.64.3.5 float decaf::util::concurrent::atomic::AtomicInteger::floatValue () const [virtual]

Description copied from class: Number Returns the value of the specified number as a float.

This may involve rounding.

Returns

the numeric value represented by this object after conversion to type float.

Implements **decaf::lang::Number** (p. 1993).

6.64.3.6 int decaf::util::concurrent::atomic::AtomicInteger::get () const [inline]

Gets the current value.

Returns

the current value.

Referenced by `decaf::util::concurrent::LinkedBlockingQueue< E >::drainTo()`, `decaf::util::concurrent::LinkedBlockingQueue< E >::offer()`, `decaf::util::concurrent::LinkedBlockingQueue< E >::peek()`, `decaf::util::concurrent::LinkedBlockingQueue< E >::poll()`, `decaf::util::concurrent::LinkedBlockingQueue< E >::put()`, `decaf::util::concurrent::LinkedBlockingQueue< E >::remainingCapacity()`, `decaf::util::concurrent::LinkedBlockingQueue< E >::size()`, `decaf::util::concurrent::LinkedBlockingQueue< E >::take()`, `decaf::util::concurrent::LinkedBlockingQueue< E >::toArray()`, and `decaf::util::concurrent::LinkedBlockingQueue< E >::toString()`.

6.64.3.7 int decaf::util::concurrent::atomic::AtomicInteger::getAndAdd (int *delta*)

Atomically adds the given value to the current value.

Parameters

<i>delta</i>	- The value to add.
--------------	---------------------

Returns

the previous value.

Referenced by `decaf::util::concurrent::LinkedBlockingQueue< E >::drainTo()`.

6.64.3.8 int decaf::util::concurrent::atomic::AtomicInteger::getAndDecrement ()

Atomically decrements by one the current value.

Returns

the previous value.

Referenced by `decaf::util::concurrent::LinkedBlockingQueue< E >::poll()`, and `decaf::util::concurrent::LinkedBlockingQueue< E >::take()`.

6.64.3.9 int decaf::util::concurrent::atomic::AtomicInteger::getAndIncrement ()

Atomically increments by one the current value.

Returns

the previous value.

Referenced by `decaf::util::concurrent::LinkedBlockingQueue< E >::offer()`, and `decaf::util::concurrent::LinkedBlockingQueue< E >::put()`.

6.64.3.10 `int decaf::util::concurrent::atomic::AtomicInteger::getAndSet (int
newValue)`

Atomically sets to the given value and returns the old value.

Parameters

<i>newValue</i>	- the new value.
-----------------	------------------

Returns

the previous value.

Referenced by `decaf::util::concurrent::LinkedBlockingQueue< E >::clear()`.

6.64.3.11 `int decaf::util::concurrent::atomic::AtomicInteger::incrementAndGet ()`

Atomically increments by one the current value.

Returns

the updated value.

Referenced by `decaf::util::concurrent::atomic::AtomicRefCounter::AtomicRefCounter()`.

6.64.3.12 `int decaf::util::concurrent::atomic::AtomicInteger::intValue () const
[virtual]`

Description copied from class: `Number` Returns the value of the specified number as an int.

This may involve rounding or truncation.

Returns

the numeric value represented by this object after conversion to type int.

Implements **`decaf::lang::Number`** (p. 1993).

6.64.3.13 `long long decaf::util::concurrent::atomic::AtomicInteger::longValue ()
const [virtual]`

Description copied from class: `Number` Returns the value of the specified number as a long.

This may involve rounding or truncation.

Returns

the numeric value represented by this object after conversion to type long long.

Implements **`decaf::lang::Number`** (p. 1994).

6.64.3.14 `void decaf::util::concurrent::atomic::AtomicInteger::set (int newValue)`
`[inline]`

Sets to the given value.

Parameters

<i>newValue</i>	- the new value
-----------------	-----------------

Referenced by `decaf::util::concurrent::LinkedBlockingQueue< E >::clear()`.

6.64.3.15 `std::string decaf::util::concurrent::atomic::AtomicInteger::toString ()`
`const`

Returns the String representation of the current value.

Returns

the String representation of the current value.

The documentation for this class was generated from the following file:

- `src/main/decaf/util/concurrent/atomic/AtomicInteger.h`

6.65 decaf::util::concurrent::atomic::AtomicRefCounter Class - Reference

```
#include <src/main/decaf/util/concurrent/atomic/Atomic-
RefCounter.h>
```

Inherited by `decaf::lang::ArrayPointer< decaf::util::LinkedList< Pointer< MessageDispatch > >, decaf::lang::ArrayPointer< E >, decaf::lang::ArrayPointer< ServiceListener * >, decaf::lang::ArrayPointer< unsigned char >, decaf::lang::Pointer< activemq::threads::TaskRunner >, decaf::lang::Pointer< ActiveMQDestination >, decaf::lang::Pointer< ActiveMQTransactionContext >, decaf::lang::Pointer< BackupTransportPool >, decaf::lang::Pointer< BooleanExpression >, decaf::lang::Pointer< BrokerError >, decaf::lang::Pointer< BrokerId >, decaf::lang::Pointer< ByteArrayAdapter >, decaf::lang::Pointer< CloseTransportsTask >, decaf::lang::Pointer< cms::Destination >, decaf::lang::Pointer< commands::ActiveMQDestination >, decaf::lang::Pointer< commands::ConsumerId >, decaf::lang::Pointer< commands::ConsumerInfo >, decaf::lang::Pointer< commands::Message >, decaf::lang::Pointer< commands::ProducerInfo >, decaf::lang::Pointer< commands::SessionInfo >, decaf::lang::Pointer< commands::WireFormatInfo >, decaf::lang::Pointer< Comparator< E > >, decaf::lang::Pointer< CompositeTaskRunner >, decaf::lang::Pointer< ConnectionId >, decaf::lang::Pointer< ConnectionInfo >, decaf::lang::Pointer< ConsumerId >, decaf::lang::Pointer< ConsumerInfo >, decaf::lang::Pointer<`

core::ActiveMQAckHandler >, decaf::lang::Pointer< DataStructure >, decaf::lang::Pointer< decaf::lang::Runnable >, decaf::lang::Pointer< decaf::lang::Thread >, decaf::lang::Pointer< Exception >, decaf::lang::Pointer< LogManagerInternals >, decaf::lang::Pointer< Message >, decaf::lang::Pointer< MessageAck >, decaf::lang::Pointer< MessageDispatchChannel >, decaf::lang::Pointer< Messageld >, decaf::lang::Pointer< ProducerId >, decaf::lang::Pointer< ProducerInfo >, decaf::lang::Pointer< Properties >, decaf::lang::Pointer< QueueNode< E > >, decaf::lang::Pointer< Response >, decaf::lang::Pointer< ResponseBuilder >, decaf::lang::Pointer< SessionId >, decaf::lang::Pointer< SessionInfo >, decaf::lang::Pointer< Tracked >, decaf::lang::Pointer< TransactionId >, decaf::lang::Pointer< TransactionState >, decaf::lang::Pointer< Transport >, decaf::lang::Pointer< TransportListener >, decaf::lang::Pointer< URI >, decaf::lang::Pointer< URIPool >, and decaf::lang::Pointer< wireformat::WireFormat >.

Public Member Functions

- **AtomicRefCounter** ()
- **AtomicRefCounter** (const **AtomicRefCounter** &other)
- virtual ~**AtomicRefCounter** ()

Protected Member Functions

- void **swap** (**AtomicRefCounter** &other)
Swaps this instance's reference counter with the one given, this allows for copy-and-swap semantics of this object.
- bool **release** ()
Removes a reference to the counter Atomically and returns if the counter has reached zero, once the counter hits zero, the internal counter is destroyed and this instance is now considered to be unreferenced.

6.65.1 Constructor & Destructor Documentation

6.65.1.1 **decaf::util::concurrent::atomic::AtomicRefCounter::AtomicRefCounter** (
) [inline]

6.65.1.2 **decaf::util::concurrent::atomic::AtomicRefCounter::AtomicRefCounter** (
const **AtomicRefCounter** & other) [inline]

References decaf::util::concurrent::atomic::AtomicInteger::incrementAndGet().

6.65.1.3 virtual **decaf::util::concurrent::atomic::AtomicRefCounter::~~AtomicRefCounter** () [inline, virtual]

6.65.2 Member Function Documentation

6.65.2.1 `bool decaf::util::concurrent::atomic::AtomicRefCount::release ()`
`[inline, protected]`

Removes a reference to the counter Atomically and returns if the counter has reached zero, once the counter hits zero, the internal counter is destroyed and this instance is now considered to be unreferenced.

Returns

true if the count is now zero.

Reimplemented in `decaf::lang::ArrayPointer< ServiceListener * >` (p. 468), `decaf::lang::ArrayPointer< E >` (p. 468), `decaf::lang::ArrayPointer< unsigned char >` (p. 468), `decaf::lang::ArrayPointer< decaf::util::LinkedList< Pointer< MessageDispatch > > >` (p. 468), `decaf::lang::Pointer< MessageAck >` (p. 2090), `decaf::lang::Pointer< BooleanExpression >` (p. 2090), `decaf::lang::Pointer< commands::ConsumerId >` (p. 2090), `decaf::lang::Pointer< BrokerError >` (p. 2090), `decaf::lang::Pointer< Transport >` (p. 2090), `decaf::lang::Pointer< wireformat::WireFormat >` (p. 2090), `decaf::lang::Pointer< MessageDispatchChannel >` (p. 2090), `decaf::lang::Pointer< commands::WireFormatInfo >` (p. 2090), `decaf::lang::Pointer< CloseTransportsTask >` (p. 2090), `decaf::lang::Pointer< CompositeTaskRunner >` (p. 2090), `decaf::lang::Pointer< ActiveMQTransactionContext >` (p. 2090), `decaf::lang::Pointer< commands::ProducerInfo >` (p. 2090), `decaf::lang::Pointer< Comparator< E > >` (p. 2090), `decaf::lang::Pointer< BrokerId >` (p. 2090), `decaf::lang::Pointer< LogManagerInternals >` (p. 2090), `decaf::lang::Pointer< commands::SessionInfo >` (p. 2090), `decaf::lang::Pointer< Message >` (p. 2090), `decaf::lang::Pointer< URI >` (p. 2090), `decaf::lang::Pointer< DataStructure >` (p. 2090), `decaf::lang::Pointer< activemq::threads::TaskRunner >` (p. 2090), `decaf::lang::Pointer< commands::ActiveMQDestination >` (p. 2090), `decaf::lang::Pointer< ConsumerInfo >` (p. 2090), `decaf::lang::Pointer< ConnectionId >` (p. 2090), `decaf::lang::Pointer< decaf::lang::Runnable >` (p. 2090), `decaf::lang::Pointer< Properties >` (p. 2090), `decaf::lang::Pointer< BackupTransportPool >` (p. 2090), `decaf::lang::Pointer< ProducerInfo >` (p. 2090), `decaf::lang::Pointer< decaf::lang::Thread >` (p. 2090), `decaf::lang::Pointer< MessageId >` (p. 2090), `decaf::lang::Pointer< Response >` (p. 2090), `decaf::lang::Pointer< QueueNode< E > >` (p. 2090), `decaf::lang::Pointer< SessionId >` (p. 2090), `decaf::lang::Pointer< cms::Destination >` (p. 2090), `decaf::lang::Pointer< TransportListener >` (p. 2090), `decaf::lang::Pointer< ActiveMQDestination >` (p. 2090), `decaf::lang::Pointer< ProducerId >` (p. 2090), `decaf::lang::Pointer< ResponseBuilder >` (p. 2090), `decaf::lang::Pointer< SessionInfo >` (p. 2090), `decaf::lang::Pointer< commands::Message >` (p. 2090), `decaf::lang::Pointer< Tracked >` (p. 2090), `decaf::lang::Pointer< ConnectionInfo >` (p. 2090), `decaf::lang::Pointer< core::ActiveMQAckHandler >` (p. 2090), `decaf::lang::Pointer< Exception >` (p. 2090), `decaf::lang::Pointer< TransactionState >` (p. 2090), `decaf::lang::Pointer< commands::ConsumerInfo >` (p. 2090), `decaf::lang::Pointer< ConsumerId >` (p. 2090), `decaf::lang::Pointer< URIPool >` (p. 2090), `decaf::lang::Pointer< ByteArrayAdapter >` (p. 2090), and `decaf::lang::Pointer< TransactionId >` (p. 2090).

References `decaf::util::concurrent::atomic::AtomicInteger::decrementAndGet()`.

6.65.2.2 `void decaf::util::concurrent::atomic::AtomicRefCounter::swap (AtomicRefCounter & other) [inline, protected]`

Swaps this instance's reference counter with the one given, this allows for copy-and-swap semantics of this object.

Parameters

<code>other</code>	The value to swap with this one's.
--------------------	------------------------------------

The documentation for this class was generated from the following file:

- `src/main/decaf/util/concurrent/atomic/AtomicRefCounter.h`

6.66 `decaf::util::concurrent::atomic::AtomicReference< T >` - Class Template Reference

An Pointer reference that may be updated atomically.

```
#include <src/main/decaf/util/concurrent/atomic/AtomicReference.h>
```

Public Member Functions

- **AtomicReference** ()
- **AtomicReference** (T *value)
- virtual **~AtomicReference** ()
- T * **get** () const
Gets the Current Value.
- void **set** (T *newValue)
Sets the Current value of this Reference.
- bool **compareAndSet** (T *expect, T *update)
Atomically sets the value to the given updated value if the current value == the expected value.
- T * **getAndSet** (T *newValue)
Atomically sets to the given value and returns the old value.
- std::string **toString** () const
Returns the String representation of the current value.

6.66.1 Detailed Description

```
template<typename T>class decaf::util::concurrent::atomic::AtomicReference< T >
```

An Pointer reference that may be updated atomically.

6.66.2 Constructor & Destructor Documentation

6.66.2.1 `template<typename T > decaf::util::concurrent::atomic::AtomicReference< T >::AtomicReference () [inline]`

6.66.2.2 `template<typename T > decaf::util::concurrent::atomic::AtomicReference< T >::AtomicReference (T * value) [inline]`

6.66.2.3 `template<typename T > virtual decaf::util::concurrent::atomic::~AtomicReference< T >::~~AtomicReference () [inline, virtual]`

6.66.3 Member Function Documentation

6.66.3.1 `template<typename T > bool decaf::util::concurrent::atomic::~AtomicReference< T >::compareAndSet (T * expect, T * update) [inline]`

Atomically sets the value to the given updated value if the current value == the expected value.

Parameters

<i>expect</i>	- the expected value
<i>update</i>	- the new value

Returns

true if successful. False return indicates that the actual value was not equal to the expected value.

6.66.3.2 `template<typename T > T* decaf::util::concurrent::atomic::AtomicReference< T >::get () const [inline]`

Gets the Current Value.

Returns

the current value of this Reference.

```
6.66.3.3  template<typename T > T* decaf::util::concurrent::atomic-
::AtomicReference< T >::getAndSet ( T * newValue )
[inline]
```

Atomically sets to the given value and returns the old value.

Parameters

<i>newValue</i>	the new value
-----------------	---------------

Returns

the previous value.

```
6.66.3.4  template<typename T > void decaf::util::concurrent-
::atomic::AtomicReference< T >::set ( T * newValue )
[inline]
```

Sets the Current value of this Reference.

Parameters

<i>newValue</i>	The new Value of this Reference.
-----------------	----------------------------------

```
6.66.3.5  template<typename T > std::string decaf::util::concurrent-
::atomic::AtomicReference< T >::toString ( ) const
[inline]
```

Returns the String representation of the current value.

Returns

string representation of the current value.

The documentation for this class was generated from the following file:

- src/main/decaf/util/concurrent/atomic/**AtomicReference.h**

6.67 activemq::transport::failover::BackupTransport Class - Reference

```
#include <src/main/activemq/transport/failover/Backup-
Transport.h>
```

Inheritance diagram for activemq::transport::failover::BackupTransport:

Public Member Functions

- **BackupTransport** (**BackupTransportPool** *failover)
- virtual **~BackupTransport** ()
- **decaf::net::URI** **getUri** () const
Gets the URI assigned to this Backup.
- void **setUri** (const **decaf::net::URI** &uri)
*Sets the URI assigned to this **Transport** (p. 2790).*
- const **Pointer**< **Transport** > & **getTransport** ()
Gets the currently held transport.
- void **setTransport** (const **Pointer**< **Transport** > &transport)
Sets the held transport, if not NULL then start to listen for exceptions from the held transport.
- virtual void **onException** (const **decaf::lang::Exception** &ex)
Event handler for an exception from a command transport.
- bool **isClosed** () const
*Has the **Transport** (p. 2790) been shutdown and no longer usable.*
- void **setClosed** (bool value)
*Sets the closed flag on this **Transport** (p. 2790).*

6.67.1 Constructor & Destructor Documentation

- 6.67.1.1 **activemq::transport::failover::BackupTransport::BackupTransport** (**BackupTransportPool** * failover)
- 6.67.1.2 virtual **activemq::transport::failover::BackupTransport::~~BackupTransport** () [virtual]

6.67.2 Member Function Documentation

- 6.67.2.1 const **Pointer**<**Transport**>& **activemq::transport::failover::BackupTransport::getTransport** () [inline]

Gets the currently held transport.

Returns

pointer to the held transport or NULL if not set.

- 6.67.2.2 **decaf::net::URI** **activemq::transport::failover::BackupTransport::getUri** () const [inline]

Gets the URI assigned to this Backup.

Returns

the assigned URI

6.67.2.3 `bool activemq::transport::failover::BackupTransport::isClosed () const`
`[inline]`

Has the **Transport** (p. 2790) been shutdown and no longer usable.

Returns

true if the **Transport** (p. 2790)

6.67.2.4 `virtual void activemq::transport::failover::BackupTransport::onException (`
`const decaf::lang::Exception & ex) [virtual]`

Event handler for an exception from a command transport.

The **BackupTransport** (p. 486) closes its internal **Transport** (p. 2790) when an exception is received and returns the URI to the pool of URIs to attempt connections to.

Parameters

<i>ex</i>	The exception that was passed to this listener to handle.
-----------	---

Implements **activemq::transport::TransportListener** (p. 2811).

6.67.2.5 `void activemq::transport::failover::BackupTransport::setClosed (bool`
`value) [inline]`

Sets the closed flag on this **Transport** (p. 2790).

Parameters

<i>value</i>	- true for closed.
--------------	--------------------

6.67.2.6 `void activemq::transport::failover::BackupTransport::setTransport (const`
`Pointer< Transport > & transport) [inline]`

Sets the held transport, if not NULL then start to listen for exceptions from the held transport.

Parameters

<i>transport</i>	The transport to hold.
------------------	------------------------

6.67.2.7 `void activemq::transport::failover::BackupTransport::setUri (const`
`decaf::net::URI & uri) [inline]`

Sets the URI assigned to this **Transport** (p. 2790).

The documentation for this class was generated from the following file:

- src/main/activemq/transport/failover/**BackupTransport.h**

6.68 activemq::transport::failover::BackupTransportPool Class - Reference

```
#include <src/main/activemq/transport/failover/Backup-TransportPool.h>
```

Inheritance diagram for activemq::transport::failover::BackupTransportPool:

Public Member Functions

- **BackupTransportPool** (const **Pointer**< **CompositeTaskRunner** > &taskRunner, const **Pointer**< **CloseTransportsTask** > &closeTask, const **Pointer**< **URIPool** > &uriPool)
- **BackupTransportPool** (int backupPoolSize, const **Pointer**< **CompositeTaskRunner** > &taskRunner, const **Pointer**< **CloseTransportsTask** > &closeTask, const **Pointer**< **URIPool** > &uriPool)
- virtual ~**BackupTransportPool** ()
- virtual bool **isPending** () const
Return true if we don't currently have enough Connected Transports.
- **Pointer**< **BackupTransport** > **getBackup** ()
*Get a Connected **Transport** (p. 2790) from the pool of Backups if any are present, otherwise it return a NULL Pointer.*
- virtual bool **iterate** ()
Connect to a Backup Broker if we haven't already connected to the max number of Backups.
- int **getBackupPoolSize** () const
Gets the Max number of Backups this Task will create.
- void **setBackupPoolSize** (int size)
Sets the Max number of Backups this Task will create.
- bool **isEnabled** () const
*Gets if the backup **Transport** (p. 2790) Pool has been enabled or not, when not enabled no backups are created and any that were are destroyed.*
- void **setEnabled** (bool value)
*Sets if this Backup **Transport** (p. 2790) Pool is enabled.*

Friends

- class **BackupTransport**

6.68.1 Constructor & Destructor Documentation

6.68.1.1 `activemq::transport::failover::BackupTransportPool::BackupTransportPool (const Pointer< CompositeTaskRunner > & taskRunner, const Pointer< CloseTransportsTask > & closeTask, const Pointer< URIPool > & uriPool)`

6.68.1.2 `activemq::transport::failover::BackupTransportPool::BackupTransportPool (int backupPoolSize, const Pointer< CompositeTaskRunner > & taskRunner, const Pointer< CloseTransportsTask > & closeTask, const Pointer< URIPool > & uriPool)`

6.68.1.3 `virtual activemq::transport::failover::BackupTransportPool::~~BackupTransportPool () [virtual]`

6.68.2 Member Function Documentation

6.68.2.1 `Pointer<BackupTransport> activemq::transport::failover::BackupTransportPool::getBackup ()`

Get a Connected **Transport** (p. 2790) from the pool of Backups if any are present, otherwise it return a NULL Pointer.

Returns

Pointer to a Connected **Transport** (p. 2790) or NULL

6.68.2.2 `int activemq::transport::failover::BackupTransportPool::getBackupPoolSize () const [inline]`

Gets the Max number of Backups this Task will create.

Returns

the max number of active BackupTransports that will be created.

6.68.2.3 `bool activemq::transport::failover::BackupTransportPool::isEnabled () const [inline]`

Gets if the backup **Transport** (p. 2790) Pool has been enabled or not, when not enabled no backups are created and any that were are destroyed.

Returns

true if enable.

6.68.2.4 `virtual bool activemq::transport::failover::BackupTransportPool::isPending () const [virtual]`

Return true if we don't currently have enough Connected Transports.

Implements **activemq::threads::CompositeTask** (p. 893).

6.68.2.5 `virtual bool activemq::transport::failover::BackupTransportPool::iterate () [virtual]`

Connect to a Backup Broker if we haven't already connected to the max number of Backups.

Implements **activemq::threads::Task** (p. 2676).

6.68.2.6 `void activemq::transport::failover::BackupTransportPool::setBackupPoolSize (int size) [inline]`

Sets the Max number of Backups this Task will create.

Parameters

<i>size</i>	- the max number of active BackupTransports that will be created.
-------------	---

6.68.2.7 `void activemq::transport::failover::BackupTransportPool::setEnabled (bool value)`

Sets if this Backup **Transport** (p. 2790) Pool is enabled.

When not enabled no Backups are created and any that were are destroyed.

Parameters

<i>value</i>	- true to enable backup creation, false to disable.
--------------	---

6.68.3 Friends And Related Function Documentation

6.68.3.1 `friend class BackupTransport [friend]`

The documentation for this class was generated from the following file:

- `src/main/activemq/transport/failover/BackupTransportPool.h`

6.69 activemq::commands::BaseCommand Class Reference

```
#include <src/main/activemq/commands/BaseCommand.h>
```

Inheritance diagram for `activemq::commands::BaseCommand`:

Public Member Functions

- **BaseCommand** ()
- virtual **~BaseCommand** ()
- virtual void **setCommandId** (int id)
*Sets the **Command** (p. 866) Id of this **Message** (p. 1821).*
- virtual int **getCommandId** () const
*Gets the **Command** (p. 866) Id of this **Message** (p. 1821).*
- virtual void **setResponseRequired** (const bool required)
*Set if this **Message** (p. 1821) requires a **Response** (p. 2298).*
- virtual bool **isResponseRequired** () const
*Is a **Response** (p. 2298) required for this **Command** (p. 866).*
- virtual void **copyDataStructure** (const **DataStructure** *src)
Copy the contents of the passed object into this objects members, overwriting any existing data.
- virtual std::string **toString** () const
*Returns a string containing the information for this **DataStructure** (p. 1133) such as its type and value of its elements.*
- virtual bool **equals** (const **DataStructure** *value) const
*Compares the **DataStructure** (p. 1133) passed in to this one, and returns if they are equivalent.*
- virtual bool **isConnectionControl** () const
- virtual bool **isConnectionInfo** () const
- virtual bool **isConsumerInfo** () const
- virtual bool **isBrokerInfo** () const
- virtual bool **isMessage** () const
- virtual bool **isMessageAck** () const
- virtual bool **isKeepAliveInfo** () const
- virtual bool **isMessageDispatch** () const
- virtual bool **isMessageDispatchNotification** () const
- virtual bool **isProducerAck** () const
- virtual bool **isProducerInfo** () const
- virtual bool **isResponse** () const
- virtual bool **isRemoveInfo** () const
- virtual bool **isRemoveSubscriptionInfo** () const
- virtual bool **isShutdownInfo** () const
- virtual bool **isTransactionInfo** () const
- virtual bool **isWireFormatInfo** () const

6.69.1 Constructor & Destructor Documentation

6.69.1.1 `activemq::commands::BaseCommand::BaseCommand ()` `[inline]`

6.69.1.2 `virtual activemq::commands::BaseCommand::~~BaseCommand ()`
`[inline, virtual]`

6.69.2 Member Function Documentation

6.69.2.1 `virtual void activemq::commands::BaseCommand::copyDataStructure (`
`const DataStructure * src)` `[inline, virtual]`

Copy the contents of the passed object into this objects members, overwriting any existing data.

Returns

src - Source Object

Implements `activemq::commands::DataStructure` (p. 1134).

Reimplemented in `activemq::commands::Message` (p. 1827), `activemq::commands::ConsumerInfo` (p. 1012), `activemq::commands::BrokerError` (p. 560), `activemq::commands::ActiveMQBytesMessage` (p. 169), `activemq::commands::BrokerInfo` (p. 574), `activemq::commands::ConnectionInfo` (p. 970), `activemq::commands::MessageAck` (p. 1869), `activemq::commands::ConsumerControl` (p. 993), `activemq::commands::ProducerInfo` (p. 2192), `activemq::commands::ConnectionControl` (p. 940), `activemq::commands::DestinationInfo` (p. 1215), `activemq::commands::MessagePull` (p. 1941), `activemq::commands::SessionInfo` (p. 2388), `activemq::commands::MessageDispatch` (p. 1883), `activemq::commands::MessageDispatchNotification` (p. 1896), `activemq::commands::TransactionInfo` (p. 2775), `activemq::commands::ConnectionError` (p. 949), `activemq::commands::RemoveSubscriptionInfo` (p. 2277), `activemq::commands::ActiveMQStreamMessage` (p. 362), `activemq::commands::ProducerAck` (p. 2173), `activemq::commands::RemoveInfo` (p. 2269), `activemq::commands::DataArrayResponse` (p. 1070), `activemq::commands::DataResponse` (p. 1113), `activemq::commands::ExceptionResponse` (p. 1289), `activemq::commands::ReplayCommand` (p. 2285), `activemq::commands::ControlCommand` (p. 1024), `activemq::commands::IntegerResponse` (p. 1517), `activemq::commands::Response` (p. 2299), `activemq::commands::ActiveMQMapMessage` (p. 272), `activemq::commands::FlushCommand` (p. 1382), `activemq::commands::KeepAliveInfo` (p. 1592), `activemq::commands::ShutdownInfo` (p. 2432), `activemq::commands::ActiveMQBlobMessage` (p. 158), `activemq::commands::WireFormatInfo` (p. 2892), `activemq::commands::ActiveMQTextMessage` (p. 407), `activemq::commands::ActiveMQObjectMessage` (p. 302), and `activemq::commands::ActiveMQMessage` (p. 289).

References `getCommandId()`, and `isResponseRequired()`.

6.69.2.2 `virtual bool activemq::commands::BaseCommand::equals (const
DataStructure * value) const [inline, virtual]`

Compares the **DataStructure** (p. 1133) passed in to this one, and returns if they are equivalent.

Equivalent here means that they are of the same type, and that each element of the objects are the same.

Returns

true if **DataStructure** (p. 1133)'s are Equal.

Implements **activemq::commands::DataStructure** (p. 1135).

Reimplemented in **activemq::commands::Message** (p. 1827), **activemq::commands::ConsumerInfo** (p. 1012), **activemq::commands::ActiveMQBytesMessage** (p. 169), **activemq::commands::BrokerInfo** (p. 574), **activemq::commands::ConnectionInfo** (p. 970), **activemq::commands::MessageAck** (p. 1869), **activemq::commands::ConsumerControl** (p. 994), **activemq::commands::ProducerInfo** (p. 2192), **activemq::commands::ConnectionControl** (p. 940), **activemq::commands::DestinationInfo** (p. 1215), **activemq::commands::MessagePull** (p. 1941), **activemq::commands::SessionInfo** (p. 2388), **activemq::commands::MessageDispatch** (p. 1883), **activemq::commands::MessageDispatchNotification** (p. 1896), **activemq::commands::TransactionInfo** (p. 2776), **activemq::commands::ConnectionError** (p. 949), **activemq::commands::RemoveSubscriptionInfo** (p. 2277), **activemq::commands::ActiveMQStreamMessage** (p. 362), **activemq::commands::ProducerAck** (p. 2173), **activemq::commands::RemoveInfo** (p. 2269), **activemq::commands::ActiveMQMapMessage** (p. 272), **activemq::commands::DataArrayResponse** (p. 1070), **activemq::commands::DataResponse** (p. 1114), **activemq::commands::ExceptionResponse** (p. 1289), **activemq::commands::ReplayCommand** (p. 2285), **activemq::commands::ControlCommand** (p. 1024), **activemq::commands::IntegerResponse** (p. 1518), **activemq::commands::Response** (p. 2299), **activemq::commands::FlushCommand** (p. 1382), **activemq::commands::KeepAliveInfo** (p. 1593), **activemq::commands::ShutdownInfo** (p. 2433), **activemq::commands::ActiveMQMessageTemplate< T >** (p. 297), **activemq::commands::ActiveMQMessageTemplate< cms::BytesMessage >** (p. 297), **activemq::commands::ActiveMQMessageTemplate< cms::MapMessage >** (p. 297), **activemq::commands::ActiveMQMessageTemplate< cms::Message >** (p. 297), **activemq::commands::ActiveMQMessageTemplate< cms::StreamMessage >** (p. 297), **activemq::commands::ActiveMQMessageTemplate< cms::TextMessage >** (p. 297), **activemq::commands::ActiveMQMessageTemplate< cms::ObjectMessage >** (p. 297), **activemq::commands::ActiveMQBlobMessage** (p. 159), **activemq::commands::WireFormatInfo** (p. 2892), **activemq::commands::ActiveMQTextMessage** (p. 408), **activemq::commands::ActiveMQObjectMessage** (p. 303), and **activemq::commands::ActiveMQMessage** (p. 290).

References **activemq::commands::BaseDataStructure::equals()**.

6.69.2.3 `virtual int activemq::commands::BaseCommand::getCommandId () const`
[inline, virtual]

Gets the **Command** (p. 866) Id of this **Message** (p. 1821).

Returns

Command (p. 866) Id

Implements **activemq::commands::Command** (p. 867).

Referenced by copyDataStructure().

6.69.2.4 `virtual bool activemq::commands::BaseCommand::isBrokerInfo () const`
[inline, virtual]

Implements **activemq::commands::Command** (p. 867).

Reimplemented in **activemq::commands::BrokerInfo** (p. 575).

6.69.2.5 `virtual bool activemq::commands::BaseCommand::isConnectionControl () const` [inline, virtual]

Implements **activemq::commands::Command** (p. 867).

Reimplemented in **activemq::commands::ConnectionControl** (p. 941).

6.69.2.6 `virtual bool activemq::commands::BaseCommand::isConnectionInfo () const` [inline, virtual]

Implements **activemq::commands::Command** (p. 867).

Reimplemented in **activemq::commands::ConnectionInfo** (p. 971).

6.69.2.7 `virtual bool activemq::commands::BaseCommand::isConsumerInfo () const` [inline, virtual]

Implements **activemq::commands::Command** (p. 868).

Reimplemented in **activemq::commands::ConsumerInfo** (p. 1014).

6.69.2.8 `virtual bool activemq::commands::BaseCommand::isKeepAliveInfo () const` [inline, virtual]

Implements **activemq::commands::Command** (p. 868).

Reimplemented in **activemq::commands::KeepAliveInfo** (p. 1593).

6.69.2.9 `virtual bool activemq::commands::BaseCommand::isMessage () const`
`[inline, virtual]`

Implements **activemq::commands::Command** (p. 868).

Reimplemented in **activemq::commands::Message** (p. 1832).

6.69.2.10 `virtual bool activemq::commands::BaseCommand::isMessageAck ()`
`const [inline, virtual]`

Implements **activemq::commands::Command** (p. 868).

Reimplemented in **activemq::commands::MessageAck** (p. 1871).

6.69.2.11 `virtual bool activemq::commands::BaseCommand::isMessageDispatch (`
`)const [inline, virtual]`

Implements **activemq::commands::Command** (p. 868).

Reimplemented in **activemq::commands::MessageDispatch** (p. 1884).

6.69.2.12 `virtual bool activemq::commands::BaseCommand::is-`
`MessageDispatchNotification () const [inline,`
`virtual]`

Implements **activemq::commands::Command** (p. 868).

Reimplemented in **activemq::commands::MessageDispatchNotification** (p. 1898).

6.69.2.13 `virtual bool activemq::commands::BaseCommand::isProducerAck ()`
`const [inline, virtual]`

Implements **activemq::commands::Command** (p. 868).

Reimplemented in **activemq::commands::ProducerAck** (p. 2173).

6.69.2.14 `virtual bool activemq::commands::BaseCommand::isProducerInfo ()`
`const [inline, virtual]`

Implements **activemq::commands::Command** (p. 869).

Reimplemented in **activemq::commands::ProducerInfo** (p. 2193).

6.69.2.15 `virtual bool activemq::commands::BaseCommand::isRemoveInfo ()`
`const [inline, virtual]`

Implements **activemq::commands::Command** (p. 869).

Reimplemented in **activemq::commands::RemoveInfo** (p. 2270).

6.69.2.16 `virtual bool activemq::commands::BaseCommand::isRemoveSubscriptionInfo () const [inline, virtual]`

Implements **activemq::commands::Command** (p. 869).

Reimplemented in **activemq::commands::RemoveSubscriptionInfo** (p. 2278).

6.69.2.17 `virtual bool activemq::commands::BaseCommand::isResponse () const [inline, virtual]`

Implements **activemq::commands::Command** (p. 869).

Reimplemented in **activemq::commands::Response** (p. 2300).

6.69.2.18 `virtual bool activemq::commands::BaseCommand::isResponseRequired () const [inline, virtual]`

Is a **Response** (p. 2298) required for this **Command** (p. 866).

Returns

true if a response is required.

Implements **activemq::commands::Command** (p. 869).

Referenced by `copyDataStructure()`.

6.69.2.19 `virtual bool activemq::commands::BaseCommand::isShutdownInfo () const [inline, virtual]`

Implements **activemq::commands::Command** (p. 869).

Reimplemented in **activemq::commands::ShutdownInfo** (p. 2433).

6.69.2.20 `virtual bool activemq::commands::BaseCommand::isTransactionInfo () const [inline, virtual]`

Implements **activemq::commands::Command** (p. 869).

Reimplemented in **activemq::commands::TransactionInfo** (p. 2777).

6.69.2.21 `virtual bool activemq::commands::BaseCommand::isWireFormatInfo () const [inline, virtual]`

Implements **activemq::commands::Command** (p. 870).

Reimplemented in **activemq::commands::WireFormatInfo** (p. 2896).

6.69.2.22 `virtual void activemq::commands::BaseCommand::setCommandId (int id) [inline, virtual]`

Sets the **Command** (p. 866) Id of this **Message** (p. 1821).

Parameters

<i>id</i>	Command (p. 866) Id
-----------	----------------------------

Implements **activemq::commands::Command** (p. 870).

6.69.2.23 `virtual void activemq::commands::BaseCommand::setResponseRequired (const bool required) [inline, virtual]`

Set if this **Message** (p. 1821) requires a **Response** (p. 2298).

Parameters

<i>required</i>	true if response is required
-----------------	------------------------------

Implements **activemq::commands::Command** (p. 870).

6.69.2.24 `virtual std::string activemq::commands::BaseCommand::toString () const [inline, virtual]`

Returns a string containing the information for this **DataStructure** (p. 1133) such as its type and value of its elements.

Returns

formatted string useful for debugging.

Implements **activemq::commands::Command** (p. 870).

Reimplemented in **activemq::commands::Message** (p. 1836), **activemq::commands::ConsumerInfo** (p. 1016), **activemq::commands::ActiveMQBytesMessage** (p. 178), **activemq::commands::BrokerInfo** (p. 577), **activemq::commands::ConnectionInfo** (p. 972), **activemq::commands::MessageAck** (p. 1871), **activemq::commands::ConsumerControl** (p. 995), **activemq::commands::ProducerInfo** (p. 2194), **activemq::commands::ConnectionControl** (p. 942), **activemq::commands::DestinationInfo** (p. 1217), **activemq::commands::MessagePull** (p. 1943), **activemq::commands::SessionInfo** (p. 2389), **activemq::commands::MessageDispatch** (p. 1885), **activemq::commands::MessageDispatchNotification** (p. 1898), **activemq::commands::TransactionInfo** (p. 2777), **activemq::commands::ConnectionError** (p. 950), **activemq::commands::RemoveSubscriptionInfo** (p. 2279), **activemq::commands::ActiveMQStreamMessage** (p. 369), **activemq::commands::ProducerAck** (p. 2174), **activemq::commands::RemoveInfo** (p. 2270), **activemq::commands::ActiveMQMapMessage** (p. 283),

6.70

activemq::wireformat::openwire::marshal::generated::BaseCommandMarshaller Class Reference 499

[activemq::commands::DataArrayResponse](#) (p. 1071), [activemq::commands::DataResponse](#) (p. 1114), [activemq::commands::ExceptionResponse](#) (p. 1290), [activemq::commands::ReplayCommand](#) (p. 2286), [activemq::commands::ControlCommand](#) (p. 1025), [activemq::commands::IntegerResponse](#) (p. 1518), [activemq::commands::Response](#) (p. 2300), [activemq::commands::FlushCommand](#) (p. 1383), [activemq::commands::KeepAliveInfo](#) (p. 1593), [activemq::commands::ShutdownInfo](#) (p. 2433), [activemq::commands::ActiveMQBlobMessage](#) (p. 161), [activemq::commands::WireFormatInfo](#) (p. 2899), [activemq::commands::ActiveMQTextMessage](#) (p. 409), [activemq::commands::ActiveMQObjectMessage](#) (p. 303), and [activemq::commands::ActiveMQMessage](#) (p. 290).

References [activemq::commands::BaseDataStructure::toString\(\)](#).

The documentation for this class was generated from the following file:

- `src/main/activemq/commands/BaseCommand.h`

6.70 activemq::wireformat::openwire::marshal::generated::Base- CommandMarshaller Class Reference

Marshaling code for Open Wire Format for **BaseCommandMarshaller** (p. 499).

```
#include <src/main/activemq/wireformat/openwire/marshal/generated/-  
BaseCommandMarshaller.h>
```

Inheritance diagram for `activemq::wireformat::openwire::marshal::generated::Base-
CommandMarshaller`:

Public Member Functions

- **BaseCommandMarshaller** ()
- virtual **~BaseCommandMarshaller** ()
- virtual void **tightUnmarshal** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataInputStream** *dataIn, **utils::BooleanStream** *bs)
Tight Un-marhsal to the given stream.
- virtual int **tightMarshal1** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **utils::BooleanStream** *bs)
Tight Marhsal to the given stream.
- virtual void **tightMarshal2** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataOutputStream** *dataOut, **utils::BooleanStream** *bs)
Tight Marhsal to the given stream.
- virtual void **looseUnmarshal** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataInputStream** *dataIn)

Loose Un-marhsal to the given stream.

- virtual void **looseMarshal** (**OpenWireFormat** *wireFormat, **commands::Data-Structure** *dataStructure, **decaf::io::DataOutputStream** *dataOut)

Tight Marhsal to the given stream.

6.70.1 Detailed Description

Marshaling code for Open Wire Format for **BaseCommandMarshaller** (p. 499).

NOTE!: This file is auto generated - do not modify! if you need to make a change, please see the Java Classes in the activemq-openwire-generator module

6.70.2 Constructor & Destructor Documentation

6.70.2.1 **activemq::wireformat::openwire::marshal::generated::BaseCommandMarshaller::BaseCommandMarshaller** ()
[inline]

6.70.2.2 **virtual activemq::wireformat::openwire::marshal::generated::BaseCommandMarshaller::~~BaseCommandMarshaller** () [inline, virtual]

6.70.3 Member Function Documentation

6.70.3.1 **virtual void activemq::wireformat::openwire::marshal::generated::BaseCommandMarshaller::looseMarshal** (**OpenWireFormat** * format, **commands::DataStructure** * command, **decaf::io::DataOutputStream** * ds) [virtual]

Tight Marhsal to the given stream.

Parameters

<i>format</i>	- The OpenWireFormat properties
<i>command</i>	- the object to Marshal
<i>ds</i>	- DataOutputStream to marshal to

Exceptions

<i>IOException</i>	if an error occurs.
--------------------	---------------------

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1123).

Reimplemented in **activemq::wireformat::openwire::marshal::generated::ActiveMQBlobMessageMarshaller** (p. 163), **activemq::wireformat::openwire::marshal::generated::ActiveMQBytesMessageMarshaller** (p. 185), **activemq::wireformat::openwire::marshal::generated::ActiveMQMapMessageMarshaller** (p. 286),

6.70

activemq::wireformat::openwire::marshal::generated::BaseCommandMarshaller
 Class Reference 501
 activemq::wireformat::openwire::marshal::generated::ActiveMQMessageMarshaller
 (p. 292), activemq::wireformat::openwire::marshal::generated::ActiveMQObject-
 MessageMarshaller (p. 305), activemq::wireformat::openwire::marshal::generated-
 ::ActiveMQStreamMessageMarshaller (p. 376), activemq::wireformat::openwire-
 ::marshal::generated::ActiveMQTextMessageMarshaller (p. 412), activemq-
 ::wireformat::openwire::marshal::generated::BrokerInfoMarshaller (p. 580),
 activemq::wireformat::openwire::marshal::generated::ConnectionControl-
 Marshaller (p. 945), activemq::wireformat::openwire::marshal::generated::-
 ConnectionErrorMarshaller (p. 953), activemq::wireformat::openwire::marshal-
 ::generated::ConnectionInfoMarshaller (p. 976), activemq::wireformat::openwire-
 ::marshal::generated::ConsumerControlMarshaller (p. 998), activemq::wireformat-
 ::openwire::marshal::generated::ConsumerInfoMarshaller (p. 1019), activemq-
 ::wireformat::openwire::marshal::generated::ControlCommandMarshaller (p. 1027),
 activemq::wireformat::openwire::marshal::generated::DataArrayResponse-
 Marshaller (p. 1073), activemq::wireformat::openwire::marshal::generated::-
 DataResponseMarshaller (p. 1117), activemq::wireformat::openwire::marshal-
 ::generated::DestinationInfoMarshaller (p. 1220), activemq::wireformat::openwire-
 ::marshal::generated::ExceptionResponseMarshaller (p. 1292), activemq-
 ::wireformat::openwire::marshal::generated::FlushCommandMarshaller (p. 1385),
 activemq::wireformat::openwire::marshal::generated::IntegerResponseMarshaller
 (p. 1521), activemq::wireformat::openwire::marshal::generated::KeepAliveInfo-
 Marshaller (p. 1596), activemq::wireformat::openwire::marshal::generated::-
 MessageAckMarshaller (p. 1874), activemq::wireformat::openwire::marshal-
 ::generated::MessageDispatchMarshaller (p. 1892), activemq::wireformat-
 ::openwire::marshal::generated::MessageDispatchNotificationMarshaller (p. 1901),
 activemq::wireformat::openwire::marshal::generated::MessagePullMarshaller
 (p. 1946), activemq::wireformat::openwire::marshal::generated::ProducerAck-
 Marshaller (p. 2177), activemq::wireformat::openwire::marshal::generated::-
 ProducerInfoMarshaller (p. 2197), activemq::wireformat::openwire::marshal-
 ::generated::RemoveInfoMarshaller (p. 2273), activemq::wireformat::openwire-
 ::marshal::generated::RemoveSubscriptionInfoMarshaller (p. 2281), activemq-
 ::wireformat::openwire::marshal::generated::ReplayCommandMarshaller (p. 2289),
 activemq::wireformat::openwire::marshal::generated::ResponseMarshaller
 (p. 2309), activemq::wireformat::openwire::marshal::generated::SessionInfo-
 Marshaller (p. 2391), activemq::wireformat::openwire::marshal::generated::-
 ShutdownInfoMarshaller (p. 2436), activemq::wireformat::openwire::marshal-
 ::generated::TransactionInfoMarshaller (p. 2780), and activemq::wireformat-
 ::openwire::marshal::generated::MessageMarshaller (p. 1919).

6.70.3.2 virtual void activemq::wireformat::openwire::marshal::generated::Base-
 CommandMarshaller::looseUnmarshal (OpenWireFormat * format,
 commands::DataStructure * command, decaf::io::DataInputStream * dis)
 [virtual]

Loose Un-marhsal to the given stream.

Parameters

<i>format</i>	- The OpenwireFormat properties
<i>command</i>	- the object to Un-Marshal
<i>dis</i>	- the DataInputStream to Un-Marshal from

Exceptions

<i>IOException</i>	if an error occurs.
--------------------	---------------------

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1125).

Reimplemented in **activemq::wireformat::openwire::marshal::generated::ActiveMQBlobMessageMarshaller** (p. 164), **activemq::wireformat::openwire::marshal::generated::ActiveMQBytesMessageMarshaller** (p. 185), **activemq::wireformat::openwire::marshal::generated::ActiveMQMapMessageMarshaller** (p. 286), **activemq::wireformat::openwire::marshal::generated::ActiveMQMessageMarshaller** (p. 293), **activemq::wireformat::openwire::marshal::generated::ActiveMQObjectMessageMarshaller** (p. 306), **activemq::wireformat::openwire::marshal::generated::ActiveMQStreamMessageMarshaller** (p. 377), **activemq::wireformat::openwire::marshal::generated::ActiveMQTextMessageMarshaller** (p. 412), **activemq::wireformat::openwire::marshal::generated::BrokerInfoMarshaller** (p. 580), **activemq::wireformat::openwire::marshal::generated::ConnectionControlMarshaller** (p. 946), **activemq::wireformat::openwire::marshal::generated::ConnectionErrorMarshaller** (p. 953), **activemq::wireformat::openwire::marshal::generated::ConnectionInfoMarshaller** (p. 976), **activemq::wireformat::openwire::marshal::generated::ConsumerControlMarshaller** (p. 998), **activemq::wireformat::openwire::marshal::generated::ConsumerInfoMarshaller** (p. 1020), **activemq::wireformat::openwire::marshal::generated::ControlCommandMarshaller** (p. 1028), **activemq::wireformat::openwire::marshal::generated::DataArrayResponseMarshaller** (p. 1074), **activemq::wireformat::openwire::marshal::generated::DataResponseMarshaller** (p. 1117), **activemq::wireformat::openwire::marshal::generated::DestinationInfoMarshaller** (p. 1220), **activemq::wireformat::openwire::marshal::generated::ExceptionResponseMarshaller** (p. 1293), **activemq::wireformat::openwire::marshal::generated::FlushCommandMarshaller** (p. 1386), **activemq::wireformat::openwire::marshal::generated::IntegerResponseMarshaller** (p. 1521), **activemq::wireformat::openwire::marshal::generated::KeepAliveInfoMarshaller** (p. 1596), **activemq::wireformat::openwire::marshal::generated::MessageAckMarshaller** (p. 1875), **activemq::wireformat::openwire::marshal::generated::MessageDispatchMarshaller** (p. 1893), **activemq::wireformat::openwire::marshal::generated::MessageDispatchNotificationMarshaller** (p. 1902), **activemq::wireformat::openwire::marshal::generated::MessagePullMarshaller** (p. 1946), **activemq::wireformat::openwire::marshal::generated::ProducerAckMarshaller** (p. 2177), **activemq::wireformat::openwire::marshal::generated::ProducerInfoMarshaller** (p. 2197), **activemq::wireformat::openwire::marshal::generated::RemoveInfoMarshaller** (p. 2273), **activemq::wireformat::openwire::marshal::generated::RemoveSubscriptionInfoMarshaller** (p. 2282), **activemq::wireformat::openwire::marshal::generated::ReplayCommandMarshaller** (p. 2289), **activemq::wireformat::openwire::marshal::generated::ResponseMarshaller**

6.70

activemq::wireformat::openwire::marshal::generated::BaseCommandMarshaller
Class Reference 503
(p. 2310), **activemq::wireformat::openwire::marshal::generated::SessionInfoMarshaller** (p. 2392), **activemq::wireformat::openwire::marshal::generated::ShutdownInfoMarshaller** (p. 2436), **activemq::wireformat::openwire::marshal::generated::TransactionInfoMarshaller** (p. 2780), and **activemq::wireformat::openwire::marshal::generated::MessageMarshaller** (p. 1919).

6.70.3.3 **virtual int activemq::wireformat::openwire::marshal::generated::BaseCommandMarshaller::tightMarshal1 (OpenWireFormat * *format*, commands::DataStructure * *command*, utils::BooleanStream * *bs*)**
[virtual]

Tight Marshal to the given stream.

Parameters

<i>format</i>	The OpenWireFormat properties
<i>command</i>	The object to Marshal
<i>bs</i>	The boolean stream to marshal to.

Exceptions

<i>IOException</i>	if an error occurs.
--------------------	---------------------

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller**
(p. 1127).

Reimplemented in **activemq::wireformat::openwire::marshal::generated::ActiveMQBlobMessageMarshaller** (p. 164), **activemq::wireformat::openwire::marshal::generated::ActiveMQBytesMessageMarshaller** (p. 186), **activemq::wireformat::openwire::marshal::generated::ActiveMQMapMessageMarshaller** (p. 286), **activemq::wireformat::openwire::marshal::generated::ActiveMQMessageMarshaller** (p. 293), **activemq::wireformat::openwire::marshal::generated::ActiveMQObjectMessageMarshaller** (p. 306), **activemq::wireformat::openwire::marshal::generated::ActiveMQStreamMessageMarshaller** (p. 377), **activemq::wireformat::openwire::marshal::generated::ActiveMQTextMessageMarshaller** (p. 413), **activemq::wireformat::openwire::marshal::generated::BrokerInfoMarshaller** (p. 581), **activemq::wireformat::openwire::marshal::generated::ConnectionControlMarshaller** (p. 946), **activemq::wireformat::openwire::marshal::generated::ConnectionErrorMarshaller** (p. 954), **activemq::wireformat::openwire::marshal::generated::ConnectionInfoMarshaller** (p. 976), **activemq::wireformat::openwire::marshal::generated::ConsumerControlMarshaller** (p. 999), **activemq::wireformat::openwire::marshal::generated::ConsumerInfoMarshaller** (p. 1020), **activemq::wireformat::openwire::marshal::generated::ControlCommandMarshaller** (p. 1028), **activemq::wireformat::openwire::marshal::generated::DataArrayResponseMarshaller** (p. 1074), **activemq::wireformat::openwire::marshal::generated::DataResponseMarshaller** (p. 1118), **activemq::wireformat::openwire::marshal::generated::DestinationInfoMarshaller** (p. 1220), **activemq::wireformat::openwire::marshal::generated::ExceptionResponseMarshaller** (p. 1293), **activemq::wireformat::openwire::marshal::generated::FlushCommandMarshaller** (p. 1386),

`activemq::wireformat::openwire::marshal::generated::IntegerResponseMarshaller` (p. 1521), `activemq::wireformat::openwire::marshal::generated::KeepAliveInfoMarshaller` (p. 1597), `activemq::wireformat::openwire::marshal::generated::MessageAckMarshaller` (p. 1875), `activemq::wireformat::openwire::marshal::generated::MessageDispatchMarshaller` (p. 1893), `activemq::wireformat::openwire::marshal::generated::MessageDispatchNotificationMarshaller` (p. 1902), `activemq::wireformat::openwire::marshal::generated::MessagePullMarshaller` (p. 1946), `activemq::wireformat::openwire::marshal::generated::ProducerAckMarshaller` (p. 2177), `activemq::wireformat::openwire::marshal::generated::ProducerInfoMarshaller` (p. 2197), `activemq::wireformat::openwire::marshal::generated::RemoveInfoMarshaller` (p. 2274), `activemq::wireformat::openwire::marshal::generated::RemoveSubscriptionInfoMarshaller` (p. 2282), `activemq::wireformat::openwire::marshal::generated::ReplayCommandMarshaller` (p. 2290), `activemq::wireformat::openwire::marshal::generated::ResponseMarshaller` (p. 2310), `activemq::wireformat::openwire::marshal::generated::SessionInfoMarshaller` (p. 2392), `activemq::wireformat::openwire::marshal::generated::ShutdownInfoMarshaller` (p. 2437), `activemq::wireformat::openwire::marshal::generated::TransactionInfoMarshaller` (p. 2781), and `activemq::wireformat::openwire::marshal::generated::MessageMarshaller` (p. 1920).

6.70.3.4 virtual void `activemq::wireformat::openwire::marshal::generated::BaseCommandMarshaller::tightMarshal2` (`OpenWireFormat * format`, `commands::DataStructure * command`, `decaf::io::DataOutputStream * ds`, `utils::BooleanStream * bs`) [virtual]

Tight Marshal to the given stream.

Parameters

<i>format</i>	- The OpenWireFormat properties
<i>command</i>	- the object to Marshal
<i>ds</i>	- the DataOutputStream to Marshal to
<i>bs</i>	- boolean stream to marshal to.

Exceptions

<i>IOException</i>	if an error occurs.
--------------------	---------------------

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1129).

Reimplemented in `activemq::wireformat::openwire::marshal::generated::ActiveMQBlobMessageMarshaller` (p. 165), `activemq::wireformat::openwire::marshal::generated::ActiveMQBytesMessageMarshaller` (p. 186), `activemq::wireformat::openwire::marshal::generated::ActiveMQMapMessageMarshaller` (p. 287), `activemq::wireformat::openwire::marshal::generated::ActiveMQMessageMarshaller` (p. 294), `activemq::wireformat::openwire::marshal::generated::ActiveMQObjectMessageMarshaller` (p. 307), `activemq::wireformat::openwire::marshal::generated::ActiveMQStreamMessageMarshaller` (p. 378), `activemq::wireformat::openwire-`

6.70

activemq::wireformat::openwire::marshal::generated::BaseCommandMarshaller
 Class Reference 505
 ::marshal::generated::ActiveMQTextMessageMarshaller (p. 413), activemq-
 ::wireformat::openwire::marshal::generated::BrokerInfoMarshaller (p. 581),
 activemq::wireformat::openwire::marshal::generated::ConnectionControl-
 Marshaller (p. 947), activemq::wireformat::openwire::marshal::generated::-
 ConnectionErrorMarshaller (p. 954), activemq::wireformat::openwire::marshal-
 ::generated::ConnectionInfoMarshaller (p. 977), activemq::wireformat::openwire-
 ::marshal::generated::ConsumerControlMarshaller (p. 999), activemq::wireformat-
 ::openwire::marshal::generated::ConsumerInfoMarshaller (p. 1021), activemq-
 ::wireformat::openwire::marshal::generated::ControlCommandMarshaller (p. 1029),
 activemq::wireformat::openwire::marshal::generated::DataArrayResponse-
 Marshaller (p. 1075), activemq::wireformat::openwire::marshal::generated::-
 DataResponseMarshaller (p. 1118), activemq::wireformat::openwire::marshal-
 ::generated::DestinationInfoMarshaller (p. 1221), activemq::wireformat::openwire-
 ::marshal::generated::ExceptionResponseMarshaller (p. 1293), activemq-
 ::wireformat::openwire::marshal::generated::FlushCommandMarshaller (p. 1386),
 activemq::wireformat::openwire::marshal::generated::IntegerResponseMarshaller
 (p. 1522), activemq::wireformat::openwire::marshal::generated::KeepAliveInfo-
 Marshaller (p. 1597), activemq::wireformat::openwire::marshal::generated::-
 MessageAckMarshaller (p. 1876), activemq::wireformat::openwire::marshal-
 ::generated::MessageDispatchMarshaller (p. 1893), activemq::wireformat-
 ::openwire::marshal::generated::MessageDispatchNotificationMarshaller (p. 1902),
 activemq::wireformat::openwire::marshal::generated::MessagePullMarshaller
 (p. 1947), activemq::wireformat::openwire::marshal::generated::ProducerAck-
 Marshaller (p. 2178), activemq::wireformat::openwire::marshal::generated::-
 ProducerInfoMarshaller (p. 2198), activemq::wireformat::openwire::marshal-
 ::generated::RemoveInfoMarshaller (p. 2274), activemq::wireformat::openwire-
 ::marshal::generated::RemoveSubscriptionInfoMarshaller (p. 2283), activemq-
 ::wireformat::openwire::marshal::generated::ReplayCommandMarshaller (p. 2290),
 activemq::wireformat::openwire::marshal::generated::ResponseMarshaller
 (p. 2311), activemq::wireformat::openwire::marshal::generated::SessionInfo-
 Marshaller (p. 2393), activemq::wireformat::openwire::marshal::generated::-
 ShutdownInfoMarshaller (p. 2437), activemq::wireformat::openwire::marshal-
 ::generated::TransactionInfoMarshaller (p. 2781), and activemq::wireformat-
 ::openwire::marshal::generated::MessageMarshaller (p. 1921).

6.70.3.5 virtual void activemq::wireformat::openwire::marshal::generated::Base-
 CommandMarshaller::tightUnmarshal (OpenWireFormat * *format*,
 commands::DataStructure * *command*, decaf::io::DataInputStream * *dis*,
 utils::BooleanStream * *bs*) [virtual]

Tight Un-marhsal to the given stream.

Parameters

<i>format</i>	- The OpenWireFormat properties
<i>command</i>	- the object to Un-Marshal
<i>dis</i>	- the DataInputStream to Un-Marshal from
<i>bs</i>	- boolean stream to unmarshal from.

Exceptions

<i>IOException</i> if an error occurs.
--

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1131).

Reimplemented in **activemq::wireformat::openwire::marshal::generated::ActiveMQBlobMessageMarshaller** (p. 165), **activemq::wireformat::openwire::marshal::generated::ActiveMQBytesMessageMarshaller** (p. 187), **activemq::wireformat::openwire::marshal::generated::ActiveMQMapMessageMarshaller** (p. 287), **activemq::wireformat::openwire::marshal::generated::ActiveMQMessageMarshaller** (p. 294), **activemq::wireformat::openwire::marshal::generated::ActiveMQObjectMessageMarshaller** (p. 307), **activemq::wireformat::openwire::marshal::generated::ActiveMQStreamMessageMarshaller** (p. 378), **activemq::wireformat::openwire::marshal::generated::ActiveMQTextMessageMarshaller** (p. 414), **activemq::wireformat::openwire::marshal::generated::BrokerInfoMarshaller** (p. 582), **activemq::wireformat::openwire::marshal::generated::ConnectionControlMarshaller** (p. 947), **activemq::wireformat::openwire::marshal::generated::ConnectionErrorMarshaller** (p. 955), **activemq::wireformat::openwire::marshal::generated::ConnectionInfoMarshaller** (p. 977), **activemq::wireformat::openwire::marshal::generated::ConsumerControlMarshaller** (p. 1000), **activemq::wireformat::openwire::marshal::generated::ConsumerInfoMarshaller** (p. 1021), **activemq::wireformat::openwire::marshal::generated::ControlCommandMarshaller** (p. 1029), **activemq::wireformat::openwire::marshal::generated::DataArrayResponseMarshaller** (p. 1075), **activemq::wireformat::openwire::marshal::generated::DataResponseMarshaller** (p. 1119), **activemq::wireformat::openwire::marshal::generated::DestinationInfoMarshaller** (p. 1221), **activemq::wireformat::openwire::marshal::generated::ExceptionResponseMarshaller** (p. 1294), **activemq::wireformat::openwire::marshal::generated::FlushCommandMarshaller** (p. 1387), **activemq::wireformat::openwire::marshal::generated::IntegerResponseMarshaller** (p. 1522), **activemq::wireformat::openwire::marshal::generated::KeepAliveInfoMarshaller** (p. 1598), **activemq::wireformat::openwire::marshal::generated::MessageAckMarshaller** (p. 1876), **activemq::wireformat::openwire::marshal::generated::MessageDispatchMarshaller** (p. 1894), **activemq::wireformat::openwire::marshal::generated::MessageDispatchNotificationMarshaller** (p. 1903), **activemq::wireformat::openwire::marshal::generated::MessagePullMarshaller** (p. 1947), **activemq::wireformat::openwire::marshal::generated::ProducerAckMarshaller** (p. 2178), **activemq::wireformat::openwire::marshal::generated::ProducerInfoMarshaller** (p. 2198), **activemq::wireformat::openwire::marshal::generated::RemoveInfoMarshaller** (p. 2275), **activemq::wireformat::openwire::marshal::generated::RemoveSubscriptionInfoMarshaller** (p. 2283), **activemq::wireformat::openwire::marshal::generated::ReplayCommandMarshaller** (p. 2290), **activemq::wireformat::openwire::marshal::generated::ResponseMarshaller** (p. 2311), **activemq::wireformat::openwire::marshal::generated::SessionInfoMarshaller** (p. 2393), **activemq::wireformat::openwire::marshal::generated::ShutdownInfoMarshaller** (p. 2438), **activemq::wireformat::openwire::marshal::generated::TransactionInfoMarshaller** (p. 2781), and **activemq::wireformat::openwire::marshal::generated::MessageMarshaller** (p. 1921).

The documentation for this class was generated from the following file:

- src/main/activemq/wireformat/openwire/marshal/generated/**BaseCommandMarshaller.h**

6.71 activemq::wireformat::openwire::marshal::BaseDataStreamMarshaller Class Reference

Base class for all Marshallers that marshal DataStructures to and from the wire using the OpenWire protocol.

```
#include <src/main/activemq/wireformat/openwire/marshal/-
BaseDataStreamMarshaller.h>
```

Inheritance diagram for activemq::wireformat::openwire::marshal::BaseDataStreamMarshaller:

Public Member Functions

- virtual **~BaseDataStreamMarshaller** ()
- virtual int **tightMarshal1** (**OpenWireFormat** *format AMQCPP_UNUSED, **commands::DataStructure** *command AMQCPP_UNUSED, **utils::BooleanStream** *bs AMQCPP_UNUSED)
Tight Marshal to the given stream.
- virtual void **tightMarshal2** (**OpenWireFormat** *format AMQCPP_UNUSED, **commands::DataStructure** *command AMQCPP_UNUSED, **decaf::io::DataOutputStream** *ds AMQCPP_UNUSED, **utils::BooleanStream** *bs AMQCPP_UNUSED)
Tight Marshal to the given stream.
- virtual void **tightUnmarshal** (**OpenWireFormat** *format AMQCPP_UNUSED, **commands::DataStructure** *command AMQCPP_UNUSED, **decaf::io::DataInputStream** *dis AMQCPP_UNUSED, **utils::BooleanStream** *bs AMQCPP_UNUSED)
Tight Un-Marshal to the given stream.
- virtual void **looseMarshal** (**OpenWireFormat** *format AMQCPP_UNUSED, **commands::DataStructure** *command AMQCPP_UNUSED, **decaf::io::DataOutputStream** *ds AMQCPP_UNUSED)
Tight Marshal to the given stream.
- virtual void **looseUnmarshal** (**OpenWireFormat** *format AMQCPP_UNUSED, **commands::DataStructure** *command AMQCPP_UNUSED, **decaf::io::DataInputStream** *dis AMQCPP_UNUSED)
Loose Un-Marshal to the given stream.

Static Public Member Functions

- static std::string **toString** (const **commands::MessageId** *id)

Converts the object to a String.

- static std::string **toString** (const **commands::ProducerId** *id)

Converts the object to a String.

- static std::string **toString** (const **commands::TransactionId** *txnId)

Converts the given transaction ID into a String.

- static std::string **toHexFromBytes** (const std::vector< unsigned char > &data)

given an array of bytes, convert that array to a Hexidecimal coded string that represents that data.

Protected Member Functions

- virtual **commands::DataStructure** * **tightUnmarshalCachedObject** (**OpenWireFormat** *wireFormat, **decaf::io::DataInputStream** *dataIn, **utils::BooleanStream** *bs)

Tight Unmarshal the cached object.

- virtual int **tightMarshalCachedObject1** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *data, **utils::BooleanStream** *bs)

Tightly marshals the passed DataStructure based object to the passed BooleanStream returning the size of the data marshaled.

- virtual void **tightMarshalCachedObject2** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *data, **decaf::io::DataOutputStream** *dataOut, **utils::BooleanStream** *bs)

Tightly marshals the passed DataStructure based object to the passed streams returning nothing.

- virtual void **looseMarshalCachedObject** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *data, **decaf::io::DataOutputStream** *dataOut)

Loosely marshals the passed DataStructure based object to the passed stream returning nothing.

- virtual **commands::DataStructure** * **looseUnmarshalCachedObject** (**OpenWireFormat** *wireFormat, **decaf::io::DataInputStream** *dataIn)

Loose Unmarshal the cached object.

- virtual int **tightMarshalNestedObject1** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *object, **utils::BooleanStream** *bs)

Tightly marshals the passed DataStructure based object to the passed BooleanStream returning the size of the data marshaled.

- virtual void **tightMarshalNestedObject2** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *object, **decaf::io::DataOutputStream** *dataOut, **utils::BooleanStream** *bs)

Tightly marshals the passed DataStructure based object to the passed streams returning nothing.

- virtual **commands::DataStructure** * **tightUnmarshalNestedObject** (**OpenWireFormat** *wireFormat, **decaf::io::DataInputStream** *dataIn, **utils::BooleanStream** *bs)

Tight Unmarshal the nested object.

- virtual **commands::DataStructure** * **looseUnmarshalNestedObject** (**OpenWireFormat** *wireFormat, **decaf::io::DataInputStream** *dataIn)

Loose Unmarshal the nested object.

- virtual void **looseMarshalNestedObject** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *object, **decaf::io::DataOutputStream** *dataOut)

Loose marshal the nested object.

- virtual std::string **tightUnmarshalString** (**decaf::io::DataInputStream** *dataIn, **utils::BooleanStream** *bs)

Performs Tight Unmarshaling of String Objects.

- virtual int **tightMarshalString1** (const std::string &value, **utils::BooleanStream** *bs)

Tight Marshals the String to a Booleans Stream Object, returns the marshaled size.

- virtual void **tightMarshalString2** (const std::string &value, **decaf::io::DataOutputStream** *dataOut, **utils::BooleanStream** *bs)

Tight Marshals the passed string to the streams passed.

- virtual void **looseMarshalString** (const std::string value, **decaf::io::DataOutputStream** *dataOut)

Loose Marshal the String to the DataOuputStream passed.

- virtual std::string **looseUnmarshalString** (**decaf::io::DataInputStream** *dataIn)

Loose Un-Marshal the String to the DataOuputStream passed.

- virtual int **tightMarshalLong1** (**OpenWireFormat** *wireFormat, long long value, **utils::BooleanStream** *bs)

Tightly marshal the long long to the BooleanStream passed.

- virtual void **tightMarshalLong2** (**OpenWireFormat** *wireFormat, long long value, **decaf::io::DataOutputStream** *dataOut, **utils::BooleanStream** *bs)

Tightly marshal the long long to the Streams passed.

- virtual long long **tightUnmarshalLong** (**OpenWireFormat** *wireFormat, **decaf::io::DataInputStream** *dataIn, **utils::BooleanStream** *bs)

Tight marshal the long long type.

- virtual void **looseMarshalLong** (**OpenWireFormat** *wireFormat, long long value, **decaf::io::DataOutputStream** *dataOut)

Tightly marshal the long long to the BooleanStream passed.

- virtual long long **looseUnmarshalLong** (**OpenWireFormat** *wireFormat, **decaf::io::DataInputStream** *dataIn)

Loose marshal the long long type.

- virtual std::vector< unsigned char > **tightUnmarshalByteArray** (**decaf::io::DataInputStream** *dataIn, **utils::BooleanStream** *bs)

Tight Unmarshal an array of char.

- virtual std::vector< unsigned char > **looseUnmarshalByteArray** (**decaf::io::DataInputStream** *dataIn)

Loose Unmarshal an array of char.

- virtual std::vector< unsigned char > **tightUnmarshalConstByteArray** (**decaf::io::DataInputStream** *dataIn, **utils::BooleanStream** *bs, int size)

Tight Unmarshal a fixed size array from that data input stream and return an stl vector of char as the resultant.

- virtual `std::vector< unsigned char > looseUnmarshalConstByteArray (decaf::io::DataInputStream *dataIn, int size)`
Tight Unmarshal a fixed size array from that data input stream and return an stl vector of char as the resultant.
- virtual `commands::DataStructure * tightUnmarshalBrokerError (OpenWireFormat *wireFormat, decaf::io::DataInputStream *dataIn, utils::BooleanStream *bs)`
Tight Unmarshal the Error object.
- virtual `int tightMarshalBrokerError1 (OpenWireFormat *wireFormat, commands::DataStructure *data, utils::BooleanStream *bs)`
Tight Marshal the Error object.
- virtual `void tightMarshalBrokerError2 (OpenWireFormat *wireFormat, commands::DataStructure *data, decaf::io::DataOutputStream *dataOut, utils::BooleanStream *bs)`
Tight Marshal the Error object.
- virtual `commands::DataStructure * looseUnmarshalBrokerError (OpenWireFormat *wireFormat, decaf::io::DataInputStream *dataIn)`
Loose Unmarshal the Error object.
- virtual `void looseMarshalBrokerError (OpenWireFormat *wireFormat, commands::DataStructure *data, decaf::io::DataOutputStream *dataOut)`
Tight Marshal the Error object.
- `template<typename T >`
`int tightMarshalObjectArray1 (OpenWireFormat *wireFormat, std::vector< T > objects, utils::BooleanStream *bs)`
Tightly Marshal an array of DataStructure objects to the provided boolean stream, and return the size that the tight marshalling is going to take.
- `template<typename T >`
`void tightMarshalObjectArray2 (OpenWireFormat *wireFormat, std::vector< T > objects, decaf::io::DataOutputStream *dataOut, utils::BooleanStream *bs)`
Tightly Marshal an array of DataStructure objects to the provided boolean stream and data output stream.
- `template<typename T >`
`void looseMarshalObjectArray (OpenWireFormat *wireFormat, std::vector< T > objects, decaf::io::DataOutputStream *dataOut)`
Loosely Marshal an array of DataStructure objects to the provided boolean stream and data output stream.
- virtual `std::string readAsciiString (decaf::io::DataInputStream *dataIn)`
Given an DataInputStream read a know ASCII formatted string from the input and return that string.

6.71.1 Detailed Description

Base class for all Marshallers that marshal DataStructures to and from the wire using the OpenWire protocol.

Since

2.0

6.71.2 Constructor & Destructor Documentation

6.71.2.1 virtual **activemq::wireformat::openwire::marshal::BaseDataStreamMarshaller::~BaseDataStreamMarshaller** () [inline, virtual]

6.71.3 Member Function Documentation

6.71.3.1 virtual void **activemq::wireformat::openwire::marshal::BaseDataStreamMarshaller::looseMarshal** (**OpenWireFormat** **format* **AMQCPP_UNUSED**, **commands::DataStructure** **command* **AMQCPP_UNUSED**, **decaf::io::DataOutputStream** **ds* **AMQCPP_UNUSED**) [inline, virtual]

Tight Marshal to the given stream.

Parameters

<i>format</i>	- The OpenWireFormat properties
<i>command</i>	- the object to Marshal
<i>ds</i>	- DataOutputStream to marshal to

Exceptions

<i>IOException</i>	if an error occurs.
--------------------	---------------------

6.71.3.2 virtual void **activemq::wireformat::openwire::marshal::BaseDataStreamMarshaller::looseMarshalBrokerError** (**OpenWireFormat** * *wireFormat*, **commands::DataStructure** * *data*, **decaf::io::DataOutputStream** * *dataOut*) [protected, virtual]

Tight Marshal the Error object.

Parameters

<i>wireFormat</i>	- The OpenWireFormat properties
<i>data</i>	- Error to Marshal
<i>dataOut</i>	- stream to write marshalled data to

Exceptions

<i>IOException</i>	if an error occurs.
--------------------	---------------------

6.71.3.3 **virtual void activemq::wireformat::openwire::marshal::BaseDataStream-Marshaller::looseMarshalCachedObject (OpenWireFormat * *wireFormat*, commands::DataStructure * *data*, decaf::io::DataOutputStream * *dataOut*)** [protected, virtual]

Loosely marshals the passed DataStructure based object to the passed stream returning nothing.

Parameters

<i>wireFormat</i>	- The OpenWireFormat properties
<i>data</i>	- DataStructure Object Pointer to marshal
<i>dataOut</i>	- stream to write marshaled data to

Exceptions

<i>IOException</i>	if an error occurs.
--------------------	---------------------

6.71.3.4 **virtual void activemq::wireformat::openwire::marshal::BaseDataStream-Marshaller::looseMarshalLong (OpenWireFormat * *wireFormat*, long long *value*, decaf::io::DataOutputStream * *dataOut*)** [protected, virtual]

Tightly marshal the long long to the BooleanStream passed.

Parameters

<i>wireFormat</i>	- The OpenWireFormat properties
<i>value</i>	- long long to marshal
<i>dataOut</i>	- DataOutputStream to marshal to.

Exceptions

<i>IOException</i>	if an error occurs.
--------------------	---------------------

6.71.3.5 **virtual void activemq::wireformat::openwire::marshal::BaseDataStream-Marshaller::looseMarshalNestedObject (OpenWireFormat * *wireFormat*, commands::DataStructure * *object*, decaf::io::DataOutputStream * *dataOut*)** [protected, virtual]

Loose marshal the nested object.

Parameters

<i>wireFormat</i>	- The OpenWireFormat properties
<i>object</i>	- DataStructure Object Pointer to marshal
<i>dataOut</i>	- stream to write marshaled data to

Exceptions

<i>IOException</i>	if an error occurs.
--------------------	---------------------

6.71.3.6 `template<typename T > void activemq::wireformat::openwire-
::marshal::BaseDataStreamMarshaller::looseMarshalObjectArray
(OpenWireFormat * wireFormat, std::vector< T > objects,
decaf::io::DataOutputStream * dataOut) [inline, protected]`

Loosely Marshal an array of DataStructure objects to the provided boolean stream and data output stream.

Parameters

<i>wireFormat</i>	- The OpenWireFormat properties
<i>objects</i>	- array of DataStructure object pointers.
<i>dataOut</i>	- stream to write marshalled data to

Returns

size of the marshalled data

Exceptions

<i>IOException</i>	if an error occurs.
--------------------	---------------------

References AMQ_CATCH_EXCEPTION_CONVERT, AMQ_CATCH_RETHROW, AMQ_CATCHALL_THROW, decaf::io::DataOutputStream::writeBoolean(), and decaf::io::DataOutputStream::writeShort().

6.71.3.7 `virtual void activemq::wireformat::openwire::marshal::BaseData-
StreamMarshaller::looseMarshalString (const std::string value,
decaf::io::DataOutputStream * dataOut) [protected, virtual]`

Loose Marshal the String to the DataOutputStream passed.

Parameters

<i>value</i>	- string to marshal
<i>dataOut</i>	- stream to write marshaled form to

Exceptions

<i>IOException</i>	if an error occurs.
--------------------	---------------------

```
6.71.3.8 virtual void activemq::wireformat::openwire::marshal::BaseData-
StreamMarshaller::looseUnmarshal ( OpenWireFormat *format
AMQCPP_UNUSED, commands::DataStructure *command AMQCPP_UNUSED,
decaf::io::DataInputStream *dis AMQCPP_UNUSED ) [inline,
virtual]
```

Loose Un-Marshal to the given stream.

Parameters

<i>format</i>	- The OpenWireFormat properties
<i>command</i>	- the object to Un-Marshal
<i>dis</i>	- the DataInputStream to Un-Marshal from

Exceptions

<i>IOException</i>	if an error occurs.
--------------------	---------------------

```
6.71.3.9 virtual commands::DataStructure* activemq::wireformat::openwire-
::marshal::BaseDataStreamMarshaller::looseUnmarshalBrokerError
( OpenWireFormat * wireFormat, decaf::io::DataInputStream * dataIn )
[protected, virtual]
```

Loose Unmarshal the Error object.

Parameters

<i>wireFormat</i>	- The OpenWireFormat properties
<i>dataIn</i>	- stream to read marshalled form from

Returns

pointer to a new DataStructure Object

Exceptions

<i>IOException</i>	if an error occurs.
--------------------	---------------------

```
6.71.3.10 virtual std::vector<unsigned char> activemq::wireformat::openwire-
::marshal::BaseDataStreamMarshaller::looseUnmarshalByteArray (
decaf::io::DataInputStream * dataIn ) [protected, virtual]
```

Loose Unmarshal an array of char.

6.71 activemq::wireformat::openwire::marshal::BaseDataStreamMarshaller

Class Reference

515

Parameters

<i>dataIn</i>	- the DataInputStream to Un-Marshal from
---------------	--

Returns

the unmarshalled vector of chars.

Exceptions

<i>IOException</i>	if an error occurs.
--------------------	---------------------

6.71.3.11 `virtual commands::DataStructure* activemq::wireformat::openwire::marshal::BaseDataStreamMarshaller::looseUnmarshalCachedObject (OpenWireFormat * wireFormat, decaf::io::DataInputStream * dataIn)` [protected, virtual]

Loose Unmarshal the cached object.

Parameters

<i>wireFormat</i>	- The OpenWireFormat properties
<i>dataIn</i>	- stream to read marshaled form from

Returns

pointer to a new DataStructure Object

Exceptions

<i>IOException</i>	if an error occurs.
--------------------	---------------------

6.71.3.12 `virtual std::vector<unsigned char> activemq::wireformat::openwire::marshal::BaseDataStreamMarshaller::looseUnmarshalConstByteArray (decaf::io::DataInputStream * dataIn, int size)` [protected, virtual]

Tight Unmarshal a fixed size array from that data input stream and return an stl vector of char as the resultant.

Parameters

<i>dataIn</i>	- the DataInputStream to Un-Marshal from
<i>size</i>	- size of the const array to unmarshal

Returns

the unmarshaled vector of chars.

Exceptions

<i>IOException</i>	if an error occurs.
--------------------	---------------------

6.71.3.13 `virtual long long activemq::wireformat::openwire::marshal::BaseDataStreamMarshaller::looseUnmarshalLong (OpenWireFormat * wireFormat, decaf::io::DataInputStream * dataIn) [protected, virtual]`

Loose marshal the long long type.

Parameters

<i>wireFormat</i>	- The OpenWireFormat properties
<i>dataIn</i>	- stream to read marshaled form from

Returns

the unmarshaled long long

Exceptions

<i>IOException</i>	if an error occurs.
--------------------	---------------------

6.71.3.14 `virtual commands::DataStructure* activemq::wireformat::openwire::marshal::BaseDataStreamMarshaller::looseUnmarshalNestedObject (OpenWireFormat * wireFormat, decaf::io::DataInputStream * dataIn) [protected, virtual]`

Loose Unmarshal the nested object.

Parameters

<i>wireFormat</i>	- The OpenWireFormat properties
<i>dataIn</i>	- stream to read marshaled form from

Returns

pointer to a new DataStructure Object

Exceptions

<i>IOException</i>	if an error occurs.
--------------------	---------------------

6.71.3.15 `virtual std::string activemq::wireformat::openwire::marshal::BaseDataStreamMarshaller::looseUnmarshalString (decaf::io::DataInputStream * dataIn)` [protected, virtual]

Loose Un-Marshal the String to the DataOutputStream passed.

Parameters

<i>dataIn</i>	- stream to read marshaled form from
---------------	--------------------------------------

Returns

the unmarshaled string

Exceptions

<i>IOException</i>	if an error occurs.
--------------------	---------------------

6.71.3.16 `virtual std::string activemq::wireformat::openwire::marshal::BaseDataStreamMarshaller::readAsciiString (decaf::io::DataInputStream * dataIn)` [protected, virtual]

Given an DataInputStream read a know ASCII formatted string from the input and return that string.

Parameters

<i>dataIn</i>	- DataInputStream to read from
---------------	--------------------------------

Returns

string value read from stream

6.71.3.17 `virtual int activemq::wireformat::openwire::marshal::BaseDataStreamMarshaller::tightMarshal1 (OpenWireFormat *format AMQCPP_UNUSED, commands::DataStructure *command AMQCPP_UNUSED, utils::BooleanStream *bs AMQCPP_UNUSED)` [inline, virtual]

Tight Marhsal to the given stream.

Parameters

<i>format</i>	- The OpenWireFormat properties
<i>command</i>	- the object to Marshal
<i>bs</i>	- boolean stream to marshal to.

Exceptions

<i>IOException</i>	if an error occurs.
--------------------	---------------------

6.71.3.18 `virtual void activemq::wireformat::openwire::marshal::BaseDataStreamMarshaller::tightMarshal2 (OpenWireFormat *format AMQCPP_UNUSED, commands::DataStructure *command AMQCPP_UNUSED, decaf::io::DataOutputStream *ds AMQCPP_UNUSED, utils::BooleanStream *bs AMQCPP_UNUSED) [inline, virtual]`

Tight Marshal to the given stream.

Parameters

<i>format</i>	- The OpenWireFormat properties
<i>command</i>	- the object to Marshal
<i>ds</i>	- the DataOutputStream to Marshal to
<i>bs</i>	- boolean stream to marshal to.

Exceptions

<i>IOException</i>	if an error occurs.
--------------------	---------------------

6.71.3.19 `virtual int activemq::wireformat::openwire::marshal::BaseDataStreamMarshaller::tightMarshalBrokerError1 (OpenWireFormat * wireFormat, commands::DataStructure * data, utils::BooleanStream * bs) [protected, virtual]`

Tight Marshal the Error object.

Parameters

<i>wireFormat</i>	- The OpenWireFormat properties
<i>data</i>	- Error to Marshal
<i>bs</i>	- boolean stream to marshal to.

Returns

size of the marshalled data

Exceptions

<i>IOException</i>	if an error occurs.
--------------------	---------------------

6.71.3.20 virtual void activemq::wireformat::openwire::marshal::BaseDataStreamMarshaller::tightMarshalBrokerError2 (OpenWireFormat * *wireFormat*, commands::DataStructure * *data*, decaf::io::DataOutputStream * *dataOut*, utils::BooleanStream * *bs*) [protected, virtual]

Tight Marshal the Error object.

Parameters

<i>wireFormat</i>	- The OpenWireFormat properties
<i>data</i>	- Error to Marshal
<i>dataOut</i>	- stream to write marshalled data to
<i>bs</i>	- boolean stream to marshal to.

Exceptions

<i>IOException</i>	if an error occurs.
--------------------	---------------------

6.71.3.21 virtual int activemq::wireformat::openwire::marshal::BaseDataStreamMarshaller::tightMarshalCachedObject1 (OpenWireFormat * *wireFormat*, commands::DataStructure * *data*, utils::BooleanStream * *bs*) [protected, virtual]

Tightly marshals the passed DataStructure based object to the passed BooleanStream returning the size of the data marshaled.

Parameters

<i>wireFormat</i>	- The OpenWireFormat properties
<i>data</i>	- DataStructure Object Pointer to marshal
<i>bs</i>	- boolean stream to marshal to.

Returns

size of data written.

Exceptions

<i>IOException</i>	if an error occurs.
--------------------	---------------------

6.71.3.22 **virtual void activemq::wireformat::openwire::marshal::BaseDataStream-Marshaller::tightMarshalCachedObject2 (OpenWireFormat * *wireFormat*, commands::DataStructure * *data*, decaf::io::DataOutputStream * *dataOut*, utils::BooleanStream * *bs*)** [protected, virtual]

Tightly marshals the passed DataStructure based object to the passed streams returning nothing.

Parameters

<i>wireFormat</i>	- The OpenWireFormat properties
<i>data</i>	- DataStructure Object Pointer to marshal
<i>bs</i>	- boolean stream to marshal to.
<i>dataOut</i>	- stream to write marshaled data to

Exceptions

<i>IOException</i>	if an error occurs.
--------------------	---------------------

6.71.3.23 **virtual int activemq::wireformat::openwire::marshal::BaseDataStream-Marshaller::tightMarshalLong1 (OpenWireFormat * *wireFormat*, long long *value*, utils::BooleanStream * *bs*)** [protected, virtual]

Tightly marshal the long long to the BooleanStream passed.

Parameters

<i>wireFormat</i>	- The OpenWireFormat properties
<i>value</i>	- long long to marshal
<i>bs</i>	- boolean stream to marshal to.

Returns

size of data written.

Exceptions

<i>IOException</i>	if an error occurs.
--------------------	---------------------

6.71.3.24 **virtual void activemq::wireformat::openwire::marshal::BaseDataStream-Marshaller::tightMarshalLong2 (OpenWireFormat * *wireFormat*, long long *value*, decaf::io::DataOutputStream * *dataOut*, utils::BooleanStream * *bs*)** [protected, virtual]

Tightly marshal the long long to the Streams passed.

6.71 activemq::wireformat::openwire::marshal::BaseDataStreamMarshaller

Class Reference

521

Parameters

<i>wireFormat</i>	- The OpenwireFormat properties
<i>value</i>	- long long to marshal
<i>dataOut</i>	- stream to write marshaled form to
<i>bs</i>	- boolean stream to marshal to.

Exceptions

<i>IOException</i>	if an error occurs.
--------------------	---------------------

6.71.3.25 virtual int activemq::wireformat::openwire::marshal::BaseDataStreamMarshaller::tightMarshalNestedObject1 (OpenWireFormat * *wireFormat*, commands::DataStructure * *object*, utils::BooleanStream * *bs*)
[protected, virtual]

Tightly marshals the passed DataStructure based object to the passed BooleanStream returning the size of the data marshaled.

Parameters

<i>wireFormat</i>	- The OpenwireFormat properties
<i>object</i>	- DataStructure Object Pointer to marshal
<i>bs</i>	- boolean stream to marshal to.

Returns

size of data written.

Exceptions

<i>IOException</i>	if an error occurs.
--------------------	---------------------

6.71.3.26 virtual void activemq::wireformat::openwire::marshal::BaseDataStreamMarshaller::tightMarshalNestedObject2 (OpenWireFormat * *wireFormat*, commands::DataStructure * *object*, decaf::io::DataOutputStream * *dataOut*, utils::BooleanStream * *bs*) [protected, virtual]

Tightly marshals the passed DataStructure based object to the passed streams returning nothing.

Parameters

<i>wireFormat</i>	- The OpenwireFormat properties
<i>object</i>	- DataStructure Object Pointer to marshal
<i>bs</i>	- boolean stream to marshal to.
<i>dataOut</i>	- stream to write marshaled data to

Exceptions

<i>IOException</i>	if an error occurs.
--------------------	---------------------

```
6.71.3.27 template<typename T > int activemq::wireformat::openwire-
::marshal::BaseDataStreamMarshaller::tightMarshalObjectArray1
( OpenWireFormat * wireFormat, std::vector< T > objects,
  utils::BooleanStream * bs ) [inline, protected]
```

Tightly Marshal an array of DataStructure objects to the provided boolean stream, and return the size that the tight marshalling is going to take.

Parameters

<i>wireFormat</i>	- The OpenwireFormat properties
<i>objects</i>	- array of DataStructure object pointers.
<i>bs</i>	- boolean stream to marshal to.

Returns

size of the marshalled data

Exceptions

<i>IOException</i>	if an error occurs.
--------------------	---------------------

References AMQ_CATCH_EXCEPTION_CONVERT, AMQ_CATCH_RETHROW, AMQ_CATCHALL_THROW, and activemq::wireformat::openwire::utils::BooleanStream::writeBoolean().

```
6.71.3.28 template<typename T > void activemq::wireformat::openwire-
::marshal::BaseDataStreamMarshaller::tightMarshalObjectArray2
( OpenWireFormat * wireFormat, std::vector< T > objects,
  decaf::io::DataOutputStream * dataOut, utils::BooleanStream * bs )
[inline, protected]
```

Tightly Marshal an array of DataStructure objects to the provided boolean stream and data output stream.

Parameters

<i>wireFormat</i>	- The OpenwireFormat properties
<i>objects</i>	- array of DataStructure object pointers.
<i>dataOut</i>	- stream to write marshalled data to
<i>bs</i>	- boolean stream to marshal to.

Returns

size of the marshalled data

Exceptions

<i>IOException</i>	if an error occurs.
--------------------	---------------------

References AMQ_CATCH_EXCEPTION_CONVERT, AMQ_CATCH_RETHROW, AMQ_CATCHALL_THROW, activemq::wireformat::openwire::utils::BooleanStream::readBoolean(), and decaf::io::DataOutputStream::writeShort().

6.71.3.29 virtual int **activemq::wireformat::openwire::marshal::BaseDataStreamMarshaller::tightMarshalString1** (const std::string & *value*,
utils::BooleanStream * *bs*) [protected, virtual]

Tight Marshals the String to a Booleans Stream Object, returns the marshaled size.

Parameters

<i>value</i>	- string to marshal
<i>bs</i>	- BooleanStream to use.

Returns

size of marshaled string.

Exceptions

<i>IOException</i>	if an error occurs.
--------------------	---------------------

6.71.3.30 virtual void **activemq::wireformat::openwire::marshal::BaseDataStreamMarshaller::tightMarshalString2** (const std::string & *value*,
decaf::io::DataOutputStream * *dataOut*, utils::BooleanStream * *bs*)
[protected, virtual]

Tight Marshals the passed string to the streams passed.

Parameters

<i>value</i>	- string to marshal
<i>dataOut</i>	- the DataOutputStream to Marshal to
<i>bs</i>	- boolean stream to marshal to.

Exceptions

<i>IOException</i>	if an error occurs.
--------------------	---------------------

6.71.3.31 `virtual void activemq::wireformat::openwire::marshal::BaseDataStreamMarshaller::tightUnmarshal (OpenWireFormat *format AMQCPP_UNUSED, commands::DataStructure *command AMQCPP_UNUSED, decaf::io::DataInputStream *dis AMQCPP_UNUSED, utils::BooleanStream *bs AMQCPP_UNUSED) [inline, virtual]`

Tight Un-Marshal to the given stream.

Parameters

<i>format</i>	- The OpenWireFormat properties
<i>command</i>	- the object to Un-Marshal
<i>dis</i>	- the DataInputStream to Un-Marshal from
<i>bs</i>	- boolean stream to Un-Marshal from.

Exceptions

<i>IOException</i>	if an error occurs.
--------------------	---------------------

6.71.3.32 `virtual commands::DataStructure* activemq::wireformat::openwire::marshal::BaseDataStreamMarshaller::tightUnmarshalBrokerError (OpenWireFormat * wireFormat, decaf::io::DataInputStream * dataIn, utils::BooleanStream * bs) [protected, virtual]`

Tight Unmarshal the Error object.

Parameters

<i>wireFormat</i>	- The OpenWireFormat properties
<i>dataIn</i>	- stream to read marshalled form from
<i>bs</i>	- boolean stream to marshal to.

Returns

pointer to a new DataStructure Object

Exceptions

<i>IOException</i>	if an error occurs.
--------------------	---------------------

6.71.3.33 `virtual std::vector<unsigned char> activemq::wireformat::openwire::marshal::BaseDataStreamMarshaller::tightUnmarshalByteArray (decaf::io::DataInputStream * dataIn, utils::BooleanStream * bs) [protected, virtual]`

Tight Unmarshal an array of char.

Parameters

<i>dataIn</i>	- the DataInputStream to Un-Marshal from
<i>bs</i>	- boolean stream to unmarshal from.

Returns

the unmarshaled vector of chars.

Exceptions

<i>IOException</i>	if an error occurs.
--------------------	---------------------

6.71.3.34 `virtual commands::DataStructure* activemq::wireformat::openwire-
::marshal::BaseDataStreamMarshaller::tightUnmarshalCachedObject
(OpenWireFormat * wireFormat, decaf::io::DataInputStream * dataIn,
utils::BooleanStream * bs) [protected, virtual]`

Tight Unmarshal the cached object.

Parameters

<i>wireFormat</i>	- The OpenWireFormat properties
<i>dataIn</i>	- stream to read marshaled form from
<i>bs</i>	- boolean stream to marshal to.

Returns

pointer to a new DataStructure Object

Exceptions

<i>IOException</i>	if an error occurs.
--------------------	---------------------

6.71.3.35 `virtual std::vector<unsigned char> activemq::wireformat::openwire-
::marshal::BaseDataStreamMarshaller::tightUnmarshalConstByteArray
(decaf::io::DataInputStream * dataIn, utils::BooleanStream * bs, int size)
[protected, virtual]`

Tight Unmarshal a fixed size array from that data input stream and return an stl vector of char as the resultant.

Parameters

<i>dataIn</i>	- the DataInputStream to Un-Marshal from
<i>bs</i>	- boolean stream to unmarshal from.
<i>size</i>	- size of the const array to unmarshal

Returns

the unmarshaled vector of chars.

Exceptions

<i>IOException</i>	if an error occurs.
--------------------	---------------------

6.71.3.36 `virtual long long activemq::wireformat::openwire::marshal::BaseDataStreamMarshaller::tightUnmarshalLong (OpenWireFormat * wireFormat, decaf::io::DataInputStream * dataIn, utils::BooleanStream * bs)`
[protected, virtual]

Tight marshal the long long type.

Parameters

<i>wireFormat</i>	- The OpenWireFormat properties
<i>dataIn</i>	- stream to read marshaled form from
<i>bs</i>	- boolean stream to marshal to.

Returns

the unmarshaled long long

Exceptions

<i>IOException</i>	if an error occurs.
--------------------	---------------------

6.71.3.37 `virtual commands::DataStructure* activemq::wireformat::openwire::marshal::BaseDataStreamMarshaller::tightUnmarshalNestedObject (OpenWireFormat * wireFormat, decaf::io::DataInputStream * dataIn, utils::BooleanStream * bs)` [protected, virtual]

Tight Unmarshal the nested object.

Parameters

<i>wireFormat</i>	- The OpenWireFormat properties
<i>dataIn</i>	- stream to read marshaled form from
<i>bs</i>	- boolean stream to marshal to.

Returns

pointer to a new DataStructure Object

Exceptions

<i>IOException</i>	if an error occurs.
--------------------	---------------------

6.71.3.38 virtual std::string activemq::wireformat::openwire::marshal::BaseDataStreamMarshaller::tightUnmarshalString (decaf::io::DataInputStream * *dataIn*, utils::BooleanStream * *bs*) [protected, virtual]

Performs Tight Unmarshaling of String Objects.

Parameters

<i>dataIn</i>	- the DataInputStream to Un-Marshal from
<i>bs</i>	- boolean stream to unmarshal from.

Returns

the unmarshaled string.

Exceptions

<i>IOException</i>	if an error occurs.
--------------------	---------------------

6.71.3.39 static std::string activemq::wireformat::openwire::marshal::BaseDataStreamMarshaller::toHexFromBytes (const std::vector< unsigned char > & *data*) [static]

given an array of bytes, convert that array to a Hexidecimal coded string that represents that data.

Parameters

<i>data</i>	- unsigned char data array pointer
-------------	------------------------------------

Returns

a string coded in hex that represents the data

6.71.3.40 static std::string activemq::wireformat::openwire::marshal::BaseDataStreamMarshaller::toString (const commands::MessageId * *id*) [static]

Converts the object to a String.

Parameters

<i>id</i>	- MessageId pointer
-----------	---------------------

Returns

string representing the id

Referenced by `activemq::commands::ActiveMQMessageTemplate< cms::ObjectMessage >::getCMSMessageID()`.

6.71.3.41 `static std::string activemq::wireformat::openwire::marshal::BaseDataStreamMarshaller::toString (const commands::ProducerId * id)`
[static]

Converts the object to a String.

Parameters

<i>id</i>	- ProducerId pointer
-----------	----------------------

Returns

string representing the id

6.71.3.42 `static std::string activemq::wireformat::openwire::marshal::BaseDataStreamMarshaller::toString (const commands::TransactionId * txnId)`
[static]

Converts the given transaction ID into a String.

Parameters

<i>txnId</i>	- TransactionId pointer
--------------	-------------------------

Returns

string representation of the id

The documentation for this class was generated from the following file:

- `src/main/activemq/wireformat/openwire/marshal/BaseDataStreamMarshaller.h`

6.72 activemq::commands::BaseDataStructure Class Reference

```
#include <src/main/activemq/commands/BaseDataStructure.h>
```

Inheritance diagram for activemq::commands::BaseDataStructure:

Public Member Functions

- virtual **~BaseDataStructure** ()
- virtual bool **isMarshalAware** () const
Determine if the class implementing this interface is really wanting to be told about marshaling.
- virtual void **beforeMarshal** (wireformat::WireFormat *wireFormat AMQCPP_UNUSED)
- virtual void **afterMarshal** (wireformat::WireFormat *wireFormat AMQCPP_UNUSED)
- virtual void **beforeUnmarshal** (wireformat::WireFormat *wireFormat AMQCPP_UNUSED)
- virtual void **afterUnmarshal** (wireformat::WireFormat *wireFormat AMQCPP_UNUSED)
- virtual void **setMarshaledForm** (wireformat::WireFormat *wireFormat AMQCPP_UNUSED, const std::vector< char > &data AMQCPP_UNUSED)
- virtual std::vector< unsigned char > **getMarshaledForm** (wireformat::WireFormat *wireFormat AMQCPP_UNUSED)
- virtual void **copyDataStructure** (const DataStructure *src AMQCPP_UNUSED)
- virtual std::string **toString** () const
Returns a string containing the information for this DataStructure (p. 1133) such as its type and value of its elements.
- virtual bool **equals** (const DataStructure *value AMQCPP_UNUSED) const

6.72.1 Constructor & Destructor Documentation

- 6.72.1.1 virtual activemq::commands::BaseDataStructure::~BaseDataStructure () [inline, virtual]

6.72.2 Member Function Documentation

- 6.72.2.1 virtual void activemq::commands::BaseDataStructure::afterMarshal (wireformat::WireFormat *wireFormat AMQCPP_UNUSED) [inline, virtual]
- 6.72.2.2 virtual void activemq::commands::BaseDataStructure::afterUnmarshal (wireformat::WireFormat *wireFormat AMQCPP_UNUSED) [inline, virtual]

Reimplemented in **activemq::commands::WireFormatInfo** (p. 2892), and **activemq::commands::Message** (p. 1826).

6.72.2.3 `virtual void activemq::commands::BaseDataStructure::beforeMarshal (wireformat::WireFormat *wireFormat AMQCPP_UNUSED) [inline, virtual]`

Reimplemented in `activemq::commands::WireFormatInfo` (p. 2892), and `activemq::commands::Message` (p. 1826).

6.72.2.4 `virtual void activemq::commands::BaseDataStructure::beforeUnmarshal (wireformat::WireFormat *wireFormat AMQCPP_UNUSED) [inline, virtual]`

6.72.2.5 `virtual void activemq::commands::BaseDataStructure::copyDataStructure (const DataStructure *src AMQCPP_UNUSED) [inline, virtual]`

Reimplemented in `activemq::commands::BooleanExpression` (p. 552).

6.72.2.6 `virtual bool activemq::commands::BaseDataStructure::equals (const DataStructure *value AMQCPP_UNUSED) const [inline, virtual]`

Referenced by `activemq::commands::BooleanExpression::equals()`, and `activemq::commands::BaseCommand::equals()`.

6.72.2.7 `virtual std::vector<unsigned char> activemq::commands::BaseDataStructure::getMarshaledForm (wireformat::WireFormat *wireFormat AMQCPP_UNUSED) [inline, virtual]`

6.72.2.8 `virtual bool activemq::commands::BaseDataStructure::isMarshalAware () const [inline, virtual]`

Determine if the class implementing this interface is really wanting to be told about marshaling.

Normally if you didn't want to be marshal aware you just wouldn't implement this interface but since this is C++ and we don't have true interfaces we need a flat inheritance hierarchy, so we always implement this.

Returns

true if this class cares about marshaling.

Implements `activemq::wireformat::MarshalAware` (p. 1795).

Reimplemented in `activemq::commands::Message` (p. 1832), `activemq::commands::WireFormatInfo` (p. 2895), and `activemq::commands::ActiveMQMapMessage` (p. 278).

6.72.2.9 virtual void **activemq::commands::BaseDataStructure::setMarshaledForm**
 (**wireformat::WireFormat** *wireFormat *AMQCPP_UNUSED*, const std::vector<
 char > &data *AMQCPP_UNUSED*) [inline, virtual]

6.72.2.10 virtual std::string **activemq::commands::BaseDataStructure::toString** ()
 const [inline, virtual]

Returns a string containing the information for this **DataStructure** (p. 1133) such as its type and value of its elements.

Returns

formatted string useful for debugging.

Implements **activemq::commands::DataStructure** (p. 1138).

Reimplemented in **activemq::commands::Message** (p. 1836), **activemq::commands::ActiveMQDestination** (p. 255), **activemq::commands::ConsumerInfo** (p. 1016), **activemq::commands::MessageId** (p. 1911), **activemq::commands::SessionId** (p. 2381), **activemq::commands::ActiveMQBytesMessage** (p. 178), **activemq::commands::BrokerInfo** (p. 577), **activemq::commands::ConnectionInfo** (p. 972), **activemq::commands::ProducerId** (p. 2185), **activemq::commands::ConnectionId** (p. 963), **activemq::commands::MessageAck** (p. 1871), **activemq::commands::ConsumerId** (p. 1004), **activemq::commands::ConsumerControl** (p. 995), **activemq::commands::JournalTopicAck** (p. 1572), **activemq::commands::ProducerInfo** (p. 2194), **activemq::commands::ConnectionControl** (p. 942), **activemq::commands::DestinationInfo** (p. 1217), **activemq::commands::MessagePull** (p. 1943), **activemq::commands::SessionInfo** (p. 2389), **activemq::commands::MessageDispatch** (p. 1885), **activemq::commands::MessageDispatchNotification** (p. 1898), **activemq::commands::SubscriptionInfo** (p. 2634), **activemq::commands::XATransactionId** (p. 2942), **activemq::commands::TransactionInfo** (p. 2777), **activemq::commands::ConnectionError** (p. 950), **activemq::commands::JournalQueueAck** (p. 1563), **activemq::commands::JournalTransaction** (p. 1586), **activemq::commands::RemoveSubscriptionInfo** (p. 2279), **activemq::commands::ActiveMQStreamMessage** (p. 369), **activemq::commands::ActiveMQTempDestination** (p. 382), **activemq::commands::LocalTransactionId** (p. 1678), **activemq::commands::NetworkBridgeFilter** (p. 1974), **activemq::commands::ProducerAck** (p. 2174), **activemq::commands::RemoveInfo** (p. 2270), **activemq::commands::ActiveMQMapMessage** (p. 283), **activemq::commands::DataArrayResponse** (p. 1071), **activemq::commands::DataResponse** (p. 1114), **activemq::commands::DiscoveryEvent** (p. 1229), **activemq::commands::ExceptionResponse** (p. 1290), **activemq::commands::PartialCommand** (p. 2078), **activemq::commands::ReplayCommand** (p. 2286), **activemq::commands::BrokerId** (p. 567), **activemq::commands::ControlCommand** (p. 1025), **activemq::commands::IntegerResponse** (p. 1518), **activemq::commands::JournalTrace** (p. 1579), **activemq::commands::Response** (p. 2300), **activemq::commands::FlushCommand** (p. 1383), **activemq::commands::KeepAliveInfo** (p. 1593), **activemq::commands::LastPartialCommand** (p. 1608), **activemq::commands::ShutdownInfo** (p. 2433), **activemq::commands::TransactionId** (p. 2770), **activemq::commands::BaseCommand** (p. 498), **activemq::commands::ActiveMQBlobMessage** (p. 161), **activemq::commands::Command**

(p. 870), `activemq::commands::WireFormatInfo` (p. 2899), `activemq::commands::ActiveMQQueue` (p. 325), `activemq::commands::ActiveMQTopic` (p. 418), `activemq::commands::ActiveMQTextMessage` (p. 409), `activemq::commands::ActiveMQTempQueue` (p. 391), `activemq::commands::ActiveMQTempTopic` (p. 400), `activemq::commands::ActiveMQObjectMessage` (p. 303), `activemq::commands::ActiveMQMessage` (p. 290), and `activemq::commands::BooleanExpression` (p. 552).

Referenced by `activemq::commands::BooleanExpression::toString()`, and `activemq::commands::BaseCommand::toString()`.

The documentation for this class was generated from the following file:

- `src/main/activemq/commands/BaseDataStructure.h`

6.73 decaf::net::BindException Class Reference

```
#include <src/main/decaf/net/BindException.h>
```

Inheritance diagram for `decaf::net::BindException`:

Public Member Functions

- **BindException** () throw ()
Default Constructor.
- **BindException** (const **Exception** &ex) throw ()
Conversion Constructor from some other Exception.
- **BindException** (const **BindException** &ex) throw ()
Copy Constructor.
- **BindException** (const char *file, const int lineNumber, const std::exception *cause, const char *msg,...) throw ()
Constructor - Initializes the file name and line number where this message occurred.
- **BindException** (const std::exception *cause) throw ()
Constructor.
- **BindException** (const char *file, const int lineNumber, const char *msg,...) throw ()
Constructor - Initializes the file name and line number where this message occurred.
- virtual **BindException** * clone () const
Clones this exception.
- virtual ~**BindException** () throw ()

6.73.1 Constructor & Destructor Documentation

6.73.1.1 **decaf::net::BindException::BindException () throw ()** `[inline]`

Default Constructor.

6.73.1.2 **decaf::net::BindException::BindException (const Exception & ex) throw ()**
`[inline]`

Conversion Constructor from some other Exception.

Parameters

<i>ex</i>	An exception that should become this type of Exception
-----------	--

6.73.1.3 **decaf::net::BindException::BindException (const BindException & ex)**
throw () `[inline]`

Copy Constructor.

Parameters

<i>ex</i>	An exception that should become this type of Exception
-----------	--

6.73.1.4 **decaf::net::BindException::BindException (const char * *file*, const int *lineNumber*, const std::exception * *cause*, const char * *msg*, ...) throw ()**
`[inline]`

Constructor - Initializes the file name and line number where this message occurred.

Sets the message to report, using an optional list of arguments to parse into the message

Parameters

<i>file</i>	The file name where exception occurs
<i>lineNumber</i>	The line number where the exception occurred.
<i>cause</i>	The exception that was the cause for this one to be thrown.
<i>msg</i>	The message to report
<i>...</i>	list of primitives that are formatted into the message

6.73.1.5 **decaf::net::BindException::BindException (const std::exception * *cause*)**
throw () `[inline]`

Constructor.

Parameters

<i>cause</i>	Pointer to the exception that caused this one to be thrown, the object is cloned caller retains ownership.
--------------	--

6.73.1.6 `decaf::net::BindException::BindException (const char * file, const int lineNumber, const char * msg, ...) throw () [inline]`

Constructor - Initializes the file name and line number where this message occurred.

Sets the message to report, using an optional list of arguments to parse into the message

Parameters

<i>file</i>	The file name where exception occurs
<i>lineNumber</i>	The line number where the exception occurred.
<i>msg</i>	The message to report
<i>...</i>	list of primitives that are formatted into the message

6.73.1.7 `virtual decaf::net::BindException::~BindException () throw () [inline, virtual]`

6.73.2 Member Function Documentation

6.73.2.1 `virtual BindException* decaf::net::BindException::clone () const [inline, virtual]`

Clones this exception.

This is useful for cases where you need to preserve the type of the original exception as well as the message. All subclasses should override.

Returns

a new Exception instance that is a copy of this Exception object.

Reimplemented from **decaf::net::SocketException** (p. 2473).

The documentation for this class was generated from the following file:

- `src/main/decaf/net/BindException.h`

6.74 decaf::io::BlockingByteArrayInputStream Class Reference

This is a blocking version of a byte buffer stream.

```
#include <src/main/decaf/io/BlockingByteArrayInputStream.h>
```

Inheritance diagram for decaf::io::BlockingByteArrayInputStream:

Public Member Functions

- **BlockingByteArrayInputStream** ()

Default Constructor - uses a default internal buffer.

- **BlockingByteArrayInputStream** (const unsigned char *buffer, int bufferSize)

Constructor that initializes the internal buffer.

- virtual ~**BlockingByteArrayInputStream** ()

- virtual void **setByteArray** (const unsigned char *buffer, int bufferSize)

- virtual int **available** () const

Indicates the number of bytes available.

The default implementation of this methods returns 0. Classes that override this method may return the total number of bytes that are currently available for reading and others may simply return a value of one indicating that there is some data available. The caller should view the result of this method as an absolute.

The default implementation of this method returns zero.

Returns

the number of bytes available on this input stream.

Exceptions

IOException (p. 1545)	<i>if an I/O error occurs.</i>
------------------------------	--------------------------------

- virtual void **close** ()

*Closes the **InputStream** (p. 1464) freeing any resources that might have been acquired during the lifetime of this stream.*

The default implementation of this method does nothing.

Exceptions

IOException (p. 1545)	<i>if an I/O error occurs while closing the InputStream (p. 1464).</i>
------------------------------	---

- virtual long long **skip** (long long num)

Skips over and discards n bytes of data from this input stream.

The skip method may, for a variety of reasons, end up skipping over some smaller number of bytes, possibly 0. This may result from any of a number of conditions; reaching end of file before n bytes have been skipped is only one possibility. The actual number of bytes skipped is returned.

*The skip method of **InputStream** (p. 1464) creates a byte array and then repeatedly reads into it until num bytes have been read or the end of the stream has been reached. Subclasses are encouraged to provide a more efficient implementation of this method.*

Parameters

num	<i>The number of bytes to skip.</i>
-----	-------------------------------------

Returns

total bytes skipped

Exceptions

IOException (p. 1545)	<i>if an I/O error occurs.</i>
UnsupportedOperation-Exception	<i>if the concrete stream class does not support skipping bytes.</i>

Protected Member Functions

- virtual int **doReadByte** ()
- virtual int **doReadArrayBounded** (unsigned char *buffer, int size, int offset, int length)

6.74.1 Detailed Description

This is a blocking version of a byte buffer stream.

Read operations block until the requested data becomes available in the internal buffer via a call to `setByteArray`.

6.74.2 Constructor & Destructor Documentation

6.74.2.1 `decaf::io::BlockingByteArrayInputStream::BlockingByteArrayInputStream ()`

Default Constructor - uses a default internal buffer.

6.74.2.2 `decaf::io::BlockingByteArrayInputStream::BlockingByteArrayInputStream (const unsigned char * buffer, int bufferSize)`

Constructor that initializes the internal buffer.

See also

setByteArray (p. 537).

6.74.2.3 `virtual decaf::io::BlockingByteArrayInputStream::~~BlockingByteArrayInputStream () [virtual]`

6.74.3 Member Function Documentation

6.74.3.1 `virtual int decaf::io::BlockingByteArrayInputStream::available () const [virtual]`

Indicates the number of bytes available.

The default implementation of this methods returns 0. Classes that override this method may return the total number of bytes that are currently available for reading and others

may simply return a value of one indicating that there is some data available. The caller should view the result of this method as an absolute.

The default implementation of this method returns zero.

Returns

the number of bytes available on this input stream.

Exceptions

<i>IOException</i> (p. 1545)	if an I/O error occurs.
--	-------------------------

Reimplemented from **decaf::io::InputStream** (p. 1466).

6.74.3.2 virtual void **decaf::io::BlockingByteArrayInputStream::close** ()
[virtual]

Closes the **InputStream** (p. 1464) freeing any resources that might have been acquired during the lifetime of this stream.

The default implementation of this method does nothing.

Exceptions

<i>IOException</i> (p. 1545)	if an I/O error occurs while closing the InputStream (p. 1464).
--	--

Reimplemented from **decaf::io::InputStream** (p. 1466).

6.74.3.3 virtual int **decaf::io::BlockingByteArrayInputStream::doRead-
ArrayBounded** (unsigned char * *buffer*, int *size*, int *offset*, int *length*)
[protected, virtual]

Reimplemented from **decaf::io::InputStream** (p. 1467).

6.74.3.4 virtual int **decaf::io::BlockingByteArrayInputStream::doReadByte** ()
[protected, virtual]

Implements **decaf::io::InputStream** (p. 1467).

6.74.3.5 virtual void **decaf::io::BlockingByteArrayInputStream::setByteArray** (const
unsigned char * *buffer*, int *bufferSize*) [virtual]

6.74.3.6 virtual long long decaf::io::BlockingByteArrayInputStream::skip (long long num) [virtual]

Skips over and discards n bytes of data from this input stream.

The skip method may, for a variety of reasons, end up skipping over some smaller number of bytes, possibly 0. This may result from any of a number of conditions; reaching end of file before n bytes have been skipped is only one possibility. The actual number of bytes skipped is returned.

The skip method of **InputStream** (p. 1464) creates a byte array and then repeatedly reads into it until num bytes have been read or the end of the stream has been reached. Subclasses are encouraged to provide a more efficient implementation of this method.

Parameters

<i>num</i>	The number of bytes to skip.
------------	------------------------------

Returns

total bytes skipped

Exceptions

<i>IOException</i> (p. 1545)	if an I/O error occurs.
<i>Unsupported-OperationException</i>	if the concrete stream class does not support skipping bytes.

Reimplemented from **decaf::io::InputStream** (p. 1472).

The documentation for this class was generated from the following file:

- src/main/decaf/io/**BlockingByteArrayInputStream.h**

6.75 decaf::util::concurrent::BlockingQueue< E > Class Template Reference

A **decaf::util::Queue** (p. 2222) that additionally supports operations that wait for the queue to become non-empty when retrieving an element, and wait for space to become available in the queue when storing an element.

```
#include <src/main/decaf/util/concurrent/BlockingQueue.h>
```

Inheritance diagram for decaf::util::concurrent::BlockingQueue< E >:

6.75 decaf::util::concurrent::BlockingQueue< E > Class Template Reference 539

Public Member Functions

- virtual **~BlockingQueue** ()
- virtual void **put** (const E &value)=0
Inserts the specified element into this queue, waiting if necessary for space to become available.
- virtual bool **offer** (const E &e, long long timeout, const **TimeUnit** &unit)=0
Inserts the specified element into this queue, waiting up to the specified wait time if necessary for space to become available.
- virtual E **take** ()=0
Retrieves and removes the head of this queue, waiting if necessary until an element becomes available.
- virtual bool **poll** (E &result, long long timeout, const **TimeUnit** &unit)=0
Retrieves and removes the head of this queue, waiting up to the specified wait time if necessary for an element to become available.
- virtual int **remainingCapacity** () const =0
Returns the number of additional elements that this queue can ideally (in the absence of memory or resource constraints) accept without blocking, or `Integer::MAX_VALUE` if there is no intrinsic limit.
- virtual int **drainTo** (**Collection**< E > &c)=0
Removes all available elements from this queue and adds them to the given collection.
- virtual int **drainTo** (**Collection**< E > &c, int maxElements)=0
Removes at most the given number of available elements from this queue and adds them to the given collection.

6.75.1 Detailed Description

template<typename E>class decaf::util::concurrent::BlockingQueue< E >

A **decaf::util::Queue** (p. 2222) that additionally supports operations that wait for the queue to become non-empty when retrieving an element, and wait for space to become available in the queue when storing an element.

BlockingQueue (p. 538) methods come in four forms, with different ways of handling operations that cannot be satisfied immediately, but may be satisfied at some point in the future: one throws an exception, the second returns a special value (either `true` or `false`, depending on the operation), the third blocks the current thread indefinitely until the operation can succeed, and the fourth blocks for only a given maximum time limit before giving up. These methods are summarized in the following table:

	<i>Throws exception</i>	<i>Boolean Flag</i>	<i>Blocks</i>	<i>Times out</i>
Insert	add(e) (p. 139)	offer(e) (p. 543)	put(e) (p. 544)	offer(e, time, unit) (p. ??)
Remove	remove() (p. 142)	poll() (p. 543)	take() (p. 545)	poll(time, unit) (p. ??)
Examine	element() (p. 142)	peek() (p. 2225)	<i>not applicable</i>	<i>not applicable</i>

A **BlockingQueue** (p. 538) may be capacity bounded. At any given time it may have a `remainingCapacity` beyond which no additional elements can be put without blocking. A **BlockingQueue** (p. 538) without any intrinsic capacity constraints always reports a remaining capacity of `Integer::MAX_VALUE`.

BlockingQueue (p. 538) implementations are designed to be used primarily for producer-consumer queues, but additionally support **decaf::util::Collection** (p. 851) interface. So, for example, it is possible to remove an arbitrary element from a queue using `remove(x)`. However, such operations are in general *not* performed very efficiently, and are intended for only occasional use, such as when a queued message is cancelled.

BlockingQueue (p. 538) implementations are thread-safe. All queuing methods achieve their effects atomically using internal locks or other forms of concurrency control. However, the *bulk Collection* (p. 851) operations `addAll`, `containsAll`, `retainAll` and `removeAll` are *not* necessarily performed atomically unless specified otherwise in an implementation. So it is possible, for example, for `addAll(c)` to fail (throwing an exception) after adding only some of the elements in `c`.

A **BlockingQueue** (p. 538) does *not* intrinsically support any kind of "close" or "shutdown" operation to indicate that no more items will be added. The needs and usage of such features tend to be implementation-dependent. For example, a common tactic is for producers to insert special *end-of-stream* or *poison* objects, that are interpreted accordingly when taken by consumers.

Usage example, based on a typical producer-consumer scenario. Note that a **BlockingQueue** (p. 538) can safely be used with multiple producers and multiple consumers.

```
class Producer : public Runnable {
private:

    BlockingQueue* queue;

public:

    Producer( BlockingQueue* q ) : queue( q ) {}

    virtual void run() {
        try {
            while( true ) { queue->put( produce() ); }
        } catch( InterruptedException& ex ) { ... handle ...}
    }

    Object produce() { ... }
}

class Consumer : public Runnable {
private:

    BlockingQueue* queue;
```

6.75 decaf::util::concurrent::BlockingQueue< E > Class Template Reference 541

```
public:

    Consumer( BlockingQueue* q ) : queue( q ) {}

    virtual void run() {
        try {
            while( true ) { consume( queue->take() (p.545) ); }
        } catch( InterruptedException& ex ) { ... handle ...}
    }

    void consume( Object& x ) { ... }
}

int main( int argc, char** argv ) {

    BlockingQueue (p.538) q = new SomeQueueImplementation();
    Producer p( &q );
    Consumer c1( &q );
    Consumer c2( &q );
    Thread t1( &p ).start();
    Thread t2( &c1 ).start();
    Thread t3( &c2 ).start();
}
```

Memory consistency effects: As with other concurrent collections, actions in a thread prior to placing an object into a **BlockingQueue** (p.538) *happen-before* actions subsequent to the access or removal of that element from the **BlockingQueue** (p.538) in another thread.

Since

1.0

6.75.2 Constructor & Destructor Documentation

6.75.2.1 `template<typename E> virtual decaf::util::concurrent::BlockingQueue< E >::~BlockingQueue()` [inline, virtual]

6.75.3 Member Function Documentation

6.75.3.1 `template<typename E> virtual int decaf::util::concurrent::BlockingQueue< E >::drainTo(Collection< E > & c)` [pure virtual]

Removes all available elements from this queue and adds them to the given collection.

This operation may be more efficient than repeatedly polling this queue. A failure encountered while attempting to add elements to collection *c* may result in elements being in neither, either or both collections when the associated exception is thrown. Attempts to drain a queue to itself result in `IllegalArgumentException`. Further, the behavior of this operation is undefined if the specified collection is modified while the operation is in progress.

Parameters

<i>c</i>	the collection to transfer elements into
----------	--

Returns

the number of elements transferred

Exceptions

<i>Unsupported- OperationException</i>	if addition of elements is not supported by the specified collection
<i>IllegalArgument- Exception</i>	if the specified collection is this queue, or some property of an element of this queue prevents it from being added to the specified collection

Implemented in **decaf::util::concurrent::LinkedBlockingQueue< E >** (p. 1625), and **decaf::util::concurrent::SynchronousQueue< E >** (p. 2659).

6.75.3.2 `template<typename E> virtual int decaf::util::concurrent::BlockingQueue< E >::drainTo (Collection< E > & c, int maxElements) [pure virtual]`

Removes at most the given number of available elements from this queue and adds them to the given collection.

A failure encountered while attempting to add elements to collection *c* may result in elements being in neither, either or both collections when the associated exception is thrown. Attempts to drain a queue to itself result in `IllegalArgumentException`. Further, the behavior of this operation is undefined if the specified collection is modified while the operation is in progress.

Parameters

<i>c</i>	the collection to transfer elements into
<i>max- Elements</i>	the maximum number of elements to transfer

Returns

the number of elements transferred

Exceptions

<i>Unsupported- OperationException</i>	if addition of elements is not supported by the specified collection
<i>IllegalArgument- Exception</i>	if the specified collection is this queue, or some property of an element of this queue prevents it from being added to the specified collection

6.75 `decaf::util::concurrent::BlockingQueue< E >` Class Template Reference 543

Implemented in `decaf::util::concurrent::LinkedBlockingQueue< E >` (p. 1626), and `decaf::util::concurrent::SynchronousQueue< E >` (p. 2659).

6.75.3.3 `template<typename E> virtual bool decaf::util::concurrent::BlockingQueue< E >::offer (const E & e, long long timeout, const TimeUnit & unit)` [pure virtual]

Inserts the specified element into this queue, waiting up to the specified wait time if necessary for space to become available.

Parameters

<i>e</i>	the element to add
<i>timeout</i>	how long to wait before giving up, in units of <i>unit</i>
<i>unit</i>	a TimeUnit (p. 2756) determining how to interpret the <i>timeout</i> parameter

Returns

`true` if successful, or `false` if the specified waiting time elapses before space is available

Exceptions

<i>InterruptedException</i>	if interrupted while waiting
<i>NullPointerException</i>	if the specified element is null
<i>IllegalArgumentException</i>	if some property of the specified element prevents it from being added to this queue

Implemented in `decaf::util::concurrent::LinkedBlockingQueue< E >` (p. 1627), and `decaf::util::concurrent::SynchronousQueue< E >` (p. 2661).

6.75.3.4 `template<typename E> virtual bool decaf::util::concurrent::BlockingQueue< E >::poll (E & result, long long timeout, const TimeUnit & unit)` [pure virtual]

Retrieves and removes the head of this queue, waiting up to the specified wait time if necessary for an element to become available.

Parameters

<i>result</i>	the referenced value that will be assigned the value retrieved from the Queue (p. 2222). Undefined if this methods returned false.
<i>timeout</i>	how long to wait before giving up, in units of <i>unit</i>
<i>unit</i>	a TimeUnit (p. 2756) determining how to interpret the <i>timeout</i> parameter.

Returns

`true` if successful or `false` if the specified waiting time elapses before an element is available.

Exceptions

<i>InterruptedException</i>	if interrupted while waiting
-----------------------------	------------------------------

Implemented in **decaf::util::concurrent::LinkedBlockingQueue< E >** (p. 1629), and **decaf::util::concurrent::SynchronousQueue< E >** (p. 2662).

6.75.3.5 `template<typename E> virtual void decaf::util::concurrent::BlockingQueue< E >::put (const E & value) [pure virtual]`

Inserts the specified element into this queue, waiting if necessary for space to become available.

Parameters

<i>value</i>	the element to add
--------------	--------------------

Exceptions

<i>InterruptedException</i>	if interrupted while waiting
<i>NullPointerException</i>	if the specified element is null
<i>IllegalArgumentException</i>	if some property of the specified element prevents it from being added to this queue

Implemented in **decaf::util::concurrent::LinkedBlockingQueue< E >** (p. 1630), and **decaf::util::concurrent::SynchronousQueue< E >** (p. 2663).

6.75.3.6 `template<typename E> virtual int decaf::util::concurrent::BlockingQueue< E >::remainingCapacity () const [pure virtual]`

Returns the number of additional elements that this queue can ideally (in the absence of memory or resource constraints) accept without blocking, or `Integer::MAX_VALUE` if there is no intrinsic limit.

Note that you *cannot* always tell if an attempt to insert an element will succeed by inspecting `remainingCapacity` because it may be the case that another thread is about to insert or remove an element.

Returns

the remaining capacity

Implemented in **decaf::util::concurrent::LinkedBlockingQueue< E >** (p. 1630), and **decaf::util::concurrent::SynchronousQueue< E >** (p. 2664).

6.75.3.7 `template<typename E> virtual E decaf::util::concurrent::BlockingQueue< E >::take () [pure virtual]`

Retrieves and removes the head of this queue, waiting if necessary until an element becomes available.

Returns

the head of this queue

Exceptions

<i>InterruptedException</i>	if interrupted while waiting
-----------------------------	------------------------------

Implemented in `decaf::util::concurrent::LinkedBlockingQueue< E >` (p. 1632), and `decaf::util::concurrent::SynchronousQueue< E >` (p. 2664).

The documentation for this class was generated from the following file:

- `src/main/decaf/util/concurrent/BlockingQueue.h`

6.76 decaf::lang::Boolean Class Reference

```
#include <src/main/decaf/lang/Boolean.h>
```

Inheritance diagram for `decaf::lang::Boolean`:

Public Member Functions

- **Boolean** (bool value)
- **Boolean** (const std::string &value)
- virtual \sim **Boolean** ()
- bool **booleanValue** () const
- std::string **toString** () const
- virtual int **compareTo** (const **Boolean** &b) const
*Compares this **Boolean** (p. 545) instance with another.*
- virtual bool **operator==** (const **Boolean** &value) const
Compares equality between this object and the one passed.
- virtual bool **operator<** (const **Boolean** &value) const
Compares this object to another and returns true if this object is considered to be less than the one passed.
- bool **equals** (const **Boolean** &b) const
- virtual int **compareTo** (const bool &b) const
*Compares this **Boolean** (p. 545) instance with another.*

- virtual bool **operator==** (const bool &value) const
Compares equality between this object and the one passed.
- virtual bool **operator<** (const bool &value) const
Compares this object to another and returns true if this object is considered to be less than the one passed.
- bool **equals** (const bool &b) const

Static Public Member Functions

- static **Boolean valueOf** (bool value)
- static **Boolean valueOf** (const std::string &value)
- static bool **parseBoolean** (const std::string &value)
*Parses the **String** (p. 2620) passed and extracts an bool.*
- static std::string **toString** (bool value)
*Converts the bool to a **String** (p. 2620) representation.*

Static Public Attributes

- static const **Boolean _FALSE**
The Class object representing the primitive false boolean.
- static const **Boolean _TRUE**
The Class object representing the primitive type boolean.

6.76.1 Constructor & Destructor Documentation

6.76.1.1 decaf::lang::Boolean::Boolean (bool value)

Parameters

<i>value</i>	- primitive boolean to wrap.
--------------	------------------------------

6.76.1.2 decaf::lang::Boolean::Boolean (const std::string & value)

Parameters

<i>value</i>	- String (p. 2620) value to convert to a boolean.
--------------	--

6.76.1.3 virtual decaf::lang::Boolean::~~Boolean () [inline, virtual]

6.76.2 Member Function Documentation

6.76.2.1 bool decaf::lang::Boolean::booleanValue () const [inline]

Returns

the primitive boolean value of this object

6.76.2.2 `virtual int decaf::lang::Boolean::compareTo (const Boolean & b) const`
[virtual]

Compares this **Boolean** (p. 545) instance with another.

Parameters

<i>b</i>	- the Boolean (p. 545) instance to be compared
----------	---

Returns

zero if this object represents the same boolean value as the argument; a positive value if this object represents true and the argument represents false; and a negative value if this object represents false and the argument represents true

Implements **decaf::lang::Comparable**< **Boolean** > (p. 886).

6.76.2.3 `virtual int decaf::lang::Boolean::compareTo (const bool & b) const`
[virtual]

Compares this **Boolean** (p. 545) instance with another.

Parameters

<i>b</i>	- the Boolean (p. 545) instance to be compared
----------	---

Returns

zero if this object represents the same boolean value as the argument; a positive value if this object represents true and the argument represents false; and a negative value if this object represents false and the argument represents true

Implements **decaf::lang::Comparable**< **bool** > (p. 886).

6.76.2.4 `bool decaf::lang::Boolean::equals (const Boolean & b) const` [inline, virtual]

Returns

true if the two **Boolean** (p. 545) Objects have the same value.

Implements **decaf::lang::Comparable**< **Boolean** > (p. 887).

6.76.2.5 `bool decaf::lang::Boolean::equals (const bool & b) const` `[inline, virtual]`

Returns

true if the two **Boolean** (p. 545) Objects have the same value.

Implements **decaf::lang::Comparable**< **bool** > (p. 887).

6.76.2.6 `virtual bool decaf::lang::Boolean::operator< (const Boolean & value) const` `[virtual]`

Compares this object to another and returns true if this object is considered to be less than the one passed.

This

Parameters

<i>value</i>	- the value to be compared to this one.
--------------	---

Returns

true if this object is equal to the one passed.

Implements **decaf::lang::Comparable**< **Boolean** > (p. 887).

6.76.2.7 `virtual bool decaf::lang::Boolean::operator< (const bool & value) const` `[virtual]`

Compares this object to another and returns true if this object is considered to be less than the one passed.

This

Parameters

<i>value</i>	- the value to be compared to this one.
--------------	---

Returns

true if this object is equal to the one passed.

Implements **decaf::lang::Comparable**< **bool** > (p. 887).

6.76.2.8 `virtual bool decaf::lang::Boolean::operator== (const Boolean & value) const` `[virtual]`

Compares equality between this object and the one passed.

Parameters

<i>value</i>	- the value to be compared to this one.
--------------	---

Returns

true if this object is equal to the one passed.

Implements **decaf::lang::Comparable**< **Boolean** > (p. 888).

6.76.2.9 `virtual bool decaf::lang::Boolean::operator==(const bool & value) const`
[virtual]

Compares equality between this object and the one passed.

Parameters

<i>value</i>	- the value to be compared to this one.
--------------	---

Returns

true if this object is equal to the one passed.

Implements **decaf::lang::Comparable**< **bool** > (p. 888).

6.76.2.10 `static bool decaf::lang::Boolean::parseBoolean (const std::string & value)`
[static]

Parses the **String** (p. 2620) passed and extracts an bool.

Parameters

<i>value</i>	The std::string value to parse
--------------	--------------------------------

Returns

bool value

6.76.2.11 `std::string decaf::lang::Boolean::toString () const`

Returns

the string representation of this Booleans value.

6.76.2.12 `static std::string decaf::lang::Boolean::toString (bool value)` [static]

Converts the bool to a **String** (p. 2620) representation.

Parameters

<i>value</i>	The bool value to convert.
--------------	----------------------------

Returns

std::string representation of the bool value passed.

6.76.2.13 static Boolean decaf::lang::Boolean::valueOf (bool *value*) [static]

Parameters

<i>value</i>	The bool value to convert to a Boolean (p. 545) instance.
--------------	--

Returns

a **Boolean** (p. 545) instance of the primitive boolean value

6.76.2.14 static Boolean decaf::lang::Boolean::valueOf (const std::string & *value*) [static]

Parameters

<i>value</i>	The std::string value to convert to a Boolean (p. 545) instance.
--------------	---

Returns

a **Boolean** (p. 545) instance of the string value

6.76.3 Field Documentation

6.76.3.1 const Boolean decaf::lang::Boolean::_FALSE [static]

The Class object representing the primitive false boolean.

6.76.3.2 const Boolean decaf::lang::Boolean::_TRUE [static]

The Class object representing the primitive type boolean.

The documentation for this class was generated from the following file:

- src/main/decaf/lang/**Boolean.h**

6.77 activemq::commands::BooleanExpression Class Reference

```
#include <src/main/activemq/commands/BooleanExpression.h>
```

Inheritance diagram for `activemq::commands::BooleanExpression`:

Public Member Functions

- **BooleanExpression** ()
- virtual **~BooleanExpression** () throw ()
- virtual **DataStructure** * **cloneDataStructure** () const
Clone this object and return a new instance that the caller now owns, this will be an exact copy of this one.
- virtual void **copyDataStructure** (const **DataStructure** *src AMQCPP_UNUSED)
- virtual std::string **toString** () const
*Returns a string containing the information for this **DataStructure** (p. 1133) such as its type and value of its elements.*
- virtual bool **equals** (const **DataStructure** *value) const
*Compares the **DataStructure** (p. 1133) passed in to this one, and returns if they are equivalent.*

6.77.1 Constructor & Destructor Documentation

6.77.1.1 `activemq::commands::BooleanExpression::BooleanExpression ()`
`[inline]`

6.77.1.2 `virtual activemq::commands::BooleanExpression::~~BooleanExpression () throw ()` `[inline, virtual]`

6.77.2 Member Function Documentation

6.77.2.1 `virtual DataStructure* activemq::commands::BooleanExpression::cloneDataStructure () const` `[inline, virtual]`

Clone this object and return a new instance that the caller now owns, this will be an exact copy of this one.

Returns

new copy of this object.

Implements `activemq::commands::DataStructure` (p. 1133).

```
6.77.2.2 virtual void activemq::commands::BooleanExpression::copyData-
          Structure ( const DataStructure *src AMQCPP_UNUSED ) [inline,
          virtual]
```

Reimplemented from **activemq::commands::BaseDataStructure** (p. 530).

```
6.77.2.3 virtual bool activemq::commands::BooleanExpression::equals ( const
          DataStructure * value ) const [inline, virtual]
```

Compares the **DataStructure** (p. 1133) passed in to this one, and returns if they are equivalent.

Equivalent here means that they are of the same type, and that each element of the objects are the same.

Returns

true if **DataStructure** (p. 1133)'s are Equal.

Implements **activemq::commands::DataStructure** (p. 1135).

References **activemq::commands::BaseDataStructure::equals()**.

```
6.77.2.4 virtual std::string activemq::commands::BooleanExpression::toString ( )
          const [inline, virtual]
```

Returns a string containing the information for this **DataStructure** (p. 1133) such as its type and value of its elements.

Returns

formatted string useful for debugging.

Reimplemented from **activemq::commands::BaseDataStructure** (p. 531).

References **activemq::commands::BaseDataStructure::toString()**.

The documentation for this class was generated from the following file:

- `src/main/activemq/commands/BooleanExpression.h`

6.78 activemq::wireformat::openwire::utils::BooleanStream Class Reference

Manages the writing and reading of boolean data streams to and from a data source such as a `DataInputStream` or `DataOutputStream`.

```
#include <src/main/activemq/wireformat/openwire/utils/-
BooleanStream.h>
```

Public Member Functions

- **BooleanStream** ()
- virtual **~BooleanStream** () throw ()
- bool **readBoolean** ()
Read a boolean data element from the internal data buffer.
- void **writeBoolean** (bool value)
Writes a Boolean value to the internal data buffer.
- void **marshal** (decaf::io::DataOutputStream *dataOut)
Marshal the data to a DataOutputStream.
- void **marshal** (std::vector< unsigned char > &dataOut)
Marshal the data to a STL vector of unsigned chars.
- void **unmarshal** (decaf::io::DataInputStream *dataIn)
Unmarshal a Boolean data stream from the Input Stream.
- void **clear** ()
Clears to old position markers, data starts at the beginning.
- int **marshalledSize** ()
Calc the size that data is marshalled to.

6.78.1 Detailed Description

Manages the writing and reading of boolean data streams to and from a data source such as a DataInputStream or DataOutputStream.

The booleans are stored as single bits in the stream, with the stream size pre-pended to the stream when the data is marshalled.

The serialized form of the size field can be between 1 and 3 bytes. If the number of used bytes < 64, size=1 byte If the number of used bytes >=64 and < 256 (size of an unsigned byte), size=2 bytes If the number of used bytes >=256, size=3 bytes

The high-order 2 bits (128 and 64) of the first byte of the size field are used to encode the information about the number of bytes in the size field. The only time the first byte will contain a value >=64 is if there are more bytes in the size field. If the first byte < 64, the value of the byte is simply the size value. If the first byte = 0xC0, the following unsigned byte is the size field. If the first byte = 0x80, the following short (two bytes) are the size field.

6.78.2 Constructor & Destructor Documentation

6.78.2.1 **activemq::wireformat::openwire::utils::BooleanStream::BooleanStream** ()

6.78.2.2 **virtual activemq::wireformat::openwire::utils::BooleanStream::~~BooleanStream** () throw () [virtual]

6.78.3 Member Function Documentation

6.78.3.1 void activemq::wireformat::openwire::utils::BooleanStream::clear ()

Clears to old position markers, data starts at the beginning.

6.78.3.2 void activemq::wireformat::openwire::utils::BooleanStream::marshal (decaf::io::DataOutputStream * dataOut)

Marshal the data to a DataOutputStream.

Parameters

<i>dataOut</i>	- Stream to write the data to.
----------------	--------------------------------

Exceptions

<i>IOException</i>	if an I/O error occurs during this operation.
--------------------	---

6.78.3.3 void activemq::wireformat::openwire::utils::BooleanStream::marshal (std::vector< unsigned char > & dataOut)

Marshal the data to a STL vector of unsigned chars.

Parameters

<i>dataOut</i>	- reference to a vector to write the data to.
----------------	---

Exceptions

<i>IOException</i>	if an I/O error occurs during this operation.
--------------------	---

6.78.3.4 int activemq::wireformat::openwire::utils::BooleanStream::marshalled-Size ()

Calc the size that data is marshalled to.

Returns

int size of marshalled data.

6.78.3.5 bool activemq::wireformat::openwire::utils::BooleanStream::read-Boolean ()

Read a boolean data element from the internal data buffer.

6.78 activemq::wireformat::openwire::utils::BooleanStream Class Reference 555

Returns

boolean from the stream

Exceptions

<i>IOException</i>	if an I/O error occurs during this operation.
--------------------	---

Referenced by `activemq::wireformat::openwire::marshal::BaseDataStreamMarshaller::tightMarshalObjectArray2()`.

6.78.3.6 void `activemq::wireformat::openwire::utils::BooleanStream::unmarshal (decaf::io::DataInputStream * dataIn)`

Unmarshal a Boolean data stream from the Input Stream.

Parameters

<i>dataIn</i>	- Input Stream to read data from.
---------------	-----------------------------------

Exceptions

<i>IOException</i>	if an I/O error occurs during this operation.
--------------------	---

6.78.3.7 void `activemq::wireformat::openwire::utils::BooleanStream::write-Boolean (bool value)`

Writes a Boolean value to the internal data buffer.

Parameters

<i>value</i>	- boolean data to write.
--------------	--------------------------

Exceptions

<i>IOException</i>	if an I/O error occurs during this operation.
--------------------	---

Referenced by `activemq::wireformat::openwire::marshal::BaseDataStreamMarshaller::tightMarshalObjectArray1()`.

The documentation for this class was generated from the following file:

- `src/main/activemq/wireformat/openwire/utils/BooleanStream.h`

6.79 decaf::util::concurrent::BrokenBarrierException Class - Reference

```
#include <src/main/decaf/util/concurrent/BrokenBarrierException.h>
```

Inheritance diagram for decaf::util::concurrent::BrokenBarrierException:

Public Member Functions

- **BrokenBarrierException** () throw ()
Default Constructor.
- **BrokenBarrierException** (const **decaf::lang::Exception** &ex) throw ()
Conversion Constructor from some other Exception.
- **BrokenBarrierException** (const **BrokenBarrierException** &ex) throw ()
Copy Constructor.
- **BrokenBarrierException** (const std::exception ***cause**) throw ()
Constructor.
- **BrokenBarrierException** (const char *file, const int lineNumber, const char *msg,...) throw ()
Constructor - Initializes the file name and line number where this message occurred.
- **BrokenBarrierException** (const char *file, const int lineNumber, const std::exception ***cause**, const char *msg,...) throw ()
Constructor - Initializes the file name and line number where this message occurred.
- virtual **BrokenBarrierException** * **clone** () const
Clones this exception.
- virtual ~**BrokenBarrierException** () throw ()

6.79.1 Constructor & Destructor Documentation

6.79.1.1 decaf::util::concurrent::BrokenBarrierException::BrokenBarrierException () throw () [inline]

Default Constructor.

6.79.1.2 decaf::util::concurrent::BrokenBarrierException::BrokenBarrierException (const decaf::lang::Exception & ex) throw () [inline]

Conversion Constructor from some other Exception.

Parameters

<i>ex</i>	An exception that should become this type of Exception
-----------	--

6.79.1.3 decaf::util::concurrent::BrokenBarrierException::BrokenBarrierException (const BrokenBarrierException & *ex*) throw ()
`[inline]`

Copy Constructor.

Parameters

<i>ex</i>	The Exception to copy in this new instance.
-----------	---

6.79.1.4 decaf::util::concurrent::BrokenBarrierException::BrokenBarrierException (const std::exception * *cause*) throw ()
`[inline]`

Constructor.

Parameters

<i>cause</i>	Pointer to the exception that caused this one to be thrown, the object is cloned caller retains ownership.
--------------	--

6.79.1.5 decaf::util::concurrent::BrokenBarrierException::BrokenBarrierException (const char * *file*, const int *lineNumber*, const char * *msg*, ...) throw ()
`[inline]`

Constructor - Initializes the file name and line number where this message occurred.

Sets the message to report, using an optional list of arguments to parse into the message

Parameters

<i>file</i>	- The file name where exception occurs
<i>lineNumber</i>	- The line number where the exception occurred.
<i>msg</i>	- The message to report
<i>...</i>	- list of primitives that are formatted into the message

6.79.1.6 decaf::util::concurrent::BrokenBarrierException::BrokenBarrierException (const char * *file*, const int *lineNumber*, const std::exception * *cause*, const char * *msg*, ...) throw () `[inline]`

Constructor - Initializes the file name and line number where this message occurred.

Sets the message to report, using an optional list of arguments to parse into the message

Parameters

<i>file</i>	- The file name where exception occurs
<i>lineNumber</i>	- The line number where the exception occurred.
<i>cause</i>	- The exception that was the cause for this one to be thrown.
<i>msg</i>	- The message to report
<i>...</i>	- list of primitives that are formatted into the message

6.79.1.7 `virtual decaf::util::concurrent::BrokenBarrierException::~BrokenBarrierException () throw () [virtual]`

6.79.2 Member Function Documentation

6.79.2.1 `virtual BrokenBarrierException* decaf::util::concurrent::BrokenBarrierException::clone () const [inline, virtual]`

Clones this exception.

This is useful for cases where you need to preserve the type of the original exception as well as the message. All subclasses should override.

Returns

a new instance of an exception that is a clone of this one.

Reimplemented from `decaf::lang::Exception` (p. 1282).

The documentation for this class was generated from the following file:

- `src/main/decaf/util/concurrent/BrokenBarrierException.h`

6.80 activemq::commands::BrokerError Class Reference

This class represents an Exception sent from the Broker.

```
#include <src/main/activemq/commands/BrokerError.h>
```

Inheritance diagram for `activemq::commands::BrokerError`:

Data Structures

- struct `StackTraceElement`

Public Member Functions

- **BrokerError** ()
- virtual **~BrokerError** ()
- virtual unsigned char **getDataStructureType** () const
*Get the **DataStructure** (p. 1133) Type as defined in CommandTypes.h.*
- virtual **BrokerError** * **cloneDataStructure** () const
Clone this object and return a new instance that the caller now owns, this will be an exact copy of this one.
- virtual void **copyDataStructure** (const **DataStructure** *src)
Copy the contents of the passed object into this objects members, overwriting any existing data.
- virtual **decaf::lang::Pointer** < **commands::Command** > **visit** (activemq::state::CommandVisitor *visitor)
Allows a Visitor to visit this command and return a response to the command based on the command type being visited.
- virtual const std::string & **getMessage** () const
Gets the string holding the error message.
- virtual void **setMessage** (const std::string &message)
*Sets the string that contains the error **Message** (p. 1821).*
- virtual const std::string & **getExceptionClass** () const
Gets the string holding the Exception Class name.
- virtual void **setExceptionClass** (const std::string &exceptionClass)
Sets the string that contains the Exception Class name.
- virtual const **decaf::lang::Pointer** < **BrokerError** > & **getCause** () const
Gets the Broker Error that caused this exception.
- virtual void **setCause** (const **decaf::lang::Pointer**< **BrokerError** > &cause)
Sets the Broker Error that caused this exception.
- virtual const std::vector < **decaf::lang::Pointer** < **StackTraceElement** > > & **getStackTraceElements** () const
Gets the Stack Trace Elements for the Exception.
- virtual void **setStackTraceElements** (const std::vector< **decaf::lang::Pointer**< **StackTraceElement** > > &stackTraceElements)
Sets the Stack Trace Elements for this Exception.

6.80.1 Detailed Description

This class represents an Exception sent from the Broker.

The Broker sends java Throwables, so we must mimic its structure here.

6.80.2 Constructor & Destructor Documentation

6.80.2.1 **activemq::commands::BrokerError::BrokerError ()**

6.80.2.2 **virtual activemq::commands::BrokerError::~~BrokerError ()**
[virtual]

6.80.3 Member Function Documentation

6.80.3.1 **virtual BrokerError* activemq::commands::BrokerError::cloneDataStructure () const** [inline, virtual]

Clone this object and return a new instance that the caller now owns, this will be an exact copy of this one.

Returns

new copy of this object.

Implements **activemq::commands::DataStructure** (p. 1133).

References `copyDataStructure()`.

6.80.3.2 **virtual void activemq::commands::BrokerError::copyDataStructure (const DataStructure * src)** [virtual]

Copy the contents of the passed object into this objects members, overwriting any existing data.

Returns

src - Source Object

Reimplemented from **activemq::commands::BaseCommand** (p. 493).

Referenced by `cloneDataStructure()`.

6.80.3.3 **virtual const decaf::lang::Pointer<BrokerError>& activemq::commands::BrokerError::getCause () const** [inline, virtual]

Gets the Broker Error that caused this exception.

Returns

Broker Error Pointer

6.80.3.4 `virtual unsigned char activemq::commands::BrokerError::getDataStructureType () const [inline, virtual]`

Get the **DataSet** (p. 1133) Type as defined in CommandTypes.h.

Returns

The type of the data structure

Implements **activemq::commands::DataSet** (p. 1137).

6.80.3.5 `virtual const std::string& activemq::commands::BrokerError::getExceptionClass () const [inline, virtual]`

Gets the string holding the Exception Class name.

Returns

Exception Class name

6.80.3.6 `virtual const std::string& activemq::commands::BrokerError::getMessage () const [inline, virtual]`

Gets the string holding the error message.

Returns

String **Message** (p. 1821)

6.80.3.7 `virtual const std::vector< decaf::lang::Pointer<StackTraceElement> >& activemq::commands::BrokerError::getStackTraceElements () const [inline, virtual]`

Gets the Stack Trace Elements for the Exception.

Returns

Stack Trace Elements

6.80.3.8 `virtual void activemq::commands::BrokerError::setCause (const decaf::lang::Pointer< BrokerError > & cause) [inline, virtual]`

Sets the Broker Error that caused this exception.

Parameters

<i>cause</i>	- Broker Error
--------------	----------------

6.80.3.9 `virtual void activemq::commands::BrokerError::setExceptionClass (const std::string & exceptionClass) [inline, virtual]`

Sets the string that contains the Exception Class name.

Parameters

<i>exception- Class</i>	- String Exception Class name
-----------------------------	-------------------------------

6.80.3.10 `virtual void activemq::commands::BrokerError::setMessage (const std::string & message) [inline, virtual]`

Sets the string that contains the error **Message** (p. 1821).

Parameters

<i>message</i>	- String Error Message (p. 1821)
----------------	---

6.80.3.11 `virtual void activemq::commands::BrokerError::setStackTraceElements (const std::vector< decaf::lang::Pointer< StackTraceElement > > & stackTraceElements) [inline, virtual]`

Sets the Stack Trace Elements for this Exception.

Parameters

<i>stackTrace- Elements</i>	- Stack Trace Elements
---------------------------------	------------------------

6.80.3.12 `virtual decaf::lang::Pointer<commands::Command> activemq::commands::BrokerError::visit (activemq::state::CommandVisitor * visitor) [virtual]`

Allows a Visitor to visit this command and return a response to the command based on the command type being visited.

The command will call the proper processXXX method in the visitor.

Returns

a **Response** (p. 2298) to the visitor being called or NULL if no response.

Implements **activemq::commands::Command** (p. 871).

The documentation for this class was generated from the following file:

- src/main/activemq/commands/**BrokerError.h**

6.81 activemq::exceptions::BrokerException Class Reference

```
#include <src/main/activemq/exceptions/BrokerException.h>
```

Inheritance diagram for activemq::exceptions::BrokerException:

Public Member Functions

- **BrokerException** () throw ()
- **BrokerException** (const **exceptions::ActiveMQException** &ex) throw ()
- **BrokerException** (const **BrokerException** &ex) throw ()
- **BrokerException** (const char *file, const int lineNumber, const char *msg,...) throw ()
- **BrokerException** (const char *file, const int lineNumber, const **commands::BrokerError** *error) throw ()
- virtual **BrokerException** * **clone** () const
Clones this exception.
- virtual ~**BrokerException** () throw ()

6.81.1 Constructor & Destructor Documentation

6.81.1.1 **activemq::exceptions::BrokerException::BrokerException** () throw ()
[inline]

6.81.1.2 **activemq::exceptions::BrokerException::BrokerException** (const **exceptions::ActiveMQException** & ex) throw () [inline]

6.81.1.3 **activemq::exceptions::BrokerException::BrokerException** (const **BrokerException** & ex) throw () [inline]

6.81.1.4 **activemq::exceptions::BrokerException::BrokerException** (const char * *file*, const int *lineNumber*, const char * *msg*, ...) throw () [inline]

6.81.1.5 **activemq::exceptions::BrokerException::BrokerException** (const char *
file, const int *lineNumber*, const commands::BrokerError * *error*) throw ()
 [inline]

References decaf::lang::Exception::getMessage().

6.81.1.6 **virtual activemq::exceptions::BrokerException::~~BrokerException** ()
 throw () [inline, virtual]

6.81.2 Member Function Documentation

6.81.2.1 **virtual BrokerException* activemq::exceptions::BrokerException::clone** (
) const [inline, virtual]

Clones this exception.

This is useful for cases where you need to preserve the type of the original exception as well as the message. All subclasses should override.

Reimplemented from **activemq::exceptions::ActiveMQException** (p. 263).

The documentation for this class was generated from the following file:

- src/main/activemq/exceptions/**BrokerException.h**

6.82 activemq::commands::BrokerId Class Reference

```
#include <src/main/activemq/commands/BrokerId.h>
```

Inheritance diagram for activemq::commands::BrokerId:

Public Types

- typedef **decaf::lang::PointerComparator** < **BrokerId** > **COMPARATOR**

Public Member Functions

- **BrokerId** ()
- **BrokerId** (const **BrokerId** &other)
- virtual **~BrokerId** ()
- virtual unsigned char **getDataStructureType** () const
*Get the **DataStructure** (p. 1133) Type as defined in CommandTypes.h.*
- virtual **BrokerId** * **cloneDataStructure** () const
Clone this object and return a new instance that the caller now owns, this will be an exact copy of this one.

- virtual void **copyDataStructure** (const **DataStructure** *src)
Copy the contents of the passed object into this objects members, overwriting any existing data.
- virtual std::string **toString** () const
*Returns a string containing the information for this **DataStructure** (p. 1133) such as its type and value of its elements.*
- virtual bool **equals** (const **DataStructure** *value) const
*Compares the **DataStructure** (p. 1133) passed in to this one, and returns if they are equivalent.*
- virtual const std::string & **getValue** () const
- virtual std::string & **getValue** ()
- virtual void **setValue** (const std::string &value)
- virtual int **compareTo** (const **BrokerId** &value) const
- virtual bool **equals** (const **BrokerId** &value) const
- virtual bool **operator==** (const **BrokerId** &value) const
- virtual bool **operator<** (const **BrokerId** &value) const
- **BrokerId** & **operator=** (const **BrokerId** &other)

Static Public Attributes

- static const unsigned char **ID_BROKERID** = 124

Protected Attributes

- std::string **value**

6.82.1 Member Typedef Documentation

- 6.82.1.1 `typedef decaf::lang::PointerComparator<BrokerId>
 activemq::commands::BrokerId::COMPARATOR`

6.82.2 Constructor & Destructor Documentation

- 6.82.2.1 `activemq::commands::BrokerId::BrokerId ()`
- 6.82.2.2 `activemq::commands::BrokerId::BrokerId (const BrokerId & other)`
- 6.82.2.3 `virtual activemq::commands::BrokerId::~~BrokerId () [virtual]`

6.82.3 Member Function Documentation

- 6.82.3.1 `virtual BrokerId* activemq::commands::BrokerId::cloneDataStructure ()
 const [virtual]`

Clone this obbect and return a new instance that the caller now owns, this will be an exact copy of this one.

Returns

new copy of this object.

Implements **activemq::commands::DataStructure** (p. 1133).

6.82.3.2 `virtual int activemq::commands::BrokerId::compareTo (const BrokerId & value) const [virtual]`

6.82.3.3 `virtual void activemq::commands::BrokerId::copyDataStructure (const DataStructure * src) [virtual]`

Copy the contents of the passed object into this objects members, overwriting any existing data.

Returns

src - Source Object

Implements **activemq::commands::DataStructure** (p. 1134).

6.82.3.4 `virtual bool activemq::commands::BrokerId::equals (const DataStructure * value) const [virtual]`

Compares the **DataStructure** (p. 1133) passed in to this one, and returns if they are equivalent.

Equivalent here means that they are of the same type, and that each element of the objects are the same.

Returns

true if **DataStructure** (p. 1133)'s are Equal.

Implements **activemq::commands::DataStructure** (p. 1135).

6.82.3.5 `virtual bool activemq::commands::BrokerId::equals (const BrokerId & value) const [virtual]`

6.82.3.6 `virtual unsigned char activemq::commands::BrokerId::getDataStructureType () const [virtual]`

Get the **DataStructure** (p. 1133) Type as defined in CommandTypes.h.

Returns

The type of the data structure

Implements **activemq::commands::DataStructure** (p. 1137).

6.83 activemq::wireformat::openwire::marshal::generated::BrokerIdMarshaller Class Reference 567

- 6.82.3.7 `virtual const std::string& activemq::commands::BrokerId::getValue () const`
[virtual]
- 6.82.3.8 `virtual std::string& activemq::commands::BrokerId::getValue ()`
[virtual]
- 6.82.3.9 `virtual bool activemq::commands::BrokerId::operator< (const BrokerId & value) const` [virtual]
- 6.82.3.10 `BrokerId& activemq::commands::BrokerId::operator= (const BrokerId & other)`
- 6.82.3.11 `virtual bool activemq::commands::BrokerId::operator== (const BrokerId & value) const` [virtual]
- 6.82.3.12 `virtual void activemq::commands::BrokerId::setValue (const std::string & value)` [virtual]
- 6.82.3.13 `virtual std::string activemq::commands::BrokerId::toString () const`
[virtual]

Returns a string containing the information for this **DataSet** (p. 1133) such as its type and value of its elements.

Returns

formatted string useful for debugging.

Reimplemented from **activemq::commands::BaseDataSet** (p. 531).

6.82.4 Field Documentation

- 6.82.4.1 `const unsigned char activemq::commands::BrokerId::ID_BROKERID = 124`
[static]
- 6.82.4.2 `std::string activemq::commands::BrokerId::value` [protected]

The documentation for this class was generated from the following file:

- `src/main/activemq/commands/BrokerId.h`

6.83 activemq::wireformat::openwire::marshal::generated::BrokerIdMarshaller Class Reference

Marshaling code for Open Wire Format for **BrokerIdMarshaller** (p. 567).

```
#include <src/main/activemq/wireformat/openwire/marshal/generated/-  
BrokerIdMarshaller.h>
```

Inheritance diagram for activemq::wireformat::openwire::marshal::generated::BrokerIdMarshaller:

Public Member Functions

- **BrokerIdMarshaller** ()
- virtual **~BrokerIdMarshaller** ()
- virtual **commands::DataStructure * createObject** () const
Creates a new instance of the class that this class is a marshaling director for.
- virtual unsigned char **getDataStructureType** () const
Gets the DataStructureType that this class marshals/unmarshals.
- virtual void **tightUnmarshal** (**OpenWireFormat *wireFormat**, **commands::DataStructure *dataStructure**, **decaf::io::DataInputStream *dataIn**, **utils::BooleanStream *bs**)
Tight Un-marhsal to the given stream.
- virtual int **tightMarshal1** (**OpenWireFormat *wireFormat**, **commands::DataStructure *dataStructure**, **utils::BooleanStream *bs**)
Tight Marhsal to the given stream.
- virtual void **tightMarshal2** (**OpenWireFormat *wireFormat**, **commands::DataStructure *dataStructure**, **decaf::io::DataOutputStream *dataOut**, **utils::BooleanStream *bs**)
Tight Marhsal to the given stream.
- virtual void **looseUnmarshal** (**OpenWireFormat *wireFormat**, **commands::DataStructure *dataStructure**, **decaf::io::DataInputStream *dataIn**)
Loose Un-marhsal to the given stream.
- virtual void **looseMarshal** (**OpenWireFormat *wireFormat**, **commands::DataStructure *dataStructure**, **decaf::io::DataOutputStream *dataOut**)
Tight Marhsal to the given stream.

6.83.1 Detailed Description

Marshaling code for Open Wire Format for **BrokerIdMarshaller** (p. 567).

NOTE!: This file is auto generated - do not modify! if you need to make a change, please see the Java Classes in the activemq-openwire-generator module

6.83.2 Constructor & Destructor Documentation

- 6.83.2.1 **activemq::wireformat::openwire::marshal::generated::BrokerIdMarshaller::BrokerIdMarshaller** ()
[inline]

6.83.2.2 `virtual activemq::wireformat::openwire::marshal::generated-
::BrokerIdMarshaller::~~BrokerIdMarshaller () [inline,
virtual]`

6.83.3 Member Function Documentation

6.83.3.1 `virtual commands::DataStructure* activemq::wireformat::openwire-
::marshal::generated::BrokerIdMarshaller::createObject () const
[virtual]`

Creates a new instance of the class that this class is a marshaling director for.

Returns

newly allocated Command

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller`
(p. 1120).

6.83.3.2 `virtual unsigned char activemq::wireformat::openwire::marshal-
::generated::BrokerIdMarshaller::getDataStructureType () const
[virtual]`

Gets the DataStructureType that this class marshals/unmarshals.

Returns

byte Id of this classes DataStructureType

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller`
(p. 1122).

6.83.3.3 `virtual void activemq::wireformat::openwire::marshal::generated-
::BrokerIdMarshaller::looseMarshal (OpenWireFormat * format,
commands::DataStructure * command, decaf::io::DataOutputStream * ds
) [virtual]`

Tight Marshal to the given stream.

Parameters

<i>format</i>	- The OpenWireFormat properties
<i>command</i>	- the object to Marshal
<i>ds</i>	- DataOutputStream to marshal to

Exceptions

<i>IOException</i>	if an error occurs.
--------------------	---------------------

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1123).

6.83.3.4 **virtual void activemq::wireformat::openwire::marshal::generated::BrokerIdMarshaller::looseUnmarshal (OpenWireFormat * *format*, commands::DataStructure * *command*, decaf::io::DataInputStream * *dis*)**
[virtual]

Loose Un-marhsal to the given stream.

Parameters

<i>format</i>	- The OpenwireFormat properties
<i>command</i>	- the object to Un-Marshal
<i>dis</i>	- the DataInputStream to Un-Marshal from

Exceptions

<i>IOException</i>	if an error occurs.
--------------------	---------------------

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1125).

6.83.3.5 **virtual int activemq::wireformat::openwire::marshal::generated::BrokerIdMarshaller::tightMarshal1 (OpenWireFormat * *format*, commands::DataStructure * *command*, utils::BooleanStream * *bs*)**
[virtual]

Tight Marhsal to the given stream.

Parameters

<i>format</i>	The OpenwireFormat properties
<i>command</i>	The object to Marshal
<i>bs</i>	The boolean stream to marshal to.

Exceptions

<i>IOException</i>	if an error occurs.
--------------------	---------------------

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1127).

6.83.3.6 virtual void activemq::wireformat::openwire::marshal::generated::BrokerIdMarshaller::tightMarshal2 (OpenWireFormat * *format*, commands::DataStructure * *command*, decaf::io::DataOutputStream * *ds*, utils::BooleanStream * *bs*) [virtual]

Tight Marshal to the given stream.

Parameters

<i>format</i>	- The OpenWireFormat properties
<i>command</i>	- the object to Marshal
<i>ds</i>	- the DataOutputStream to Marshal to
<i>bs</i>	- boolean stream to marshal to.

Exceptions

<i>IOException</i>	if an error occurs.
--------------------	---------------------

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1129).

6.83.3.7 virtual void activemq::wireformat::openwire::marshal::generated::BrokerIdMarshaller::tightUnmarshal (OpenWireFormat * *format*, commands::DataStructure * *command*, decaf::io::DataInputStream * *dis*, utils::BooleanStream * *bs*) [virtual]

Tight Un-marshal to the given stream.

Parameters

<i>format</i>	- The OpenWireFormat properties
<i>command</i>	- the object to Un-Marshal
<i>dis</i>	- the DataInputStream to Un-Marshal from
<i>bs</i>	- boolean stream to unmarshal from.

Exceptions

<i>IOException</i>	if an error occurs.
--------------------	---------------------

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1131).

The documentation for this class was generated from the following file:

- src/main/activemq/wireformat/openwire/marshal/generated/**BrokerIdMarshaller.h**

6.84 activemq::commands::BrokerInfo Class Reference

```
#include <src/main/activemq/commands/BrokerInfo.h>
```

Inheritance diagram for activemq::commands::BrokerInfo:

Public Member Functions

- **BrokerInfo** ()
- virtual **~BrokerInfo** ()
- virtual unsigned char **getDataStructureType** () const
*Get the **DataStructure** (p. 1133) Type as defined in CommandTypes.h.*
- virtual **BrokerInfo** * **cloneDataStructure** () const
Clone this obbect and return a new instance that the caller now owns, this will be an exact copy of this one.
- virtual void **copyDataStructure** (const **DataStructure** *src)
Copy the contents of the passed object into this objects members, overwriting any existing data.
- virtual std::string **toString** () const
*Returns a string containing the information for this **DataStructure** (p. 1133) such as its type and value of its elements.*
- virtual bool **equals** (const **DataStructure** *value) const
*Compares the **DataStructure** (p. 1133) passed in to this one, and returns if they are equivalent.*
- virtual const **Pointer**< **BrokerId** > & **getBrokerId** () const
- virtual **Pointer**< **BrokerId** > & **getBrokerId** ()
- virtual void **setBrokerId** (const **Pointer**< **BrokerId** > &brokerId)
- virtual const std::string & **getBrokerURL** () const
- virtual std::string & **getBrokerURL** ()
- virtual void **setBrokerURL** (const std::string &brokerURL)
- virtual const std::vector < **decaf::lang::Pointer** < **BrokerInfo** > > & **getPeerBrokerInfos** () const
- virtual std::vector < **decaf::lang::Pointer** < **BrokerInfo** > > & **getPeerBrokerInfos** ()
- virtual void **setPeerBrokerInfos** (const std::vector< **decaf::lang::Pointer**< **BrokerInfo** > > &peerBrokerInfos)
- virtual const std::string & **getBrokerName** () const
- virtual std::string & **getBrokerName** ()
- virtual void **setBrokerName** (const std::string &brokerName)
- virtual bool **isSlaveBroker** () const
- virtual void **setSlaveBroker** (bool slaveBroker)
- virtual bool **isMasterBroker** () const
- virtual void **setMasterBroker** (bool masterBroker)
- virtual bool **isFaultTolerantConfiguration** () const

- virtual void **setFaultTolerantConfiguration** (bool **faultTolerantConfiguration**)
- virtual bool **isDuplexConnection** () const
- virtual void **setDuplexConnection** (bool **duplexConnection**)
- virtual bool **isNetworkConnection** () const
- virtual void **setNetworkConnection** (bool **networkConnection**)
- virtual long long **getConnectionId** () const
- virtual void **setConnectionId** (long long **connectionId**)
- virtual const std::string & **getBrokerUploadUrl** () const
- virtual std::string & **getBrokerUploadUrl** ()
- virtual void **setBrokerUploadUrl** (const std::string &**brokerUploadUrl**)
- virtual const std::string & **getNetworkProperties** () const
- virtual std::string & **getNetworkProperties** ()
- virtual void **setNetworkProperties** (const std::string &**networkProperties**)
- virtual bool **isBrokerInfo** () const
- virtual **Pointer**< **Command** > **visit** (activemq::state::CommandVisitor *visitor)

Allows a Visitor to visit this command and return a response to the command based on the command type being visited.

Static Public Attributes

- static const unsigned char **ID_BROKERINFO** = 2

Protected Attributes

- **Pointer**< **BrokerId** > **brokerId**
- std::string **brokerURL**
- std::vector < **decaf::lang::Pointer** < **BrokerInfo** > > **peerBrokerInfos**
- std::string **brokerName**
- bool **slaveBroker**
- bool **masterBroker**
- bool **faultTolerantConfiguration**
- bool **duplexConnection**
- bool **networkConnection**
- long long **connectionId**
- std::string **brokerUploadUrl**
- std::string **networkProperties**

6.84.1 Constructor & Destructor Documentation

6.84.1.1 **activemq::commands::BrokerInfo::BrokerInfo** ()

6.84.1.2 **virtual activemq::commands::BrokerInfo::~~BrokerInfo** () [virtual]

6.84.2 Member Function Documentation

6.84.2.1 `virtual BrokerInfo* activemq::commands::BrokerInfo::cloneDataStructure () const [virtual]`

Clone this object and return a new instance that the caller now owns, this will be an exact copy of this one.

Returns

new copy of this object.

Implements `activemq::commands::DataStructure` (p. 1133).

6.84.2.2 `virtual void activemq::commands::BrokerInfo::copyDataStructure (const DataStructure * src) [virtual]`

Copy the contents of the passed object into this objects members, overwriting any existing data.

Returns

src - Source Object

Reimplemented from `activemq::commands::BaseCommand` (p. 493).

6.84.2.3 `virtual bool activemq::commands::BrokerInfo::equals (const DataStructure * value) const [virtual]`

Compares the `DataStructure` (p. 1133) passed in to this one, and returns if they are equivalent.

Equivalent here means that they are of the same type, and that each element of the objects are the same.

Returns

true if `DataStructure` (p. 1133)'s are Equal.

Reimplemented from `activemq::commands::BaseCommand` (p. 494).

6.84.2.4 `virtual const Pointer<BrokerId>& activemq::commands::BrokerInfo::getBrokerId () const [virtual]`

6.84.2.5 `virtual Pointer<BrokerId>& activemq::commands::BrokerInfo::getBrokerId () [virtual]`

6.84.2.6 virtual const std::string& activemq::commands::BrokerInfo::getBrokerName () const [virtual]

6.84.2.7 virtual std::string& activemq::commands::BrokerInfo::getBrokerName () [virtual]

6.84.2.8 virtual const std::string& activemq::commands::BrokerInfo::getBrokerUploadUrl () const [virtual]

6.84.2.9 virtual std::string& activemq::commands::BrokerInfo::getBrokerUploadUrl () [virtual]

6.84.2.10 virtual const std::string& activemq::commands::BrokerInfo::getBrokerURL () const [virtual]

6.84.2.11 virtual std::string& activemq::commands::BrokerInfo::getBrokerURL () [virtual]

6.84.2.12 virtual long long activemq::commands::BrokerInfo::getConnectionId () const [virtual]

6.84.2.13 virtual unsigned char activemq::commands::BrokerInfo::getDataStructureType () const [virtual]

Get the **DataSetructure** (p. 1133) Type as defined in CommandTypes.h.

Returns

The type of the data structure

Implements **activemq::commands::DataSetructure** (p. 1137).

6.84.2.14 virtual const std::string& activemq::commands::BrokerInfo::getNetworkProperties () const [virtual]

6.84.2.15 virtual std::string& activemq::commands::BrokerInfo::getNetworkProperties () [virtual]

6.84.2.16 virtual const std::vector< decaf::lang::Pointer<BrokerInfo> >& activemq::commands::BrokerInfo::getPeerBrokerInfos () const [virtual]

6.84.2.17 virtual std::vector< decaf::lang::Pointer<BrokerInfo> >& activemq::commands::BrokerInfo::getPeerBrokerInfos () [virtual]

6.84.2.18 virtual bool activemq::commands::BrokerInfo::isBrokerInfo () const [inline, virtual]

Returns

an answer of true to the **isBrokerInfo()** (p. 575) query.

Reimplemented from **activemq::commands::BaseCommand** (p. 495).

- 6.84.2.19 `virtual bool activemq::commands::BrokerInfo::isDuplexConnection ()
const [virtual]`
- 6.84.2.20 `virtual bool activemq::commands::BrokerInfo::isFaultTolerant-
Configuration ()const [virtual]`
- 6.84.2.21 `virtual bool activemq::commands::BrokerInfo::isMasterBroker ()const
[virtual]`
- 6.84.2.22 `virtual bool activemq::commands::BrokerInfo::isNetworkConnection ()
const [virtual]`
- 6.84.2.23 `virtual bool activemq::commands::BrokerInfo::isSlaveBroker ()const
[virtual]`
- 6.84.2.24 `virtual void activemq::commands::BrokerInfo::setBrokerId (const
Pointer< BrokerId > & brokerId) [virtual]`
- 6.84.2.25 `virtual void activemq::commands::BrokerInfo::setBrokerName (const
std::string & brokerName) [virtual]`
- 6.84.2.26 `virtual void activemq::commands::BrokerInfo::setBrokerUploadUrl (const
std::string & brokerUploadUrl) [virtual]`
- 6.84.2.27 `virtual void activemq::commands::BrokerInfo::setBrokerURL (const
std::string & brokerURL) [virtual]`
- 6.84.2.28 `virtual void activemq::commands::BrokerInfo::setConnectionId (long long
connectionId) [virtual]`
- 6.84.2.29 `virtual void activemq::commands::BrokerInfo::setDuplexConnection (
bool duplexConnection) [virtual]`
- 6.84.2.30 `virtual void activemq::commands::BrokerInfo::setFault-
TolerantConfiguration (bool faultTolerantConfiguration)
[virtual]`
- 6.84.2.31 `virtual void activemq::commands::BrokerInfo::setMasterBroker (bool
masterBroker) [virtual]`
- 6.84.2.32 `virtual void activemq::commands::BrokerInfo::setNetworkConnection (
bool networkConnection) [virtual]`

- 6.84.2.33 `virtual void activemq::commands::BrokerInfo::setNetworkProperties (const std::string & networkProperties)` [virtual]
- 6.84.2.34 `virtual void activemq::commands::BrokerInfo::setPeerBrokerInfos (const std::vector< decaf::lang::Pointer< BrokerInfo > > & peerBrokerInfos)` [virtual]
- 6.84.2.35 `virtual void activemq::commands::BrokerInfo::setSlaveBroker (bool slaveBroker)` [virtual]
- 6.84.2.36 `virtual std::string activemq::commands::BrokerInfo::toString () const` [virtual]

Returns a string containing the information for this **DataSet** (p. 1133) such as its type and value of its elements.

Returns

formatted string useful for debugging.

Reimplemented from **activemq::commands::BaseCommand** (p. 498).

- 6.84.2.37 `virtual Pointer<Command> activemq::commands::BrokerInfo::visit (activemq::state::CommandVisitor * visitor)` [virtual]

Allows a Visitor to visit this command and return a response to the command based on the command type being visited.

The command will call the proper processXXX method in the visitor.

Returns

a **Response** (p. 2298) to the visitor being called or NULL if no response.

Implements **activemq::commands::Command** (p. 871).

6.84.3 Field Documentation

- 6.84.3.1 `Pointer<BrokerId> activemq::commands::BrokerInfo::brokerId` [protected]
- 6.84.3.2 `std::string activemq::commands::BrokerInfo::brokerName` [protected]
- 6.84.3.3 `std::string activemq::commands::BrokerInfo::brokerUploadUrl` [protected]
- 6.84.3.4 `std::string activemq::commands::BrokerInfo::brokerURL` [protected]

- 6.84.3.5 `long long activemq::commands::BrokerInfo::connectionId`
[protected]
- 6.84.3.6 `bool activemq::commands::BrokerInfo::duplexConnection`
[protected]
- 6.84.3.7 `bool activemq::commands::BrokerInfo::faultTolerantConfiguration`
[protected]
- 6.84.3.8 `const unsigned char activemq::commands::BrokerInfo::ID_BROKERINFO = 2` [static]
- 6.84.3.9 `bool activemq::commands::BrokerInfo::masterBroker` [protected]
- 6.84.3.10 `bool activemq::commands::BrokerInfo::networkConnection`
[protected]
- 6.84.3.11 `std::string activemq::commands::BrokerInfo::networkProperties`
[protected]
- 6.84.3.12 `std::vector< decaf::lang::Pointer<BrokerInfo> >`
`activemq::commands::BrokerInfo::peerBrokerInfos` [protected]
- 6.84.3.13 `bool activemq::commands::BrokerInfo::slaveBroker` [protected]

The documentation for this class was generated from the following file:

- `src/main/activemq/commands/BrokerInfo.h`

6.85 `activemq::wireformat::openwire::marshal::generated::BrokerInfoMarshaller` Class Reference

Marshaling code for Open Wire Format for **BrokerInfoMarshaller** (p. 578).

```
#include <src/main/activemq/wireformat/openwire/marshal/generated/-
BrokerInfoMarshaller.h>
```

Inheritance diagram for `activemq::wireformat::openwire::marshal::generated::BrokerInfoMarshaller`:

Public Member Functions

- **BrokerInfoMarshaller** ()
- virtual **~BrokerInfoMarshaller** ()
- virtual **commands::DataStructure * createObject** () const

Creates a new instance of the class that this class is a marshaling director for.

- virtual unsigned char **getDataStructureType** () const
Gets the DataStructureType that this class marshals/unmarshals.
- virtual void **tightUnmarshal** (OpenWireFormat *wireFormat, commands::DataStructure *dataStructure, decaf::io::DataInputStream *dataIn, utils::BooleanStream *bs)
Tight Un-marhsal to the given stream.
- virtual int **tightMarshal1** (OpenWireFormat *wireFormat, commands::DataStructure *dataStructure, utils::BooleanStream *bs)
Tight Marhsal to the given stream.
- virtual void **tightMarshal2** (OpenWireFormat *wireFormat, commands::DataStructure *dataStructure, decaf::io::DataOutputStream *dataOut, utils::BooleanStream *bs)
Tight Marhsal to the given stream.
- virtual void **looseUnmarshal** (OpenWireFormat *wireFormat, commands::DataStructure *dataStructure, decaf::io::DataInputStream *dataIn)
Loose Un-marhsal to the given stream.
- virtual void **looseMarshal** (OpenWireFormat *wireFormat, commands::DataStructure *dataStructure, decaf::io::DataOutputStream *dataOut)
Tight Marhsal to the given stream.

6.85.1 Detailed Description

Marshaling code for Open Wire Format for **BrokerInfoMarshaller** (p. 578).

NOTE!: This file is auto generated - do not modify! if you need to make a change, please see the Java Classes in the activemq-openwire-generator module

6.85.2 Constructor & Destructor Documentation

- 6.85.2.1 **activemq::wireformat::openwire::marshal::generated::BrokerInfoMarshaller::BrokerInfoMarshaller** ()
[inline]
- 6.85.2.2 **virtual activemq::wireformat::openwire::marshal::generated::BrokerInfoMarshaller::~~BrokerInfoMarshaller** () [inline, virtual]

6.85.3 Member Function Documentation

- 6.85.3.1 **virtual commands::DataStructure* activemq::wireformat::openwire::marshal::generated::BrokerInfoMarshaller::createObject** () const
[virtual]

Creates a new instance of the class that this class is a marshaling director for.

Returns

newly allocated Command

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1120).

6.85.3.2 `virtual unsigned char activemq::wireformat::openwire::marshal::generated::BrokerInfoMarshaller::getDataStructureType () const`
[virtual]

Gets the DataStructureType that this class marshals/unmarshals.

Returns

byte Id of this classes DataStructureType

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1122).

6.85.3.3 `virtual void activemq::wireformat::openwire::marshal::generated::BrokerInfoMarshaller::looseMarshal (OpenWireFormat * format, commands::DataStructure * command, decaf::io::DataOutputStream * ds)` [virtual]

Tight Marshal to the given stream.

Parameters

<i>format</i>	- The OpenWireFormat properties
<i>command</i>	- the object to Marshal
<i>ds</i>	- DataOutputStream to marshal to

Exceptions

<i>IOException</i>	if an error occurs.
--------------------	---------------------

Reimplemented from **activemq::wireformat::openwire::marshal::generated::Base-CommandMarshaller** (p. 500).

6.85.3.4 `virtual void activemq::wireformat::openwire::marshal::generated::BrokerInfoMarshaller::looseUnmarshal (OpenWireFormat * format, commands::DataStructure * command, decaf::io::DataInputStream * dis)`
[virtual]

Loose Un-marshal to the given stream.

Parameters

<i>format</i>	- The OpenWireFormat properties
<i>command</i>	- the object to Un-Marshal
<i>dis</i>	- the DataInputStream to Un-Marshal from

Exceptions

<i>IOException</i>	if an error occurs.
--------------------	---------------------

Reimplemented from `activemq::wireformat::openwire::marshal::generated::Base-CommandMarshaller` (p. 501).

6.85.3.5 `virtual int activemq::wireformat::openwire::marshal::generated::-`
`BrokerInfoMarshaller::tightMarshal1 (OpenWireFormat * format,`
`commands::DataStructure * command, utils::BooleanStream * bs)`
`[virtual]`

Tight Marshal to the given stream.

Parameters

<i>format</i>	The OpenWireFormat properties
<i>command</i>	The object to Marshal
<i>bs</i>	The boolean stream to marshal to.

Exceptions

<i>IOException</i>	if an error occurs.
--------------------	---------------------

Reimplemented from `activemq::wireformat::openwire::marshal::generated::Base-CommandMarshaller` (p. 503).

6.85.3.6 `virtual void activemq::wireformat::openwire::marshal::generated::-`
`BrokerInfoMarshaller::tightMarshal2 (OpenWireFormat * format,`
`commands::DataStructure * command, decaf::io::DataOutputStream * ds,`
`utils::BooleanStream * bs) [virtual]`

Tight Marshal to the given stream.

Parameters

<i>format</i>	- The OpenWireFormat properties
<i>command</i>	- the object to Marshal
<i>ds</i>	- the DataOutputStream to Marshal to
<i>bs</i>	- boolean stream to marshal to.

Exceptions

<i>IOException</i>	if an error occurs.
--------------------	---------------------

Reimplemented from **activemq::wireformat::openwire::marshal::generated::Base-CommandMarshaller** (p. 504).

6.85.3.7 virtual void **activemq::wireformat::openwire::marshal::generated::BrokerInfoMarshaller::tightUnmarshal** (**OpenWireFormat** * *format*, **commands::DataStructure** * *command*, **decaf::io::DataInputStream** * *dis*, **utils::BooleanStream** * *bs*) [virtual]

Tight Un-marhsal to the given stream.

Parameters

<i>format</i>	- The OpenWireFormat properties
<i>command</i>	- the object to Un-Marshal
<i>dis</i>	- the DataInputStream to Un-Marshal from
<i>bs</i>	- boolean stream to unmarshal from.

Exceptions

<i>IOException</i>	if an error occurs.
--------------------	---------------------

Reimplemented from **activemq::wireformat::openwire::marshal::generated::Base-CommandMarshaller** (p. 505).

The documentation for this class was generated from the following file:

- src/main/activemq/wireformat/openwire/marshal/generated/**BrokerInfoMarshaller.h**

6.86 decaf::nio::Buffer Class Reference

A container for data of a specific primitive type.

```
#include <src/main/decaf/nio/Buffer.h>
```

Inheritance diagram for decaf::nio::Buffer:

Public Member Functions

- **Buffer** (int capacity)
- **Buffer** (const **Buffer** &other)
- virtual ~**Buffer** ()

- virtual int **capacity** () const
- virtual int **position** () const
- virtual **Buffer** & **position** (int newPosition)
Sets this buffer's position.
- virtual int **limit** () const
- virtual **Buffer** & **limit** (int newLimit)
Sets this buffer's limit.
- virtual **Buffer** & **mark** ()
Sets this buffer's mark at its position.
- virtual **Buffer** & **reset** ()
Resets this buffer's position to the previously-marked position.
- virtual **Buffer** & **clear** ()
Clears this buffer.
- virtual **Buffer** & **flip** ()
Flips this buffer.
- virtual **Buffer** & **rewind** ()
Rewinds this buffer.
- virtual int **remaining** () const
Returns the number of elements between the current position and the limit.
- virtual bool **hasRemaining** () const
Tells whether there are any elements between the current position and the limit.
- virtual bool **isReadOnly** () const =0
Tells whether or not this buffer is read-only.

Protected Attributes

- int **_position**
- int **_capacity**
- int **_limit**
- int **_mark**
- bool **_markSet**

6.86.1 Detailed Description

A container for data of a specific primitive type.

A buffer is a linear, finite sequence of elements of a specific primitive type. Aside from its content, the essential properties of a buffer are its capacity, limit, and position:

A buffer's capacity is the number of elements it contains. The capacity of a buffer is never negative and never changes.

A buffer's limit is the index of the first element that should not be read or written. A buffer's limit is never negative and is never greater than its capacity.

A buffer's position is the index of the next element to be read or written. A buffer's position is never negative and is never greater than its limit.

There is one subclass of this class for each non-boolean primitive type.

Transferring data: Each subclass of this class defines two categories of get and put operations: * Relative operations read or write one or more elements starting at the current position and then increment the position by the number of elements transferred. If the requested transfer exceeds the limit then a relative get operation throws a **BufferUnderflowException** (p. 611) and a relative put operation throws a **BufferOverflowException** (p. 609); in either case, no data is transferred. * Absolute operations take an explicit element index and do not affect the position. Absolute get and put operations throw an **IndexOutOfBoundsException** if the index argument exceeds the limit.

Data may also, of course, be transferred in to or out of a buffer by the I/O operations of an appropriate channel, which are always relative to the current position.

Marking and resetting:

A buffer's mark is the index to which its position will be reset when the reset method is invoked. The mark is not always defined, but when it is defined it is never negative and is never greater than the position. If the mark is defined then it is discarded when the position or the limit is adjusted to a value smaller than the mark. If the mark is not defined then invoking the reset method causes an **InvalidMarkException** (p. 1539) to be thrown.

Invariants:

The following invariant holds for the mark, position, limit, and capacity values: $0 \leq \text{mark} \leq \text{position} \leq \text{limit} \leq \text{capacity}$

A newly-created buffer always has a position of zero and a mark that is undefined. The initial limit may be zero, or it may be some other value that depends upon the type of the buffer and the manner in which it is constructed. The initial content of a buffer is, in general, undefined.

Clearing, flipping, and rewinding:

In addition to methods for accessing the position, limit, and capacity values and for marking and resetting, this class also defines the following operations upon buffers:

clear() (p. 585) makes a buffer ready for a new sequence of channel-read or relative put operations: It sets the limit to the capacity and the position to zero.

flip() (p. 585) makes a buffer ready for a new sequence of channel-write or relative get operations: It sets the limit to the current position and then sets the position to zero.

rewind() (p. 588) makes a buffer ready for re-reading the data that it already contains: It leaves the limit unchanged and sets the position to zero.

Read-only buffers:

Every buffer is readable, but not every buffer is writable. The mutation methods of each buffer class are specified as optional operations that will throw a **ReadOnlyBufferException** (p. 2244) when invoked upon a read-only buffer. A read-only buffer does not allow its content to be changed, but its mark, position, and limit values are mutable. Whether or not a buffer is read-only may be determined by invoking its `isReadOnly` method.

Thread safety:

Buffers are not safe for use by multiple concurrent threads. If a buffer is to be used by more than one thread then access to the buffer should be controlled by appropriate synchronization.

Invocation chaining:

Methods in this class that do not otherwise have a value to return are specified to return the buffer upon which they are invoked. This allows method invocations to be chained; for example, the sequence of statements

```
b.flip(); b.position(23); b.limit(42);
```

can be replaced by the single, more compact statement `b.flip().position(23).limit(42);`

6.86.2 Constructor & Destructor Documentation

6.86.2.1 `decaf::nio::Buffer::Buffer (int capacity)`

6.86.2.2 `decaf::nio::Buffer::Buffer (const Buffer & other)`

6.86.2.3 `virtual decaf::nio::Buffer::~~Buffer () [inline, virtual]`

6.86.3 Member Function Documentation

6.86.3.1 `virtual int decaf::nio::Buffer::capacity () const [inline, virtual]`

Returns

this buffer's capacity.

6.86.3.2 `virtual Buffer& decaf::nio::Buffer::clear () [virtual]`

Clears this buffer.

The position is set to zero, the limit is set to the capacity, and the mark is discarded.

Invoke this method before using a sequence of channel-read or put operations to fill this buffer. For example:

```
buf.clear(); // Prepare buffer for reading in.read(buf); // Read data
```

This method does not actually erase the data in the buffer, but it is named as if it did because it will most often be used in situations in which that might as well be the case.

Returns

a reference to this buffer.

6.86.3.3 `virtual Buffer& decaf::nio::Buffer::flip () [virtual]`

Flips this buffer.

The limit is set to the current position and then the position is set to zero. If the mark is defined then it is discarded.

After a sequence of channel-read or put operations, invoke this method to prepare for a sequence of channel-write or relative get operations. For example:

```
buf.put(magic); // Prepend header in.read(buf); // Read data into rest of buffer buf.flip();
// Flip buffer out.write(buf); // Write header + data to channel
```

This method is often used in conjunction with the compact method when transferring data from one place to another.

Returns

a reference to this buffer.

6.86.3.4 `virtual bool decaf::nio::Buffer::hasRemaining () const [inline, virtual]`

Tells whether there are any elements between the current position and the limit.

Returns

true if, and only if, there is at least one element remaining in this buffer.

6.86.3.5 `virtual bool decaf::nio::Buffer::isReadOnly () const [pure virtual]`

Tells whether or not this buffer is read-only.

Returns

true if, and only if, this buffer is read-only.

Implemented in **decaf::nio::ByteBuffer** (p. 707), **decaf::internal::nio::ByteBuffer** (p. 671), **decaf::internal::nio::CharArrayBuffer** (p. 782), **decaf::internal::nio::DoubleArrayBuffer** (p. 1257), **decaf::internal::nio::FloatArrayBuffer** (p. 1366), **decaf::internal::nio::IntArrayBuffer** (p. 1486), **decaf::internal::nio::LongArrayBuffer** (p. 1750), and **decaf::internal::nio::ShortArrayBuffer** (p. 2417).

6.86.3.6 `virtual int decaf::nio::Buffer::limit () const [inline, virtual]`

Returns

this buffers Limit

6.86.3.7 virtual Buffer& decaf::nio::Buffer::limit (int *newLimit*) [virtual]

Sets this buffer's limit.

If the position is larger than the new limit then it is set to the new limit. If the mark is defined and larger than the new limit then it is discarded.

Parameters

<i>newLimit</i>	The new limit value; must be no larger than this buffer's capacity.
-----------------	---

Returns

A reference to This buffer

Exceptions

<i>IllegalArgument-Exception</i>	if preconditions on the new pos don't hold.
----------------------------------	---

6.86.3.8 virtual Buffer& decaf::nio::Buffer::mark () [virtual]

Sets this buffer's mark at its position.

Returns

a reference to this buffer.

6.86.3.9 virtual int decaf::nio::Buffer::position () const [inline, virtual]**Returns**

the current position in the buffer

6.86.3.10 virtual Buffer& decaf::nio::Buffer::position (int *newPosition*) [virtual]

Sets this buffer's position.

If the mark is defined and larger than the new position then it is discarded.

Parameters

<i>newPosition</i>	The new postion in the buffer to set.
--------------------	---------------------------------------

Returns

a reference to This buffer.

Exceptions

<i>IllegalArgument-Exception</i>	if preconditions on the new pos don't hold.
----------------------------------	---

6.86.3.11 `virtual int decaf::nio::Buffer::remaining () const [inline, virtual]`

Returns the number of elements between the current position and the limit.

Returns

The number of elements remaining in this buffer

6.86.3.12 `virtual Buffer& decaf::nio::Buffer::reset () [virtual]`

Resets this buffer's position to the previously-marked position.

Returns

a reference to this buffer.

Exceptions

<i>InvalidMark-Exception</i> (p. 1539)	- If the mark has not been set
---	--------------------------------

6.86.3.13 `virtual Buffer& decaf::nio::Buffer::rewind () [virtual]`

Rewinds this buffer.

The position is set to zero and the mark is discarded.

Invoke this method before a sequence of channel-write or get operations, assuming that the limit has already been set appropriately. For example:

```
out.write(buf); // Write remaining data buf.rewind(); // Rewind buffer buf.get(array); // Copy data into array
```

Returns

a reference to this buffer.

6.86.4 Field Documentation

6.86.4.1 int decaf::nio::Buffer::_capacity [protected]

6.86.4.2 int decaf::nio::Buffer::_limit [protected]

6.86.4.3 int decaf::nio::Buffer::_mark [protected]

6.86.4.4 bool decaf::nio::Buffer::_markSet [protected]

6.86.4.5 int decaf::nio::Buffer::_position [mutable, protected]

The documentation for this class was generated from the following file:

- src/main/decaf/nio/**Buffer.h**

6.87 decaf::io::BufferedInputStream Class Reference

A wrapper around another input stream that performs a buffered read, where it reads more data than it needs in order to reduce the number of io operations on the input stream.

```
#include <src/main/decaf/io/BufferedInputStream.h>
```

Inheritance diagram for decaf::io::BufferedInputStream:

Public Member Functions

- **BufferedInputStream** (InputStream *stream, bool own=false)

Constructor.

- **BufferedInputStream** (InputStream *stream, int bufferSize, bool own=false)

Constructor.

- virtual ~**BufferedInputStream** ()

- virtual int **available** () const

Indicates the number of bytes available.

The default implementation of this methods returns 0. Classes that override this method may return the total number of bytes that are currently available for reading and others may simply return a value of one indicating that there is some data available. The caller should view the result of this method as an absolute.

The default implementation of this method returns zero.

Returns

the number of bytes available on this input stream.

Exceptions

IOException (p. 1545)	<i>if an I/O error occurs.</i>
------------------------------	--------------------------------

- virtual void **close** ()

*Closes the **InputStream** (p. 1464) freeing any resources that might have been acquired during the lifetime of this stream.*

The default implementation of this method does nothing.

Exceptions

IOException (p. 1545)	<i>if an I/O error occurs while closing the InputStream (p. 1464).</i>
------------------------------	---

- virtual long long **skip** (long long num)

*Skips over and discards *n* bytes of data from this input stream.*

*The skip method may, for a variety of reasons, end up skipping over some smaller number of bytes, possibly 0. This may result from any of a number of conditions; reaching end of file before *n* bytes have been skipped is only one possibility. The actual number of bytes skipped is returned.*

*The skip method of **InputStream** (p. 1464) creates a byte array and then repeatedly reads into it until *num* bytes have been read or the end of the stream has been reached. Subclasses are encouraged to provide a more efficient implementation of this method.*

Parameters

num	<i>The number of bytes to skip.</i>
-----	-------------------------------------

Returns

total bytes skipped

Exceptions

IOException (p. 1545)	<i>if an I/O error occurs.</i>
------------------------------	--------------------------------

UnsupportedOperation-Exception	<i>if the concrete stream class does not support skipping bytes.</i>
--------------------------------	--

- virtual void **mark** (int readLimit)

Marks the current position in the stream. A subsequent call to the reset method repositions this stream at the last marked position so that subsequent reads re-read the same bytes.

If a stream instance reports that marks are supported then the stream will ensure that the same bytes can be read again after the reset method is called so long the readLimit is not reached.

Calling mark on a closed stream instance should have no effect.

The default implementation of this method does nothing.

Parameters

readLimit	<i>The max bytes read before marked position is invalid.</i>
-----------	--

- virtual void **reset** ()

Repositions this stream to the position at the time the mark method was last called on this input stream.

*If the method markSupported returns true, then: * If the method mark has not been called since the stream was created, or the number of bytes read from the stream since mark was last called is larger than the argument to mark at that last call, then an **IOException** (p. 1545) might be thrown. * If such an **IOException** (p. 1545) is not thrown, then the stream is reset to a state such that all the bytes read since the most recent call to mark (or since the start of the file, if mark has not been called) will be resupplied to subsequent callers of the read method, followed by any bytes that otherwise would have been the next input data as of the time of the call to reset.*

*If the method markSupported returns false, then: * The call to reset may throw an **IOException** (p. 1545). * If an **IOException** (p. 1545) is not thrown, then the stream*

is reset to a fixed state that depends on the particular type of the input stream and how it was created. The bytes that will be supplied to subsequent callers of the read method depend on the particular type of the input stream.

The default implementation of this method throws an **IOException** (p. 1545).

Exceptions

IOException (p. 1545)	if an I/O error occurs.
------------------------------	-------------------------

- virtual bool **markSupported** () const

Determines if this input stream supports the mark and reset methods.

Whether or not mark and reset are supported is an invariant property of a particular input stream instance.

The default implementation of this method returns false.

Returns

true if this stream instance supports marks

Protected Member Functions

- virtual int **doReadByte** ()
- virtual int **doReadArrayBounded** (unsigned char *buffer, int size, int offset, int length)

6.87.1 Detailed Description

A wrapper around another input stream that performs a buffered read, where it reads more data than it needs in order to reduce the number of io operations on the input stream.

6.87.2 Constructor & Destructor Documentation

6.87.2.1 decaf::io::BufferedInputStream::BufferedInputStream (InputStream * stream, bool own = false)

Constructor.

Parameters

<i>stream</i>	The target input stream to buffer.
<i>own</i>	Indicates if we own the stream object, defaults to false.

6.87.2.2 decaf::io::BufferedInputStream::BufferedInputStream (InputStream * stream, int bufferSize, bool own = false)

Constructor.

Parameters

<i>stream</i>	The target input stream to buffer.
<i>bufferSize</i>	The size in bytes to allocate for the internal buffer.
<i>own</i>	Indicates if we own the stream object, defaults to false.

Exceptions

<i>IllegalArgumentException</i>	if the size is zero or negative.
---------------------------------	----------------------------------

6.87.2.3 `virtual decaf::io::BufferedInputStream::~~BufferedInputStream ()`
[virtual]

6.87.3 Member Function Documentation

6.87.3.1 `virtual int decaf::io::BufferedInputStream::available () const`
[virtual]

Indicates the number of bytes available.

The default implementation of this methods returns 0. Classes that override this method may return the total number of bytes that are currently available for reading and others may simply return a value of one indicating that there is some data available. The caller should view the result of this method as an absolute.

The default implementation of this method returns zero.

Returns

the number of bytes available on this input stream.

Exceptions

IOException (p. 1545)	if an I/O error occurs.
---------------------------------	-------------------------

Reimplemented from **decaf::io::FilterInputStream** (p. 1337).

6.87.3.2 `virtual void decaf::io::BufferedInputStream::close ()` [virtual]

Closes the **InputStream** (p. 1464) freeing any resources that might have been acquired during the lifetime of this stream.

The default implementation of this method does nothing.

Exceptions

IOException (p. 1545)	if an I/O error occurs while closing the InputStream (p. 1464).
---------------------------------	--

Reimplemented from `decaf::io::FilterInputStream` (p. 1337).

6.87.3.3 `virtual int decaf::io::BufferedInputStream::doReadArrayBounded (unsigned char * buffer, int size, int offset, int length)` `[protected, virtual]`

Reimplemented from `decaf::io::FilterInputStream` (p. 1338).

6.87.3.4 `virtual int decaf::io::BufferedInputStream::doReadByte ()` `[protected, virtual]`

Reimplemented from `decaf::io::FilterInputStream` (p. 1338).

6.87.3.5 `virtual void decaf::io::BufferedInputStream::mark (int readLimit)` `[virtual]`

Marks the current position in the stream. A subsequent call to the `reset` method repositions this stream at the last marked position so that subsequent reads re-read the same bytes.

If a stream instance reports that marks are supported then the stream will ensure that the same bytes can be read again after the `reset` method is called so long the `readLimit` is not reached.

Calling `mark` on a closed stream instance should have no effect.

The default implementation of this method does nothing.

Parameters

<i>readLimit</i>	The max bytes read before marked position is invalid.
------------------	---

Reimplemented from `decaf::io::FilterInputStream` (p. 1338).

6.87.3.6 `virtual bool decaf::io::BufferedInputStream::markSupported () const` `[inline, virtual]`

Determines if this input stream supports the `mark` and `reset` methods.

Whether or not `mark` and `reset` are supported is an invariant property of a particular input stream instance.

The default implementation of this method returns `false`.

Returns

`true` if this stream instance supports marks

Reimplemented from `decaf::io::FilterInputStream` (p. 1339).

6.87.3.7 virtual void decaf::io::BufferedInputStream::reset () [virtual]

Repositions this stream to the position at the time the mark method was last called on this input stream.

If the method markSupported returns true, then: * If the method mark has not been called since the stream was created, or the number of bytes read from the stream since mark was last called is larger than the argument to mark at that last call, then an **IOException** (p. 1545) might be thrown. * If such an **IOException** (p. 1545) is not thrown, then the stream is reset to a state such that all the bytes read since the most recent call to mark (or since the start of the file, if mark has not been called) will be resupplied to subsequent callers of the read method, followed by any bytes that otherwise would have been the next input data as of the time of the call to reset.

If the method markSupported returns false, then: * The call to reset may throw an **IOException** (p. 1545). * If an **IOException** (p. 1545) is not thrown, then the stream is reset to a fixed state that depends on the particular type of the input stream and how it was created. The bytes that will be supplied to subsequent callers of the read method depend on the particular type of the input stream.

The default implementation of this method throws an **IOException** (p. 1545).

Exceptions

IOException (p. 1545)	if an I/O error occurs.
---------------------------------	-------------------------

Reimplemented from **decaf::io::FilterInputStream** (p. 1339).

6.87.3.8 virtual long long decaf::io::BufferedInputStream::skip (long long num) [virtual]

Skips over and discards n bytes of data from this input stream.

The skip method may, for a variety of reasons, end up skipping over some smaller number of bytes, possibly 0. This may result from any of a number of conditions; reaching end of file before n bytes have been skipped is only one possibility. The actual number of bytes skipped is returned.

The skip method of **InputStream** (p. 1464) creates a byte array and then repeatedly reads into it until num bytes have been read or the end of the stream has been reached. Subclasses are encouraged to provide a more efficient implementation of this method.

Parameters

<i>num</i>	The number of bytes to skip.
------------	------------------------------

Returns

total bytes skipped

Exceptions

<i>IOException</i> (p. 1545)	if an I/O error occurs.
<i>Unsupported-Operation-Exception</i>	if the concrete stream class does not support skipping bytes.

Reimplemented from **decaf::io::FilterInputStream** (p. 1340).

The documentation for this class was generated from the following file:

- src/main/decaf/io/**BufferedInputStream.h**

6.88 decaf::io::BufferedOutputStream Class Reference

Wrapper around another output stream that buffers output before writing to the target output stream.

```
#include <src/main/decaf/io/BufferedOutputStream.h>
```

Inheritance diagram for decaf::io::BufferedOutputStream:

Public Member Functions

- **BufferedOutputStream** (**OutputStream** *stream, bool **own**=false)
Constructor.
- **BufferedOutputStream** (**OutputStream** *stream, int bufferSize, bool **own**=false)
Constructor.
- virtual ~**BufferedOutputStream** ()
- virtual void **flush** ()
inheritDoc
- virtual void **doWriteByte** (unsigned char c)
- virtual void **doWriteArray** (const unsigned char *buffer, int size)
- virtual void **doWriteArrayBounded** (const unsigned char *buffer, int size, int offset, int length)

6.88.1 Detailed Description

Wrapper around another output stream that buffers output before writing to the target output stream.

6.88.2 Constructor & Destructor Documentation

6.88.2.1 `decaf::io::BufferedOutputStream::BufferedOutputStream (OutputStream * stream, bool own = false)`

Constructor.

Parameters

<i>stream</i>	The target output stream.
<i>own</i>	Indicates if this class owns the stream pointer.

6.88.2.2 `decaf::io::BufferedOutputStream::BufferedOutputStream (OutputStream * stream, int bufferSize, bool own = false)`

Constructor.

Parameters

<i>stream</i>	The target output stream.
<i>bufferSize</i>	The size for the internal buffer.
<i>own</i>	Indicates if this class owns the stream pointer.

Exceptions

<i>IllegalArgumentException</i>	if the <i>bufferSize</i> given is negative.
---------------------------------	---

6.88.2.3 `virtual decaf::io::BufferedOutputStream::~~BufferedOutputStream ()`
[virtual]

6.88.3 Member Function Documentation

6.88.3.1 `virtual void decaf::io::BufferedOutputStream::doWriteArray (const unsigned char * buffer, int size)` [protected, virtual]

Reimplemented from `decaf::io::FilterOutputStream` (p. 1343).

6.88.3.2 `virtual void decaf::io::BufferedOutputStream::doWriteArrayBounded (const unsigned char * buffer, int size, int offset, int length)` [protected, virtual]

Reimplemented from `decaf::io::FilterOutputStream` (p. 1343).

6.88.3.3 virtual void **decaf::io::BufferedOutputStream::doWriteByte** (unsigned char *c*) [protected, virtual]

Reimplemented from **decaf::io::FilterOutputStream** (p. 1343).

6.88.3.4 virtual void **decaf::io::BufferedOutputStream::flush** () [virtual]

inheritDoc}

Reimplemented from **decaf::io::FilterOutputStream** (p. 1343).

The documentation for this class was generated from the following file:

- src/main/decaf/io/**BufferedOutputStream.h**

6.89 decaf::internal::nio::BufferFactory Class Reference

Factory class used by static methods in the **decaf::nio** (p. 94) package to create the various default version of the NIO interfaces.

```
#include <src/main/decaf/internal/nio/BufferFactory.h>
```

Public Member Functions

- virtual ~**BufferFactory** ()

Static Public Member Functions

- static **decaf::nio::ByteBuffer * createByteBuffer** (int capacity)
Allocates a new byte buffer whose position will be zero its limit will be its capacity and its mark is not set.
- static **decaf::nio::ByteBuffer * createByteBuffer** (unsigned char *buffer, int size, int offset, int length)
Wraps the passed buffer with a new ByteBuffer.
- static **decaf::nio::ByteBuffer * createByteBuffer** (std::vector< unsigned char > &buffer)
Wraps the passed STL Byte Vector in a ByteBuffer.
- static **decaf::nio::CharBuffer * createCharBuffer** (int capacity)
Allocates a new char buffer whose position will be zero its limit will be its capacity and its mark is not set.
- static **decaf::nio::CharBuffer * createCharBuffer** (char *buffer, int size, int offset, int length)
Wraps the passed buffer with a new CharBuffer.
- static **decaf::nio::CharBuffer * createCharBuffer** (std::vector< char > &buffer)

Wraps the passed STL Byte Vector in a CharBuffer.

- static **decaf::nio::DoubleBuffer * createDoubleBuffer** (int capacity)
Allocates a new double buffer whose position will be zero its limit will be its capacity and its mark is not set.
- static **decaf::nio::DoubleBuffer * createDoubleBuffer** (double *buffer, int size, int offset, int length)
Wraps the passed buffer with a new DoubleBuffer.
- static **decaf::nio::DoubleBuffer * createDoubleBuffer** (std::vector< double > &buffer)
Wraps the passed STL Double Vector in a DoubleBuffer.
- static **decaf::nio::FloatBuffer * createFloatBuffer** (int capacity)
Allocates a new float buffer whose position will be zero its limit will be its capacity and its mark is not set.
- static **decaf::nio::FloatBuffer * createFloatBuffer** (float *buffer, int size, int offset, int length)
Wraps the passed buffer with a new FloatBuffer.
- static **decaf::nio::FloatBuffer * createFloatBuffer** (std::vector< float > &buffer)
Wraps the passed STL Float Vector in a FloatBuffer.
- static **decaf::nio::LongBuffer * createLongBuffer** (int capacity)
Allocates a new long long buffer whose position will be zero its limit will be its capacity and its mark is not set.
- static **decaf::nio::LongBuffer * createLongBuffer** (long long *buffer, int size, int offset, int length)
Wraps the passed buffer with a new LongBuffer.
- static **decaf::nio::LongBuffer * createLongBuffer** (std::vector< long long > &buffer)
Wraps the passed STL Long Vector in a LongBuffer.
- static **decaf::nio::IntBuffer * createIntBuffer** (int capacity)
Allocates a new int buffer whose position will be zero its limit will be its capacity and its mark is not set.
- static **decaf::nio::IntBuffer * createIntBuffer** (int *buffer, int size, int offset, int length)
Wraps the passed buffer with a new IntBuffer.
- static **decaf::nio::IntBuffer * createIntBuffer** (std::vector< int > &buffer)
Wraps the passed STL int Vector in a IntBuffer.
- static **decaf::nio::ShortBuffer * createShortBuffer** (int capacity)
Allocates a new short buffer whose position will be zero its limit will be its capacity and its mark is not set.
- static **decaf::nio::ShortBuffer * createShortBuffer** (short *buffer, int size, int offset, int length)
Wraps the passed buffer with a new ShortBuffer.
- static **decaf::nio::ShortBuffer * createShortBuffer** (std::vector< short > &buffer)
Wraps the passed STL Short Vector in a ShortBuffer.

6.89.1 Detailed Description

Factory class used by static methods in the **decaf::nio** (p. 94) package to create the various default version of the NIO interfaces.

Since

1.0

6.89.2 Constructor & Destructor Documentation

6.89.2.1 virtual **decaf::internal::nio::BufferFactory::~~BufferFactory** ()
[inline, virtual]

6.89.3 Member Function Documentation

6.89.3.1 static **decaf::nio::ByteBuffer*** **decaf::internal::nio::BufferFactory::createByteBuffer** (int *capacity*)
[static]

Allocates a new byte buffer whose position will be zero its limit will be its capacity and its mark is not set.

Parameters

<i>capacity</i>	The internal buffer's capacity.
-----------------	---------------------------------

Returns

a newly allocated ByteBuffer which the caller owns.

Exceptions

<i>IndexOutOfBoundsException</i>	if the capacity specified is negative.
----------------------------------	--

6.89.3.2 static **decaf::nio::ByteBuffer*** **decaf::internal::nio::BufferFactory::createByteBuffer** (unsigned char * *buffer*, int *size*, int *offset*, int *length*)
[static]

Wraps the passed buffer with a new ByteBuffer.

The new buffer will be backed by the given byte array; that is, modifications to the buffer will cause the array to be modified and vice versa. The new buffer's capacity will be array length, its position will be offset, its limit will be offset + length, and its mark will be undefined. Its backing array will be the given array, and its array offset will be zero.

Parameters

<i>buffer</i>	The array that will back the new buffer.
<i>size</i>	The size of the specified buffer.
<i>offset</i>	The offset of the subarray to be used.
<i>length</i>	The length of the subarray to be used.

Returns

a new ByteBuffer that is backed by buffer, caller owns the returned pointer.

Exceptions

<i>NullPointerException</i>	if the buffer given in Null.
<i>IndexOutOfBoundsException</i>	if the capacity specified is negative.

6.89.3.3 `static decaf::nio::ByteBuffer* decaf::internal::nio::BufferFactory::createByteBuffer (std::vector< unsigned char > & buffer)`
`[static]`

Wraps the passed STL Byte Vector in a ByteBuffer.

The new buffer will be backed by the given byte array; modifications to the buffer will cause the array to be modified and vice versa. The new buffer's capacity and limit will be `buffer.size()`, its position will be zero, and its mark will be undefined. Its backing array will be the given array, and its array offset will be zero.

Parameters

<i>buffer</i>	The vector that will back the new buffer, the vector must have been sized to the desired size already by calling <code>vector.resize(N)</code> .
---------------	--

Returns

a new ByteBuffer that is backed by buffer, caller owns.

6.89.3.4 `static decaf::nio::CharBuffer* decaf::internal::nio::BufferFactory::createCharBuffer (int capacity)`
`[static]`

Allocates a new char buffer whose position will be zero its limit will be its capacity and its mark is not set.

Parameters

<i>capacity</i>	The internal buffer's capacity.
-----------------	---------------------------------

Returns

a newly allocated CharBuffer which the caller owns.

Exceptions

<i>IndexOutOfBoundsException</i>	if the capacity specified is negative.
----------------------------------	--

6.89.3.5 `static decaf::nio::CharBuffer* decaf::internal::nio::BufferFactory::createCharBuffer (char * buffer, int size, int offset, int length)`
`[static]`

Wraps the passed buffer with a new CharBuffer.

The new buffer will be backed by the given byte array; that is, modifications to the buffer will cause the array to be modified and vice versa. The new buffer's capacity will be array.length, its position will be offset, its limit will be offset + length, and its mark will be undefined. Its backing array will be the given array, and its array offset will be zero.

Parameters

<i>buffer</i>	The array that will back the new buffer.
<i>size</i>	The size of the specified buffer.
<i>offset</i>	The offset of the subarray to be used.
<i>length</i>	The length of the subarray to be used.

Returns

a new CharBuffer that is backed by buffer, caller owns the returned pointer.

Exceptions

<i>NullPointerException</i>	if the buffer given in Null.
<i>IndexOutOfBoundsException</i>	if the capacity specified is negative.

6.89.3.6 `static decaf::nio::CharBuffer* decaf::internal::nio::BufferFactory::createCharBuffer (std::vector< char > & buffer)`
`[static]`

Wraps the passed STL Byte Vector in a CharBuffer.

The new buffer will be backed by the given byte array; modifications to the buffer will cause the array to be modified and vice versa. The new buffer's capacity and limit will be buffer.size(), its position will be zero, and its mark will be undefined. Its backing array will be the given array, and its array offset will be zero.

Parameters

<i>buffer</i>	The vector that will back the new buffer, the vector must have been sized to the desired size already by calling <code>vector.resize(N)</code> .
---------------	--

Returns

a new CharBuffer that is backed by `buffer`, caller owns.

6.89.3.7 `static decaf::nio::DoubleBuffer* decaf::internal::nio::BufferFactory::createDoubleBuffer (int capacity)`
[static]

Allocates a new double buffer whose position will be zero its limit will be its capacity and its mark is not set.

Parameters

<i>capacity</i>	The internal buffer's capacity.
-----------------	---------------------------------

Returns

a newly allocated DoubleBuffer which the caller owns.

Exceptions

<i>IndexOutOfBoundsException</i>	if the capacity specified is negative.
----------------------------------	--

6.89.3.8 `static decaf::nio::DoubleBuffer* decaf::internal::nio::BufferFactory::createDoubleBuffer (double * buffer, int size, int offset, int length)`
[static]

Wraps the passed buffer with a new DoubleBuffer.

The new buffer will be backed by the given byte array; that is, modifications to the buffer will cause the array to be modified and vice versa. The new buffer's capacity will be `array.length`, its position will be `offset`, its limit will be `offset + length`, and its mark will be undefined. Its backing array will be the given array, and its array offset will be zero.

Parameters

<i>buffer</i>	The array that will back the new buffer.
<i>size</i>	The size of the specified buffer.
<i>offset</i>	The offset of the subarray to be used.
<i>length</i>	The length of the subarray to be used.

Returns

a new DoubleBuffer that is backed by buffer, caller owns the returned pointer.

Exceptions

<i>NullPointerException</i>	if the buffer given in Null.
<i>IndexOutOfBoundsException</i>	if the capacity specified is negative.

6.89.3.9 static decaf::nio::DoubleBuffer* decaf::internal::nio::BufferFactory::createDoubleBuffer (std::vector< double > & buffer)
[static]

Wraps the passed STL Double Vector in a DoubleBuffer.

The new buffer will be backed by the given byte array; modifications to the buffer will cause the array to be modified and vice versa. The new buffer's capacity and limit will be buffer.size(), its position will be zero, and its mark will be undefined. Its backing array will be the given array, and its array offset will be zero.

Parameters

<i>buffer</i>	The vector that will back the new buffer, the vector must have been sized to the desired size already by calling vector.resize(N).
---------------	--

Returns

a new DoubleBuffer that is backed by buffer, caller owns.

6.89.3.10 static decaf::nio::FloatBuffer* decaf::internal::nio::BufferFactory::createFloatBuffer (int capacity)
[static]

Allocates a new float buffer whose position will be zero its limit will be its capacity and its mark is not set.

Parameters

<i>capacity</i>	The internal buffer's capacity.
-----------------	---------------------------------

Returns

a newly allocated FloatBuffer which the caller owns.

Exceptions

<i>IndexOutOfBoundsException</i>	if the capacity specified is negative.
----------------------------------	--

6.89.3.11 `static decaf::nio::FloatBuffer* decaf::internal::nio::BufferFactory::createFloatBuffer (float * buffer, int size, int offset, int length)`
`[static]`

Wraps the passed buffer with a new FloatBuffer.

The new buffer will be backed by the given byte array; that is, modifications to the buffer will cause the array to be modified and vice versa. The new buffer's capacity will be `array.length`, its position will be `offset`, its limit will be `offset + length`, and its mark will be undefined. Its backing array will be the given array, and its array offset will be zero.

Parameters

<i>buffer</i>	The array that will back the new buffer.
<i>size</i>	The size of the specified buffer.
<i>offset</i>	The offset of the subarray to be used.
<i>length</i>	The length of the subarray to be used.

Returns

a new FloatBuffer that is backed by `buffer`, caller owns the returned pointer.

Exceptions

<i>NullPointerException</i>	if the buffer given in Null.
<i>IndexOutOfBoundsException</i>	if the capacity specified is negative.

6.89.3.12 `static decaf::nio::FloatBuffer* decaf::internal::nio::BufferFactory::createFloatBuffer (std::vector< float > & buffer)`
`[static]`

Wraps the passed STL Float Vector in a FloatBuffer.

The new buffer will be backed by the given byte array; modifications to the buffer will cause the array to be modified and vice versa. The new buffer's capacity and limit will be `buffer.size()`, its position will be zero, and its mark will be undefined. Its backing array will be the given array, and its array offset will be zero.

Parameters

<i>buffer</i>	The vector that will back the new buffer, the vector must have been sized to the desired size already by calling <code>vector.resize(N)</code> .
---------------	--

Returns

a new FloatBuffer that is backed by `buffer`, caller owns.

6.89.3.13 static decaf::nio::IntBuffer* decaf::internal::nio::BufferFactory::create-
IntBuffer (int *capacity*) [static]

Allocates a new int buffer whose position will be zero its limit will be its capacity and its mark is not set.

Parameters

<i>capacity</i>	The internal buffer's capacity.
-----------------	---------------------------------

Returns

a newly allocated IntBuffer which the caller owns.

Exceptions

<i>IndexOutOfBoundsException</i>	if the capacity specified is negative.
----------------------------------	--

6.89.3.14 static decaf::nio::IntBuffer* decaf::internal::nio::BufferFactory-
::createIntBuffer (int * *buffer*, int *size*, int *offset*, int *length*)
[static]

Wraps the passed buffer with a new IntBuffer.

The new buffer will be backed by the given byte array; that is, modifications to the buffer will cause the array to be modified and vice versa. The new buffer's capacity will be array.length, its position will be offset, its limit will be offset + length, and its mark will be undefined. Its backing array will be the given array, and its array offset will be zero.

Parameters

<i>buffer</i>	The array that will back the new buffer.
<i>size</i>	The size of the specified buffer.
<i>offset</i>	The offset of the subarray to be used.
<i>length</i>	The length of the subarray to be used.

Returns

a new IntBuffer that is backed by buffer, caller owns the returned pointer.

Exceptions

<i>NullPointerException</i>	if the buffer given in Null.
<i>IndexOutOfBoundsException</i>	if the capacity specified is negative.

6.89.3.15 `static decaf::nio::IntBuffer* decaf::internal::nio::BufferFactory::createIntBuffer (std::vector< int > & buffer)`
`[static]`

Wraps the passed STL int Vector in a IntBuffer.

The new buffer will be backed by the given byte array; modifications to the buffer will cause the array to be modified and vice versa. The new buffer's capacity and limit will be `buffer.size()`, its position will be zero, and its mark will be undefined. Its backing array will be the given array, and its array offset will be zero.

Parameters

<i>buffer</i>	The vector that will back the new buffer, the vector must have been sized to the desired size already by calling <code>vector.resize(N)</code> .
---------------	--

Returns

a new IntBuffer that is backed by buffer, caller owns.

6.89.3.16 `static decaf::nio::LongBuffer* decaf::internal::nio::BufferFactory::createLongBuffer (int capacity)`
`[static]`

Allocates a new long long buffer whose position will be zero its limit will be its capacity and its mark is not set.

Parameters

<i>capacity</i>	- the internal buffer's capacity.
-----------------	-----------------------------------

Returns

a newly allocated DoubleBuffer which the caller owns.

Exceptions

<i>IndexOutOfBoundsException</i>	if the capacity specified is negative.
----------------------------------	--

6.89.3.17 `static decaf::nio::LongBuffer* decaf::internal::nio::BufferFactory::createLongBuffer (long long * buffer, int size, int offset, int length)`
`[static]`

Wraps the passed buffer with a new LongBuffer.

The new buffer will be backed by the given byte array; that is, modifications to the buffer will cause the array to be modified and vice versa. The new buffer's capacity will be

array.length, its position will be offset, its limit will be offset + length, and its mark will be undefined. Its backing array will be the given array, and its array offset will be zero.

Parameters

<i>buffer</i>	The array that will back the new buffer.
<i>size</i>	The size of the specified buffer.
<i>offset</i>	The offset of the subarray to be used.
<i>length</i>	The length of the subarray to be used.

Returns

a new LongBuffer that is backed by buffer, caller owns the returned pointer.

Exceptions

<i>NullPointerException</i>	if the buffer given in Null.
<i>IndexOutOfBoundsException</i>	if the capacity specified is negative.

6.89.3.18 `static decaf::nio::LongBuffer* decaf::internal::nio::BufferFactory::createLongBuffer (std::vector< long long > & buffer)`
[static]

Wraps the passed STL Long Vector in a LongBuffer.

The new buffer will be backed by the given byte array; modifications to the buffer will cause the array to be modified and vice versa. The new buffer's capacity and limit will be buffer.size(), its position will be zero, and its mark will be undefined. Its backing array will be the given array, and its array offset will be zero.

Parameters

<i>buffer</i>	The vector that will back the new buffer, the vector must have been sized to the desired size already by calling vector.resize(N).
---------------	--

Returns

a new LongBuffer that is backed by buffer, caller owns.

6.89.3.19 `static decaf::nio::ShortBuffer* decaf::internal::nio::BufferFactory::createShortBuffer (int capacity)`
[static]

Allocates a new short buffer whose position will be zero its limit will be its capacity and its mark is not set.

Parameters

<i>capacity</i>	The internal buffer's capacity.
-----------------	---------------------------------

Returns

a newly allocated ShortBuffer which the caller owns.

Exceptions

<i>IndexOutOfBoundsException</i>	if the capacity specified is negative.
----------------------------------	--

6.89.3.20 `static decaf::nio::ShortBuffer* decaf::internal::nio::BufferFactory::createShortBuffer (short * buffer, int size, int offset, int length)`
`[static]`

Wraps the passed buffer with a new ShortBuffer.

The new buffer will be backed by the given byte array; that is, modifications to the buffer will cause the array to be modified and vice versa. The new buffer's capacity will be `array.length`, its position will be `offset`, its limit will be `offset + length`, and its mark will be undefined. Its backing array will be the given array, and its array offset will be zero.

Parameters

<i>buffer</i>	The array that will back the new buffer.
<i>size</i>	The size of the specified buffer.
<i>offset</i>	The offset of the subarray to be used.
<i>length</i>	The length of the subarray to be used.

Returns

a new ShortBuffer that is backed by `buffer`, caller owns the returned pointer.

Exceptions

<i>NullPointerException</i>	if the buffer given in Null.
<i>IndexOutOfBoundsException</i>	if the capacity specified is negative.

6.89.3.21 `static decaf::nio::ShortBuffer* decaf::internal::nio::BufferFactory::createShortBuffer (std::vector< short > & buffer)`
`[static]`

Wraps the passed STL Short Vector in a ShortBuffer.

The new buffer will be backed by the given byte array; modifications to the buffer will cause the array to be modified and vice versa. The new buffer's capacity and limit will be `buffer.size()`, its position will be zero, and its mark will be undefined. Its backing array will be the given array, and its array offset will be zero.

Parameters

<i>buffer</i>	The vector that will back the new buffer, the vector must have been sized to the desired size already by calling <code>vector.resize(N)</code> .
---------------	--

Returns

a new `DoubleBuffer` that is backed by `buffer`, caller owns.

The documentation for this class was generated from the following file:

- `src/main/decaf/internal/nio/BufferFactory.h`

6.90 decaf::nio::BufferOverflowException Class Reference

```
#include <src/main/decaf/nio/BufferOverflowException.h>
```

Inheritance diagram for `decaf::nio::BufferOverflowException`:

Public Member Functions

- **BufferOverflowException** () throw ()
Default Constructor.
- **BufferOverflowException** (const lang::Exception &ex) throw ()
Copy Constructor.
- **BufferOverflowException** (const BufferOverflowException &ex) throw ()
Copy Constructor.
- **BufferOverflowException** (const char *file, const int lineNumber, const std::exception *cause, const char *msg,...) throw ()
Constructor - Initializes the file name and line number where this message occurred.
- **BufferOverflowException** (const std::exception *cause) throw ()
Constructor.
- **BufferOverflowException** (const char *file, const int lineNumber, const char *msg,...) throw ()
Constructor.
- virtual **BufferOverflowException** * clone () const
Clones this exception.
- virtual ~**BufferOverflowException** () throw ()

6.90.1 Constructor & Destructor Documentation

6.90.1.1 `decaf::nio::BufferOverflowException::BufferOverflowException () throw ()` `[inline]`

Default Constructor.

6.90.1.2 `decaf::nio::BufferOverflowException::BufferOverflowException (const lang::Exception & ex) throw ()` `[inline]`

Copy Constructor.

Parameters

<i>ex</i>	the exception to copy
-----------	-----------------------

6.90.1.3 `decaf::nio::BufferOverflowException::BufferOverflowException (const BufferOverflowException & ex) throw ()` `[inline]`

Copy Constructor.

Parameters

<i>ex</i>	the exception to copy, which is an instance of this type
-----------	--

6.90.1.4 `decaf::nio::BufferOverflowException::BufferOverflowException (const char * file, const int lineNumber, const std::exception * cause, const char * msg, ...) throw ()` `[inline]`

Constructor - Initializes the file name and line number where this message occurred.

Sets the message to report, using an optional list of arguments to parse into the message

Parameters

<i>file</i>	The file name where exception occurs
<i>lineNumber</i>	The line number where the exception occurred.
<i>cause</i>	The exception that was the cause for this one to be thrown.
<i>msg</i>	The message to report
<i>...</i>	list of primitives that are formatted into the message

6.90.1.5 `decaf::nio::BufferOverflowException::BufferOverflowException (const std::exception * cause) throw ()` `[inline]`

Constructor.

Parameters

<i>cause</i>	Pointer to the exception that caused this one to be thrown, the object is cloned caller retains ownership.
--------------	--

6.90.1.6 **decaf::nio::BufferOverflowException::BufferOverflowException** (const char * *file*, const int *lineNumber*, const char * *msg*, ...) throw () [inline]

Constructor.

Parameters

<i>file</i>	The file name where exception occurs
<i>lineNumber</i>	The line number where the exception occurred.
<i>msg</i>	The message to report
...	list of primitives that are formatted into the message

6.90.1.7 **virtual decaf::nio::BufferOverflowException::~~BufferOverflowException** () throw () [inline, virtual]

6.90.2 Member Function Documentation

6.90.2.1 **virtual BufferOverflowException* decaf::nio::BufferOverflowException::clone** () const [inline, virtual]

Clones this exception.

This is useful for cases where you need to preserve the type of the original exception as well as the message. All subclasses should override.

Reimplemented from **decaf::lang::Exception** (p. 1282).

The documentation for this class was generated from the following file:

- src/main/decaf/nio/**BufferOverflowException.h**

6.91 decaf::nio::BufferUnderflowException Class Reference

```
#include <src/main/decaf/nio/BufferUnderflowException.h>
```

Inheritance diagram for decaf::nio::BufferUnderflowException:

Public Member Functions

- **BufferUnderflowException** () throw ()

Default Constructor.

- **BufferUnderflowException** (const lang::Exception &ex) throw ()

Copy Constructor.

- **BufferUnderflowException** (const **BufferUnderflowException** &ex) throw ()

Copy Constructor.

- **BufferUnderflowException** (const char *file, const int lineNumber, const std::exception *cause, const char *msg,...) throw ()

Constructor - Initializes the file name and line number where this message occurred.

- **BufferUnderflowException** (const std::exception *cause) throw ()

Constructor.

- **BufferUnderflowException** (const char *file, const int lineNumber, const char *msg,...) throw ()

Constructor.

- virtual **BufferUnderflowException** * clone () const

Clones this exception.

- virtual ~**BufferUnderflowException** () throw ()

6.91.1 Constructor & Destructor Documentation

6.91.1.1 **decaf::nio::BufferUnderflowException::BufferUnderflowException ()**
throw () [inline]

Default Constructor.

6.91.1.2 **decaf::nio::BufferUnderflowException::BufferUnderflowException (**
const lang::Exception & ex) throw () [inline]

Copy Constructor.

Parameters

ex	the exception to copy
----	-----------------------

6.91.1.3 **decaf::nio::BufferUnderflowException::BufferUnderflowException (**
const **BufferUnderflowException** & ex) throw () [inline]

Copy Constructor.

Parameters

ex	the exception to copy, which is an instance of this type
----	--

6.91.1.4 **decaf::nio::BufferUnderflowException::BufferUnderflowException (**
const char * *file*, const int *lineNumber*, const std::exception * *cause*, const char * *msg*, ...) throw () `[inline]`

Constructor - Initializes the file name and line number where this message occurred.

Sets the message to report, using an optional list of arguments to parse into the message

Parameters

<i>file</i>	The file name where exception occurs
<i>lineNumber</i>	The line number where the exception occurred.
<i>cause</i>	The exception that was the cause for this one to be thrown.
<i>msg</i>	The message to report
...	list of primitives that are formatted into the message

6.91.1.5 **decaf::nio::BufferUnderflowException::BufferUnderflowException (**
const std::exception * *cause*) throw () `[inline]`

Constructor.

Parameters

<i>cause</i>	Pointer to the exception that caused this one to be thrown, the object is cloned caller retains ownership.
--------------	--

6.91.1.6 **decaf::nio::BufferUnderflowException::BufferUnderflowException (**
const char * *file*, const int *lineNumber*, const char * *msg*, ...) throw () `[inline]`

Constructor.

Parameters

<i>file</i>	The file name where exception occurs
<i>lineNumber</i>	The line number where the exception occurred.
<i>msg</i>	The message to report
...	list of primitives that are formatted into the message

6.91.1.7 **virtual decaf::nio::BufferUnderflowException::~~-**
BufferUnderflowException () throw () `[inline,`
virtual]

6.91.2 Member Function Documentation

6.91.2.1 virtual **BufferUnderflowException*** **decaf::nio::BufferUnderflowException::clone** () const [inline, virtual]

Clones this exception.

This is useful for cases where you need to preserve the type of the original exception as well as the message. All subclasses should override.

Reimplemented from **decaf::lang::Exception** (p. 1282).

The documentation for this class was generated from the following file:

- src/main/decaf/nio/**BufferUnderflowException.h**

6.92 decaf::lang::Byte Class Reference

```
#include <src/main/decaf/lang/Byte.h>
```

Inheritance diagram for **decaf::lang::Byte**:

Public Member Functions

- **Byte** (unsigned char value)
- **Byte** (const std::string &value)
*Creates a new **Byte** (p. 614) instance from the given string.*
- virtual ~**Byte** ()
- virtual int **compareTo** (const **Byte** &c) const
*Compares this **Byte** (p. 614) instance with another.*
- virtual bool **operator==** (const **Byte** &c) const
Compares equality between this object and the one passed.
- virtual bool **operator<** (const **Byte** &c) const
Compares this object to another and returns true if this object is considered to be less than the one passed.
- virtual int **compareTo** (const unsigned char &c) const
*Compares this **Byte** (p. 614) instance with a char type.*
- virtual bool **operator==** (const unsigned char &c) const
Compares equality between this object and the one passed.
- virtual bool **operator<** (const unsigned char &c) const
Compares this object to another and returns true if this object is considered to be less than the one passed.
- bool **equals** (const **Byte** &c) const
- bool **equals** (const unsigned char &c) const
- std::string **toString** () const
- virtual double **doubleValue** () const

Answers the double value which the receiver represents.

- virtual float **floatValue** () const

Answers the float value which the receiver represents.

- virtual unsigned char **byteValue** () const

Answers the byte value which the receiver represents.

- virtual short **shortValue** () const

Answers the short value which the receiver represents.

- virtual int **intValue** () const

Answers the int value which the receiver represents.

- virtual long long **longValue** () const

Answers the long value which the receiver represents.

Static Public Member Functions

- static std::string **toString** (unsigned char value)

- static **Byte decode** (const std::string &value)

*Decodes a **String** (p. 2620) into a **Byte** (p. 614).*

- static unsigned char **parseByte** (const std::string &s, int radix)

Parses the string argument as a signed unsigned char in the radix specified by the second argument.

- static unsigned char **parseByte** (const std::string &s)

Parses the string argument as a signed decimal unsigned char.

- static **Byte valueOf** (unsigned char value)

*Returns a **Character** (p. 765) instance representing the specified char value.*

- static **Byte valueOf** (const std::string &value)

*Returns a **Byte** (p. 614) object holding the value given by the specified std::string.*

- static **Byte valueOf** (const std::string &value, int radix)

*Returns a **Byte** (p. 614) object holding the value extracted from the specified std::string when parsed with the radix given by the second argument.*

Static Public Attributes

- static const unsigned char **MIN_VALUE** = 0x7F

The minimum value that a unsigned char can take on.

- static const unsigned char **MAX_VALUE** = 0x80

The maximum value that a unsigned char can take on.

- static const int **SIZE** = 8

The size of the primitive charactor in bits.

6.92.1 Constructor & Destructor Documentation

6.92.1.1 `decaf::lang::Byte::Byte (unsigned char value)`

Parameters

<i>value</i>	- the primitive value to wrap
--------------	-------------------------------

6.92.1.2 `decaf::lang::Byte::Byte (const std::string & value)`

Creates a new **Byte** (p. 614) instance from the given string.

Parameters

<i>value</i>	The string to convert to an unsigned char
--------------	---

Exceptions

<i>NumberFormatException</i>	if the string is not a valid byte.
------------------------------	------------------------------------

6.92.1.3 `virtual decaf::lang::Byte::~~Byte () [inline, virtual]`

6.92.2 Member Function Documentation

6.92.2.1 `virtual unsigned char decaf::lang::Byte::byteValue () const [inline, virtual]`

Answers the byte value which the receiver represents.

Returns

byte the value of the receiver.

Reimplemented from **decaf::lang::Number** (p. 1993).

6.92.2.2 `virtual int decaf::lang::Byte::compareTo (const Byte & c) const [inline, virtual]`

Compares this **Byte** (p. 614) instance with another.

Parameters

<i>c</i>	- the Byte (p. 614) instance to be compared
----------	--

Returns

zero if this object represents the same char value as the argument; a positive value if this object represents a value greater than the passed in value, and -1 if this object represents a value less than the passed in value.

Implements **decaf::lang::Comparable**< **Byte** > (p. 886).

6.92.2.3 `virtual int decaf::lang::Byte::compareTo (const unsigned char & c) const`
`[inline, virtual]`

Compares this **Byte** (p. 614) instance with a char type.

Parameters

<code>c</code>	- the char instance to be compared
----------------	------------------------------------

Returns

zero if this object represents the same char value as the argument; a positive value if this object represents a value greater than the passed in value, and -1 if this object represents a value less than the passed in value.

Implements **decaf::lang::Comparable**< **unsigned char** > (p. 886).

6.92.2.4 `static Byte decaf::lang::Byte::decode (const std::string & value)`
`[static]`

Decodes a **String** (p. 2620) into a **Byte** (p. 614).

Accepts decimal, hexadecimal, and octal numbers given by the following grammar:

The sequence of characters following an (optional) negative sign and/or radix specifier ("0x", "0X", "#", or leading zero) is parsed as by the **Byte::parseByte** (p. 620) method with the indicated radix (10, 16, or 8). This sequence of characters must represent a positive value or a **NumberFormatException** will be thrown. The result is negated if first character of the specified **String** (p. 2620) is the minus sign. No whitespace characters are permitted in the string.

Parameters

<code>value</code>	- The string to decode
--------------------	------------------------

Returns

a **Byte** (p. 614) object containing the decoded value

Exceptions

<i>NumberFormatException</i>	if the string is not formatted correctly.
------------------------------	---

6.92.2.5 `virtual double decaf::lang::Byte::doubleValue () const [inline, virtual]`

Answers the double value which the receiver represents.

Returns

double the value of the receiver.

Implements **decaf::lang::Number** (p. 1993).

6.92.2.6 `bool decaf::lang::Byte::equals (const Byte & c) const [inline, virtual]`

Returns

true if the two **Byte** (p. 614) Objects have the same value.

Implements **decaf::lang::Comparable< Byte >** (p. 887).

6.92.2.7 `bool decaf::lang::Byte::equals (const unsigned char & c) const [inline, virtual]`

Returns

true if the two Bytes have the same value.

Implements **decaf::lang::Comparable< unsigned char >** (p. 887).

6.92.2.8 `virtual float decaf::lang::Byte::floatValue () const [inline, virtual]`

Answers the float value which the receiver represents.

Returns

float the value of the receiver.

Implements **decaf::lang::Number** (p. 1993).

6.92.2.9 `virtual int decaf::lang::Byte::intValue () const [inline, virtual]`

Answers the int value which the receiver represents.

Returns

int the value of the receiver.

Implements **decaf::lang::Number** (p. 1993).

6.92.2.10 `virtual long long decaf::lang::Byte::longValue () const [inline, virtual]`

Answers the long value which the receiver represents.

Returns

long long the value of the receiver.

Implements **decaf::lang::Number** (p. 1994).

6.92.2.11 `virtual bool decaf::lang::Byte::operator< (const Byte & c) const [inline, virtual]`

Compares this object to another and returns true if this object is considered to be less than the one passed.

This

Parameters

<code>c</code>	- the value to be compared to this one.
----------------	---

Returns

true if this object is equal to the one passed.

Implements **decaf::lang::Comparable< Byte >** (p. 887).

6.92.2.12 `virtual bool decaf::lang::Byte::operator< (const unsigned char & c) const [inline, virtual]`

Compares this object to another and returns true if this object is considered to be less than the one passed.

This

Parameters

<code>c</code>	- the value to be compared to this one.
----------------	---

Returns

true if this object is equal to the one passed.

Implements **decaf::lang::Comparable< unsigned char >** (p. 887).

6.92.2.13 `virtual bool decaf::lang::Byte::operator==(const Byte & c) const` `[inline, virtual]`

Compares equality between this object and the one passed.

Parameters

<code>c</code>	- the value to be compared to this one.
----------------	---

Returns

true if this object is equal to the one passed.

Implements **decaf::lang::Comparable**< **Byte** > (p. 888).

6.92.2.14 `virtual bool decaf::lang::Byte::operator==(const unsigned char & c) const` `[inline, virtual]`

Compares equality between this object and the one passed.

Parameters

<code>c</code>	- the value to be compared to this one.
----------------	---

Returns

true if this object is equal to the one passed.

Implements **decaf::lang::Comparable**< **unsigned char** > (p. 888).

6.92.2.15 `static unsigned char decaf::lang::Byte::parseByte (const std::string & s, int radix)` `[static]`

Parses the string argument as a signed unsigned char in the radix specified by the second argument.

The characters in the string must all be digits, of the specified radix (as determined by whether **Character.digit(char, int)** (p. 768) returns a nonnegative value) except that the first character may be an ASCII minus sign '-' to indicate a negative value. The resulting byte value is returned.

An exception of type **NumberFormatException** is thrown if any of the following situations occurs: * The first argument is null or is a string of length zero. * The radix is either smaller than **Character.MIN_RADIX** (p. 773) or larger than **Character::MAX_RADIX** (p. 772). * Any character of the string is not a digit of the specified radix, except that the first character may be a minus sign '-' provided that the string is longer than length 1. * The value represented by the string is not a value of type unsigned char.

Parameters

<i>s</i>	- the String (p. 2620) containing the unsigned char to be parsed
<i>radix</i>	- the radix to be used while parsing <i>s</i>

Returns

the unsigned char represented by the string argument in the specified radix.

Exceptions

<i>NumberFormatException</i>	- If String (p. 2620) does not contain a parsable unsigned char.
------------------------------	---

6.92.2.16 static unsigned char decaf::lang::Byte::parseByte (const std::string & s)
[static]

Parses the string argument as a signed decimal unsigned char.

The characters in the string must all be decimal digits, except that the first character may be an ASCII minus sign '-' to indicate a negative value. The resulting unsigned char value is returned, exactly as if the argument and the radix 10 were given as arguments to the parseByte(const std::string, int) method.

Parameters

<i>s</i>	- String (p. 2620) to convert to a unsigned char
----------	---

Returns

the converted unsigned char value

Exceptions

<i>NumberFormatException</i>	if the string is not a unsigned char.
------------------------------	---------------------------------------

6.92.2.17 virtual short decaf::lang::Byte::shortValue () const [inline, virtual]

Answers the short value which the receiver represents.

Returns

short the value of the receiver.

Reimplemented from **decaf::lang::Number** (p. 1994).

6.92.2.18 `std::string decaf::lang::Byte::toString () const`

Returns

this **Byte** (p. 614) Object as a **String** (p. 2620) Representation

6.92.2.19 `static std::string decaf::lang::Byte::toString (unsigned char value)` [static]

Returns

a string representing the primitive value as Base 10

6.92.2.20 `static Byte decaf::lang::Byte::valueOf (unsigned char value)` [inline, static]

Returns a **Character** (p. 765) instance representing the specified char value.

Parameters

<i>value</i>	- the primitive char to wrap.
--------------	-------------------------------

Returns

a new **Character** (p. 765) instance that wraps this value.

6.92.2.21 `static Byte decaf::lang::Byte::valueOf (const std::string & value)` [static]

Returns a **Byte** (p. 614) object holding the value given by the specified std::string.

The argument is interpreted as representing a signed decimal unsigned char, exactly as if the argument were given to the `parseByte(std::string)` method. The result is a **Byte** (p. 614) object that represents the unsigned char value specified by the string.

Parameters

<i>value</i>	- std::string to parse as base 10
--------------	-----------------------------------

Returns

new **Byte** (p. 614) Object wrapping the primitive

Exceptions

<i>NumberFormatException</i>	if the string is not a decimal unsigned char.
------------------------------	---

6.92.2.22 `static Byte decaf::lang::Byte::valueOf (const std::string & value, int radix)`
`[static]`

Returns a **Byte** (p. 614) object holding the value extracted from the specified `std::string` when parsed with the radix given by the second argument.

The first argument is interpreted as representing a signed unsigned char in the radix specified by the second argument, exactly as if the argument were given to the `parse-Byte(std::string, int)` method. The result is a **Byte** (p. 614) object that represents the unsigned char value specified by the string.

Parameters

<i>value</i>	- <code>std::string</code> to parse as base (<i>radix</i>)
<i>radix</i>	- base of the string to parse.

Returns

new **Byte** (p. 614) Object wrapping the primitive

Exceptions

<i>NumberFormatException</i> <i>Exception</i>	if the string is not a valid unsigned char.
--	---

6.92.3 Field Documentation

6.92.3.1 `const unsigned char decaf::lang::Byte::MAX_VALUE = 0x80` `[static]`

The maximum value that a unsigned char can take on.

6.92.3.2 `const unsigned char decaf::lang::Byte::MIN_VALUE = 0x7F` `[static]`

The minimum value that a unsigned char can take on.

6.92.3.3 `const int decaf::lang::Byte::SIZE = 8` `[static]`

The size of the primitive character in bits.

The documentation for this class was generated from the following file:

- `src/main/decaf/lang/Byte.h`

6.93 decaf::internal::util::ByteArrayAdapter Class Reference

This class adapts primitive type arrays to a base byte array so that the classes can inter-operate on the same base byte array without copying data.

```
#include <src/main/decaf/internal/util/ByteArrayAdapter.h>
```

Data Structures

- union **Array**

Public Member Functions

- **ByteArrayAdapter** (int size)
Creates a byte array object that is allocated internally and is then owned and deleted when this object is deleted.
- **ByteArrayAdapter** (unsigned char *array, int size, bool own=false)
Creates a byte array object that wraps the given array.
- **ByteArrayAdapter** (char *array, int size, bool own=false)
Creates a byte array object that wraps the given array.
- **ByteArrayAdapter** (double *array, int size, bool own=false)
Creates a byte array object that wraps the given array.
- **ByteArrayAdapter** (float *array, int size, bool own=false)
Creates a byte array object that wraps the given array.
- **ByteArrayAdapter** (long long *array, int size, bool own=false)
Creates a byte array object that wraps the given array.
- **ByteArrayAdapter** (int *array, int size, bool own=false)
Creates a byte array object that wraps the given array.
- **ByteArrayAdapter** (short *array, int size, bool own=false)
Creates a byte array object that wraps the given array.
- virtual ~**ByteArrayAdapter** ()
- virtual int **getCapacity** () const
Gets the size of the underlying array.
- virtual int **getCharCapacity** () const
Gets the size of the underlying array as if it contains chars.
- virtual int **getDoubleCapacity** () const
Gets the size of the underlying array as if it contains doubles.
- virtual int **getFloatCapacity** () const
Gets the size of the underlying array as if it contains doubles.
- virtual int **getLongCapacity** () const
Gets the size of the underlying array as if it contains doubles.
- virtual int **getIntCapacity** () const
Gets the size of the underlying array as if it contains ints.

- virtual int **getShortCapacity** () const
Gets the size of the underlying array as if it contains shorts.
- virtual unsigned char * **getByteArray** ()
Gets the pointer to the array we are wrapping.
- virtual char * **getCharArray** ()
Gets the pointer to the array we are wrapping.
- virtual short * **getShortArray** ()
Gets the pointer to the array we are wrapping.
- virtual int * **getIntArray** ()
Gets the pointer to the array we are wrapping.
- virtual long long * **getLongArray** ()
Gets the pointer to the array we are wrapping.
- virtual double * **getDoubleArray** ()
Gets the pointer to the array we are wrapping.
- virtual float * **getFloatArray** ()
Gets the pointer to the array we are wrapping.
- virtual void **read** (unsigned char *buffer, int size, int offset, int length) const
Reads from the Byte array starting at the specified offset and reading the specified length.
- virtual void **write** (unsigned char *buffer, int size, int offset, int length)
Writes from the Byte array given, starting at the specified offset and writing the specified amount of data into this objects internal array.
- virtual void **resize** (int size)
Resizes the underlying array to the new given size, preserving all the Data that was previously in the array, unless the resize is smaller than the current size in which case only the data that will fit into the new array is preserved.
- virtual void **clear** () throw ()
Clear all data from that Array, setting the underlying bytes to zero.
- unsigned char & **operator[]** (int index)
*Allows the **ByteArrayAdapter** (p. 624) to be indexed as a standard array.*
- const unsigned char & **operator[]** (int index) const
- virtual unsigned char **get** (int index) const
Absolute get method.
- virtual char **getChar** (int index) const
Reads one byte at the given index and returns it.
- virtual double **getDouble** (int index) const
Reads eight bytes at the given index and returns it.
- virtual double **getDoubleAt** (int index) const
Reads eight bytes at the given byte index and returns it.
- virtual float **getFloat** (int index) const
Reads four bytes at the given index and returns it.
- virtual float **getFloatAt** (int index) const
Reads four bytes at the given byte index and returns it.
- virtual long long **getLong** (int index) const

- Reads eight bytes at the given index and returns it.*
- virtual long long **getLongAt** (int index) const
- Reads eight bytes at the given byte index and returns it.*
- virtual int **getInt** (int index) const
- Reads four bytes at the given index and returns it.*
- virtual int **getIntAt** (int index) const
- Reads four bytes at the given byte index and returns it.*
- virtual short **getShort** (int index) const
- Reads two bytes at the given index and returns it.*
- virtual short **getShortAt** (int index) const
- Reads two bytes at the given byte index and returns it.*
- virtual **ByteArrayAdapter** & **put** (int index, unsigned char value)
- Writes the given byte into this buffer at the given index.*
- virtual **ByteArrayAdapter** & **putChar** (int index, char value)
- Writes one byte containing the given value, into this buffer at the given index.*
- virtual **ByteArrayAdapter** & **putDouble** (int index, double value)
- Writes eight bytes containing the given value, into this buffer at the given index.*
- virtual **ByteArrayAdapter** & **putDoubleAt** (int index, double value)
- Writes eight bytes containing the given value, into this buffer at the given byte index.*
- virtual **ByteArrayAdapter** & **putFloat** (int index, float value)
- Writes four bytes containing the given value, into this buffer at the given index.*
- virtual **ByteArrayAdapter** & **putFloatAt** (int index, float value)
- Writes four bytes containing the given value, into this buffer at the given byte index.*
- virtual **ByteArrayAdapter** & **putLong** (int index, long long value)
- Writes eight bytes containing the given value, into this buffer at the given index.*
- virtual **ByteArrayAdapter** & **putLongAt** (int index, long long value)
- Writes eight bytes containing the given value, into this buffer at the given byte index.*
- virtual **ByteArrayAdapter** & **putInt** (int index, int value)
- Writes four bytes containing the given value, into this buffer at the given index.*
- virtual **ByteArrayAdapter** & **putIntAt** (int index, int value)
- Writes four bytes containing the given value, into this buffer at the given byte index.*
- virtual **ByteArrayAdapter** & **putShort** (int index, short value)
- Writes two bytes containing the given value, into this buffer at the given index.*
- virtual **ByteArrayAdapter** & **putShortAt** (int index, short value)
- Writes two bytes containing the given value, into this buffer at the given byte index.*

6.93.1 Detailed Description

This class adapts primitive type arrays to a base byte array so that the classes can inter-operate on the same base byte array without copying data.

All the array types are mapped down to a byte array and methods are supplied for accessing the data in any of the primitive type forms.

Methods in this class that do not return a specific value return a reference to this object so that calls can be chained.

Since

1.0

6.93.2 Constructor & Destructor Documentation

6.93.2.1 decaf::internal::util::ByteArrayAdapter::ByteArrayAdapter (int size)

Creates a byte array object that is allocated internally and is then owned and deleted when this object is deleted.

The array is initially created with all elements initialized to zero.

Parameters

<i>size</i>	The size of the array, this is the limit we read and write to.
-------------	--

Exceptions

<i>IllegalArgumentException</i>	if size is negative.
---------------------------------	----------------------

6.93.2.2 decaf::internal::util::ByteArrayAdapter::ByteArrayAdapter (unsigned char * array, int size, bool own = false)

Creates a byte array object that wraps the given array.

If the own flag is set then it will delete this array when this object is deleted.

Parameters

<i>array</i>	The physical array to wrap.
<i>size</i>	The size of the array, this is the limit we read and write to.
<i>own</i>	Indicates if this class is now the owner of the pointer.

Exceptions

<i>NullPointerException</i>	if buffer is NULL
<i>IndexOutOfBoundsException</i>	if the size is negative.

6.93.2.3 decaf::internal::util::ByteArrayAdapter::ByteArrayAdapter (char * array, int size, bool own = false)

Creates a byte array object that wraps the given array.

If the own flag is set then it will delete this array when this object is deleted.

Parameters

<i>array</i>	The physical array to wrap.
<i>size</i>	The size of the array, this is the limit we read and write to.
<i>own</i>	Indicates if this class is now the owner of the pointer.

Exceptions

<i>NullPointerException</i>	if buffer is NULL
<i>IndexOutOfBoundsException</i>	if the size is negative.

6.93.2.4 **decaf::internal::util::ByteArrayAdapter::ByteArrayAdapter** (*double * array*, *int size*, *bool own = false*)

Creates a byte array object that wraps the given array.

If the own flag is set then it will delete this array when this object is deleted.

Parameters

<i>array</i>	The physical array to wrap.
<i>size</i>	The size of the array, this is the limit we read and write to.
<i>own</i>	Indicates if this class is now the owner of the pointer.

Exceptions

<i>NullPointerException</i>	if buffer is NULL
<i>IndexOutOfBoundsException</i>	if the size is negative.

6.93.2.5 **decaf::internal::util::ByteArrayAdapter::ByteArrayAdapter** (*float * array*, *int size*, *bool own = false*)

Creates a byte array object that wraps the given array.

If the own flag is set then it will delete this array when this object is deleted.

Parameters

<i>array</i>	The physical array to wrap.
<i>size</i>	The size of the array, this is the limit we read and write to.
<i>own</i>	Indicates if this class is now the owner of the pointer.

Exceptions

<i>NullPointerException</i>	if buffer is NULL
<i>IndexOutOfBoundsException</i>	if the size is negative.

6.93.2.6 `decaf::internal::util::ByteArrayAdapter::ByteArrayAdapter (long long * array, int size, bool own = false)`

Creates a byte array object that wraps the given array.

If the own flag is set then it will delete this array when this object is deleted.

Parameters

<i>array</i>	The physical array to wrap.
<i>size</i>	The size of the array, this is the limit we read and write to.
<i>own</i>	Indicates if this class is now the owner of the pointer.

Exceptions

<i>NullPointerException</i>	if buffer is NULL
<i>IndexOutOfBoundsException</i>	if the size is negative.

6.93.2.7 `decaf::internal::util::ByteArrayAdapter::ByteArrayAdapter (int * array, int size, bool own = false)`

Creates a byte array object that wraps the given array.

If the own flag is set then it will delete this array when this object is deleted.

Parameters

<i>array</i>	The physical array to wrap.
<i>size</i>	The size of the array, this is the limit we read and write to.
<i>own</i>	Indicates if this class is now the owner of the pointer.

Exceptions

<i>NullPointerException</i>	if buffer is NULL
<i>IndexOutOfBoundsException</i>	if the size is negative.

6.93.2.8 `decaf::internal::util::ByteArrayAdapter::ByteArrayAdapter (short * array, int size, bool own = false)`

Creates a byte array object that wraps the given array.

If the own flag is set then it will delete this array when this object is deleted.

Parameters

<i>array</i>	The physical array to wrap.
--------------	-----------------------------

<i>size</i>	The size of the array, this is the limit we read and write to.
<i>own</i>	Indicates if this class is now the owner of the pointer.

Exceptions

<i>NullPointerException</i>	if buffer is NULL
<i>IndexOutOfBoundsException</i>	if the size is negative.

6.93.2.9 **virtual** **decaf::internal::util::ByteArrayAdapter::~~ByteArrayAdapter** ()
[virtual]

6.93.3 Member Function Documentation

6.93.3.1 **virtual void** **decaf::internal::util::ByteArrayAdapter::clear** () throw ()
[virtual]

Clear all data from that Array, setting the underlying bytes to zero.

6.93.3.2 **virtual unsigned char** **decaf::internal::util::ByteArrayAdapter::get** (int *index*)
const [virtual]

Absolute get method.

Reads the byte at the given index.

Parameters

<i>index</i>	The index in the Buffer where the byte is to be read.
--------------	---

Returns

the byte that is located at the given index.

Exceptions

<i>IndexOutOfBoundsException</i>	If index is not smaller than the buffer's limit or is negative.
----------------------------------	---

6.93.3.3 **virtual unsigned char*** **decaf::internal::util::ByteArrayAdapter::getByteArray**
() [inline, virtual]

Gets the pointer to the array we are wrapping.

Changes to the data in this array are reflected by all **ByteArrayAdapter** (p. 624) objects

that point to this array.

Returns

an unsigned char* pointer to the array this object wraps.

6.93.3.4 virtual int **decaf::internal::util::ByteArrayAdapter::getCapacity** () const
[inline, virtual]

Gets the size of the underlying array.

Returns

the size the array.

6.93.3.5 virtual char **decaf::internal::util::ByteArrayAdapter::getChar** (int *index*)
const [virtual]

Reads one byte at the given index and returns it.

Parameters

<i>index</i>	The index in the Buffer where the byte is to be read.
--------------	---

Returns

the byte that is located at the given index.

Exceptions

<i>IndexOutOfBoundsException</i>	If index is not smaller than the buffer's limit or is negative.
----------------------------------	---

6.93.3.6 virtual char* **decaf::internal::util::ByteArrayAdapter::getCharArray** ()
[inline, virtual]

Gets the pointer to the array we are wrapping.

Changes to the data in this array are reflected by all **ByteArrayAdapter** (p. 624) objects that point to this array.

Returns

an char* pointer to the array this object wraps.

6.93.3.7 `virtual int decaf::internal::util::ByteArrayAdapter::getCharCapacity ()`
`const [inline, virtual]`

Gets the size of the underlying array as if it contains chars.

Returns

the size the array.

6.93.3.8 `virtual double decaf::internal::util::ByteArrayAdapter::getDouble (int index`
`) const [virtual]`

Reads eight bytes at the given index and returns it.

The index is a relative to the size of the type to be read, in other words when accessing the element in the array `index * sizeof(type)` if the actual start index of the type to be read.

Parameters

<i>index</i>	The index in the Buffer where the bytes are to be read.
--------------	---

Returns

the value at the given index in the buffer.

Exceptions

<i>IndexOutOfBoundsException</i>	if there are not enough bytes remaining to fill the requested Data Type, or index is negative.
----------------------------------	--

6.93.3.9 `virtual double* decaf::internal::util::ByteArrayAdapter::getDoubleArray ()`
`[inline, virtual]`

Gets the pointer to the array we are wrapping.

Changes to the data in this array are reflected by all **ByteArrayAdapter** (p. 624) objects that point to this array.

Returns

an `double*` pointer to the array this object wraps.

6.93.3.10 `virtual double decaf::internal::util::ByteArrayAdapter::getDoubleAt (int`
`index) const [virtual]`

Reads eight bytes at the given byte index and returns it.

Parameters

<i>index</i>	The index in the Buffer where the bytes are to be read
--------------	--

Returns

the value at the given index in the buffer.

Exceptions

<i>IndexOutOfBoundsException</i>	if there are not enough bytes remaining to fill the requested Data Type, or index is negative.
----------------------------------	--

6.93.3.11 virtual int decaf::internal::util::ByteArrayAdapter::getDoubleCapacity ()
const [inline, virtual]

Gets the size of the underlying array as if it contains doubles.

Returns

the size the array.

6.93.3.12 virtual float decaf::internal::util::ByteArrayAdapter::getFloat (int *index*)
const [virtual]

Reads four bytes at the given index and returns it.

The index is a relative to the size of the type to be read, in other words when accessing the element in the array `index * sizeof(type)` if the actual start index of the type to be read.

Parameters

<i>index</i>	The index in the Buffer where the bytes are to be read.
--------------	---

Returns

the value at the given index in the buffer.

Exceptions

<i>IndexOutOfBoundsException</i>	if there are not enough bytes remaining to fill the requested Data Type, or index is negative.
----------------------------------	--

6.93.3.13 `virtual float* decaf::internal::util::ByteArrayAdapter::getFloatArray ()`
`[inline, virtual]`

Gets the pointer to the array we are wrapping.

Changes to the data in this array are reflected by all **ByteArrayAdapter** (p. 624) objects that point to this array.

Returns

an float* pointer to the array this object wraps.

6.93.3.14 `virtual float decaf::internal::util::ByteArrayAdapter::getFloatAt (int index)`
`const [virtual]`

Reads four bytes at the given byte index and returns it.

Parameters

<i>index</i>	The index in the Buffer where the bytes are to be read
--------------	--

Returns

the value at the given index in the buffer.

Exceptions

<i>IndexOutOfBoundsException</i>	if there are not enough bytes remaining to fill the requested Data Type, or index is negative.
----------------------------------	--

6.93.3.15 `virtual int decaf::internal::util::ByteArrayAdapter::getFloatCapacity ()`
`const [inline, virtual]`

Gets the size of the underlying array as if it contains doubles.

Returns

the size the array.

6.93.3.16 `virtual int decaf::internal::util::ByteArrayAdapter::getInt (int index) const`
`[virtual]`

Reads four bytes at the given index and returns it.

The index is a relative to the size of the type to be read, in other words when accessing the element in the array `index * sizeof(type)` if the actual start index of the type to be read.

Parameters

<i>index</i>	The index in the Buffer where the bytes are to be read.
--------------	---

Returns

the value at the given index in the buffer.

Exceptions

<i>IndexOutOfBoundsException</i>	if there are not enough bytes remaining to fill the requested Data Type, or index is negative.
----------------------------------	--

6.93.3.17 virtual int* **decaf::internal::util::ByteArrayAdapter::getIntArray** ()
[inline, virtual]

Gets the pointer to the array we are wrapping.

Changes to the data in this array are reflected by all **ByteArrayAdapter** (p. 624) objects that point to this array.

Returns

an int* pointer to the array this object wraps.

6.93.3.18 virtual int **decaf::internal::util::ByteArrayAdapter::getIntAt** (int *index*)
const [virtual]

Reads four bytes at the given byte index and returns it.

Parameters

<i>index</i>	The index in the Buffer where the bytes are to be read
--------------	--

Returns

the value at the given index in the buffer.

Exceptions

<i>IndexOutOfBoundsException</i>	if there are not enough bytes remaining to fill the requested Data Type, or index is negative.
----------------------------------	--

6.93.3.19 `virtual int decaf::internal::util::ByteArrayAdapter::getIntCapacity () const`
`[inline, virtual]`

Gets the size of the underlying array as if it contains ints.

Returns

the size the array.

6.93.3.20 `virtual long long decaf::internal::util::ByteArrayAdapter::getLong (int index`
`) const [virtual]`

Reads eight bytes at the given index and returns it.

The index is a relative to the size of the type to be read, in other words when accessing the element in the array `index * sizeof(type)` if the actual start index of the type to be read.

Parameters

<i>index</i>	The index in the Buffer where the bytes are to be read.
--------------	---

Returns

the value at the given index in the buffer.

Exceptions

<i>IndexOutOfBoundsException</i>	if there are not enough bytes remaining to fill the requested Data Type, or index is negative.
----------------------------------	--

6.93.3.21 `virtual long long* decaf::internal::util::ByteArrayAdapter::getLongArray (`
`) [inline, virtual]`

Gets the pointer to the array we are wrapping.

Changes to the data in this array are reflected by all **ByteArrayAdapter** (p. 624) objects that point to this array.

Returns

an `long long*` pointer to the array this object wraps.

6.93.3.22 `virtual long long decaf::internal::util::ByteArrayAdapter::getLongAt (int`
`index) const [virtual]`

Reads eight bytes at the given byte index and returns it.

Parameters

<i>index</i>	The index in the Buffer where the bytes are to be read
--------------	--

Returns

the value at the given index in the buffer.

Exceptions

<i>IndexOutOfBoundsException</i>	if there are not enough bytes remaining to fill the requested Data Type, or index is negative.
----------------------------------	--

6.93.3.23 virtual int decaf::internal::util::ByteArrayAdapter::getLongCapacity ()
const [inline, virtual]

Gets the size of the underlying array as if it contains doubles.

Returns

the size the array.

6.93.3.24 virtual short decaf::internal::util::ByteArrayAdapter::getShort (int *index*)
const [virtual]

Reads two bytes at the given index and returns it.

The index is a relative to the size of the type to be read, in other words when accessing the element in the array `index * sizeof(type)` if the actual start index of the type to be read.

Parameters

<i>index</i>	The index in the Buffer where the bytes are to be read.
--------------	---

Returns

the value at the given index in the buffer.

Exceptions

<i>IndexOutOfBoundsException</i>	if there are not enough bytes remaining to fill the requested Data Type, or index is negative.
----------------------------------	--

6.93.3.25 `virtual short* decaf::internal::util::ByteArrayAdapter::getShortArray ()`
`[inline, virtual]`

Gets the pointer to the array we are wrapping.

Changes to the data in this array are reflected by all **ByteArrayAdapter** (p. 624) objects that point to this array.

Returns

an short* pointer to the array this object wraps.

6.93.3.26 `virtual short decaf::internal::util::ByteArrayAdapter::getShortAt (int index) const` `[virtual]`

Reads two bytes at the given byte index and returns it.

Parameters

<i>index</i>	The index in the Buffer where the bytes are to be read
--------------	--

Returns

the value at the given index in the buffer.

Exceptions

<i>IndexOutOfBoundsException</i>	if there are not enough bytes remaining to fill the requested Data Type, or index is negative.
----------------------------------	--

6.93.3.27 `virtual int decaf::internal::util::ByteArrayAdapter::getShortCapacity ()`
`const [inline, virtual]`

Gets the size of the underlying array as if it contains shorts.

Returns

the size the array.

6.93.3.28 `unsigned char& decaf::internal::util::ByteArrayAdapter::operator[] (int index)`

Allows the **ByteArrayAdapter** (p. 624) to be indexed as a standard array.
 calling the non constant version allows the user to change the value at index

Parameters

<i>index</i>	The position in the array to access, if the value is negative or greater than the size of the underlying array an <code>IndexOutOfBoundsException</code> is thrown.
--------------	---

Exceptions

<i>IndexOutOfBoundsException</i>	if the preconditions of <i>index</i> are not met.
----------------------------------	---

6.93.3.29 `const unsigned char& decaf::internal::util::ByteArrayAdapter::operator[] (int index) const`

6.93.3.30 `virtual ByteArrayAdapter& decaf::internal::util::ByteArrayAdapter::put (int index, unsigned char value) [virtual]`

Writes the given byte into this buffer at the given index.

The index is a relative to the size of the type to be read, in other words when accessing the element in the array `index * sizeof(type)` if the actual start index of the type to be read.

Parameters

<i>index</i>	The position in the Buffer to write the data.
<i>value</i>	The value to write to the array.

Returns

a reference to this buffer.

Exceptions

<i>IndexOutOfBoundsException</i>	if <i>index</i> greater than the buffer's limit minus the size of the type being written, or <i>index</i> is negative.
----------------------------------	--

6.93.3.31 `virtual ByteArrayAdapter& decaf::internal::util::-
ByteArrayAdapter::putChar (int index, char value)
[virtual]`

Writes one byte containing the given value, into this buffer at the given index.

The index is a relative to the size of the type to be read, in other words when accessing the element in the array `index * sizeof(type)` if the actual start index of the type to be read.

Parameters

<i>index</i>	The position in the Buffer to write the data.
<i>value</i>	The value to write to the array.

Returns

a reference to this buffer.

Exceptions

<i>IndexOutOfBoundsException</i>	if index greater than the buffer's limit minus the size of the type being written, or index is negative.
----------------------------------	--

6.93.3.32 `virtual ByteArrayAdapter& decaf::internal::util::Byte-
ArrayAdapter::putDouble (int index, double value)
[virtual]`

Writes eight bytes containing the given value, into this buffer at the given index.

The index is a relative to the size of the type to be read, in other words when accessing the element in the array `index * sizeof(type)` if the actual start index of the type to be read.

Parameters

<i>index</i>	The position in the Buffer to write the data.
<i>value</i>	The value to write to the array.

Returns

a reference to this buffer.

Exceptions

<i>IndexOutOfBoundsException</i>	if index greater than the buffer's limit minus the size of the type being written, or index is negative.
----------------------------------	--

6.93.3.33 `virtual ByteArrayAdapter& decaf::internal::util::Byte-
ArrayAdapter::putDoubleAt (int index, double value)
[virtual]`

Writes eight bytes containing the given value, into this buffer at the given byte index.

Parameters

<i>index</i>	The position in the Buffer to write the data.
<i>value</i>	The value to write.

Returns

a reference to this buffer.

Exceptions

<i>IndexOutOfBoundsException</i>	if index greater than the buffer's limit minus the size of the type being written, or index is negative.
----------------------------------	--

6.93.3.34 virtual **ByteArrayAdapter& decaf::internal::util::ByteArrayAdapter::putFloat (int *index*, float *value*)**
[virtual]

Writes four bytes containing the given value, into this buffer at the given index.

The index is a relative to the size of the type to be read, in other words when accessing the element in the array $\text{index} * \text{sizeof}(\text{type})$ if the actual start index of the type to be read.

Parameters

<i>index</i>	The position in the Buffer to write the data.
<i>value</i>	The value to write to the array.

Returns

a reference to this buffer.

Exceptions

<i>IndexOutOfBoundsException</i>	if index greater than the buffer's limit minus the size of the type being written, or index is negative.
----------------------------------	--

6.93.3.35 virtual **ByteArrayAdapter& decaf::internal::util::ByteArrayAdapter::putFloatAt (int *index*, float *value*)**
[virtual]

Writes four bytes containing the given value, into this buffer at the given byte index.

Parameters

<i>index</i>	The position in the Buffer to write the data.
<i>value</i>	The value to write.

Returns

a reference to this buffer.

Exceptions

<i>IndexOutOfBoundsException</i>	if index greater than the buffer's limit minus the size of the type being written, or index is negative.
----------------------------------	--

6.93.3.36 `virtual ByteArrayAdapter& decaf::internal::util::ByteArrayAdapter::putInt (int index, int value) [virtual]`

Writes four bytes containing the given value, into this buffer at the given index.

The index is a relative to the size of the type to be read, in other words when accessing the element in the array `index * sizeof(type)` if the actual start index of the type to be read.

Parameters

<i>index</i>	The position in the Buffer to write the data.
<i>value</i>	The value to write to the array.

Returns

a reference to this buffer.

Exceptions

<i>IndexOutOfBoundsException</i>	if index greater than the buffer's limit minus the size of the type being written, or index is negative.
----------------------------------	--

6.93.3.37 `virtual ByteArrayAdapter& decaf::internal::util::ByteArrayAdapter::putIntAt (int index, int value) [virtual]`

Writes four bytes containing the given value, into this buffer at the given byte index.

Parameters

<i>index</i>	The position in the Buffer to write the data.
<i>value</i>	The value to write.

Returns

a reference to this buffer.

Exceptions

<i>IndexOutOfBoundsException</i>	if index greater than the buffer's limit minus the size of the type being written, or index is negative.
----------------------------------	--

6.93.3.38 virtual **ByteArrayAdapter&** decaf::internal::util::Byte-
ArrayAdapter::putLong (int *index*, long long *value*)
[virtual]

Writes eight bytes containing the given value, into this buffer at the given index.

The index is a relative to the size of the type to be read, in other words when accessing the element in the array `index * sizeof(type)` if the actual start index of the type to be read.

Parameters

<i>index</i>	The position in the Buffer to write the data.
<i>value</i>	The value to write to the array.

Returns

a reference to this buffer.

Exceptions

<i>IndexOutOfBoundsException</i>	if index greater than the buffer's limit minus the size of the type being written, or index is negative.
----------------------------------	--

6.93.3.39 virtual **ByteArrayAdapter&** decaf::internal::util::Byte-
ArrayAdapter::putLongAt (int *index*, long long *value*)
[virtual]

Writes eight bytes containing the given value, into this buffer at the given byte index.

Parameters

<i>index</i>	The position in the Buffer to write the data.
<i>value</i>	The value to write.

Returns

a reference to this buffer.

Exceptions

<i>IndexOutOfBoundsException</i>	if index greater than the buffer's limit minus the size of the type being written, or index is negative.
----------------------------------	--

6.93.3.40 `virtual ByteArrayAdapter& decaf::internal::util::Byte-
ArrayAdapter::putShort (int index, short value)
[virtual]`

Writes two bytes containing the given value, into this buffer at the given index.

The index is a relative to the size of the type to be read, in other words when accessing the element in the array `index * sizeof(type)` if the actual start index of the type to be read.

Parameters

<i>index</i>	The position in the Buffer to write the data.
<i>value</i>	The value to write to the array.

Returns

a reference to this buffer.

Exceptions

<i>IndexOutOfBoundsException</i>	if index greater than the buffer's limit minus the size of the type being written, or index is negative.
----------------------------------	--

6.93.3.41 `virtual ByteArrayAdapter& decaf::internal::util::Byte-
ArrayAdapter::putShortAt (int index, short value)
[virtual]`

Writes two bytes containing the given value, into this buffer at the given byte index.

Parameters

<i>index</i>	The position in the Buffer to write the data.
<i>value</i>	The value to write.

Returns

a reference to this buffer.

Exceptions

<i>IndexOutOfBoundsException</i>	if index greater than the buffer's limit minus the size of the type being written, or index is negative.
----------------------------------	--

6.93.3.42 `virtual void decaf::internal::util::ByteArrayAdapter::read (unsigned char *
buffer, int size, int offset, int length) const [virtual]`

Reads from the Byte array starting at the specified offset and reading the specified length.

If the length is greater than the size of this underlying byte array then an `BufferUnderflowException` is thrown.

Parameters

<i>buffer</i>	The buffer to read data from this array into.
<i>size</i>	The size of the buffer passed.
<i>offset</i>	The position in this array to start reading from.
<i>length</i>	The amount of data to read from this array.

Exceptions

<i>IndexOutOfBoundsException</i>	if the offset + length exceeds the size.
<i>NullPointerException</i>	if buffer is null
<i>BufferUnderflowException</i>	if there is not enough data to read because the offset or the length is greater than the size of this array.

6.93.3.43 `virtual void decaf::internal::util::ByteArrayAdapter::resize (int size)
[virtual]`

Resizes the underlying array to the new given size, preserving all the Data that was previously in the array, unless the resize is smaller than the current size in which case only the data that will fit into the new array is preserved.

A **ByteArrayAdapter** (p. 624) can only be resized when it owns the underlying array, if it does not then it will throw an `InvalidStateException`.

Parameters

<i>size</i>	The new size of the array.
-------------	----------------------------

Exceptions

<i>IllegalArgument-Exception</i>	if the size parameter is negative.
<i>InvalidStateException</i>	if this object does not own the buffer.

6.93.3.44 `virtual void decaf::internal::util::ByteArrayAdapter::write (unsigned char *
buffer, int size, int offset, int length) [virtual]`

Writes from the Byte array given, starting at the specified offset and writing the specified amount of data into this objects internal array.

. If the length is greater than the size of this underlying byte array then an BufferOverflowException is thrown.

Parameters

<i>buffer</i>	The buffer to read data from this array into.
<i>size</i>	The size of the buffer passed.
<i>offset</i>	The position in this array to start reading from.
<i>length</i>	The amount of data to read from this array.

Exceptions

<i>IndexOutOfBoundsException</i>	if the offset + length exceeds the size.
<i>NullPointerException</i>	if buffer is null
<i>BufferOverflowException</i>	if the amount of data to be written to this array or the offset given are larger than this array's size.

The documentation for this class was generated from the following file:

- src/main/decaf/internal/util/**ByteArrayAdapter.h**

6.94 decaf::internal::nio::ByteBuffer Class Reference

This class defines six categories of operations upon byte buffers:

```
#include <src/main/decaf/internal/nio/ByteBuffer.h>
```

Inheritance diagram for decaf::internal::nio::ByteBuffer:

Public Member Functions

- **ByteBuffer** (int **capacity**, bool readOnly=false)
*Creates a **ByteBuffer** (p. 646) object that has its backing array allocated internally and is then owned and deleted when this object is deleted.*
- **ByteBuffer** (unsigned char ***array**, int size, int offset, int length, bool readOnly=false)
*Creates a **ByteBuffer** (p. 646) object that wraps the given array.*
- **ByteBuffer** (const decaf::lang::Pointer< **ByteArrayAdapter** > &**array**, int offset, int length, bool readOnly=false)

Creates a byte buffer that wraps the passed ByteBufferAdapter and start at the given offset.

- **ByteBuffer** (const **ByteBuffer** &other)

*Create a **ByteBuffer** (p. 646) that mirrors this one, meaning it shares a reference to this buffers ByteBufferAdapter and when changes are made to that data it is reflected in both.*

- virtual ~**ByteBuffer** ()
- virtual bool **isReadOnly** () const

Tells whether or not this buffer is read-only.

Returns

true if, and only if, this buffer is read-only

- virtual unsigned char * **array** ()

Returns the byte array that backs this buffer.

Modifications to this buffer's content will cause the returned array's content to be modified, and vice versa.

Invoke the hasArray method before invoking this method in order to ensure that this buffer has an accessible backing array.

Returns

The array that backs this buffer

Exceptions

ReadOnlyBuffer-Exception (p. 2244)	<i>if this buffer is backed by an array but is read-only</i>
UnsupportedOperation-Exception	<i>if this buffer is not backed by an accessible array</i>

- virtual int **arrayOffset** () const

Returns the offset within this buffer's backing array of the first element of the buffer.

*If this buffer is backed by an array then buffer position p corresponds to array index p + **arrayOffset**() (p. 696).*

Invoke the hasArray method before invoking this method in order to ensure that this buffer has an accessible backing array.

Returns

The offset within this buffer's array of the first element of the buffer.

Exceptions

ReadOnlyBuffer-Exception (p. 2244)	<i>if this buffer is backed by an array but is read-only.</i>
UnsupportedOperation-Exception	<i>if this buffer is not backed by an accessible array.</i>

- virtual bool **hasArray** () const

Tells whether or not this buffer is backed by an accessible byte array.

If this method returns true then the array and arrayOffset methods may safely be invoked. Subclasses should override this method if they do not have a backing array as this class always returns true.

Returns

true if, and only if, this buffer is backed by an array and is not read-only.

- virtual **decaf::nio::CharBuffer * asCharBuffer ()** const

Creates a view of this byte buffer as a char buffer.

The content of the new buffer will start at this buffer's current position. Changes to this buffer's content will be visible in the new buffer, and vice versa; the two buffers' position, limit, and mark values will be independent.

The new buffer's position will be zero, its capacity and its limit will be the number of bytes remaining in this buffer, and its mark will be undefined. The new buffer will be read-only if, and only if, this buffer is read-only.

Returns

*the new Char **Buffer** (p. 582), which the caller then owns.*

- virtual **decaf::nio::DoubleBuffer * asDoubleBuffer ()** const

Creates a view of this byte buffer as a double buffer.

The content of the new buffer will start at this buffer's current position. Changes to this buffer's content will be visible in the new buffer, and vice versa; the two buffers' position, limit, and mark values will be independent.

The new buffer's position will be zero, its capacity and its limit will be the number of bytes remaining in this buffer divided by eight, and its mark will be undefined. The new buffer will be read-only if, and only if, this buffer is read-only.

Returns

*the new double **Buffer** (p. 582), which the caller then owns.*

- virtual **decaf::nio::FloatBuffer * asFloatBuffer ()** const

Creates a view of this byte buffer as a float buffer.

The content of the new buffer will start at this buffer's current position. Changes to this buffer's content will be visible in the new buffer, and vice versa; the two buffers' position, limit, and mark values will be independent.

The new buffer's position will be zero, its capacity and its limit will be the number of bytes remaining in this buffer divided by four, and its mark will be undefined. The new buffer will be read-only if, and only if, this buffer is read-only.

Returns

*the new float **Buffer** (p. 582), which the caller then owns.*

- virtual **decaf::nio::IntBuffer * asIntBuffer ()** const

Creates a view of this byte buffer as a int buffer.

The content of the new buffer will start at this buffer's current position. Changes to this buffer's content will be visible in the new buffer, and vice versa; the two buffers' position, limit, and mark values will be independent.

The new buffer's position will be zero, its capacity and its limit will be the number of bytes remaining in this buffer divided by four, and its mark will be undefined. The new buffer will be read-only if, and only if, this buffer is read-only.

Returns

*the new int **Buffer** (p. 582), which the caller then owns.*

- virtual **decaf::nio::LongBuffer * asLongBuffer ()** const

Creates a view of this byte buffer as a long buffer.

The content of the new buffer will start at this buffer's current position. Changes to this buffer's content will be visible in the new buffer, and vice versa; the two buffers' position, limit, and mark values will be independent.

The new buffer's position will be zero, its capacity and its limit will be the number of bytes remaining in this buffer divided by eight, and its mark will be undefined. The new buffer will be read-only if, and only if, this buffer is read-only.

Returns

the new long **Buffer** (p. 582), which the caller then owns.

- virtual **decaf::nio::ShortBuffer * asShortBuffer ()** const

Creates a view of this byte buffer as a short buffer.

The content of the new buffer will start at this buffer's current position. Changes to this buffer's content will be visible in the new buffer, and vice versa; the two buffers' position, limit, and mark values will be independent.

The new buffer's position will be zero, its capacity and its limit will be the number of bytes remaining in this buffer divided by two, and its mark will be undefined. The new buffer will be read-only if, and only if, this buffer is read-only.

Returns

the new short **Buffer** (p. 582), which the caller then owns.

- virtual **ByteBuffer * asReadOnlyBuffer ()** const

Creates a new, read-only byte buffer that shares this buffer's content.

The content of the new buffer will be that of this buffer. Changes to this buffer's content will be visible in the new buffer; the new buffer itself, however, will be read-only and will not allow the shared content to be modified. The two buffers' position, limit, and mark values will be independent.

If this buffer is itself read-only then this method behaves in exactly the same way as the duplicate method.

The new buffer's capacity, limit, position, and mark values will be identical to those of this buffer.

Returns

The new, read-only byte buffer which the caller then owns.

- virtual **ByteBuffer & compact ()**

Compacts this buffer.

The bytes between the buffer's current position and its limit, if any, are copied to the beginning of the buffer. That is, the byte at index $p = \text{position}()$ (p. 587) is copied to index zero, the byte at index $p + 1$ is copied to index one, and so forth until the byte at index $\text{limit}()$ (p. 586) - 1 is copied to index $n = \text{limit}()$ (p. 586) - 1 - p . The buffer's position is then set to $n + 1$ and its limit is set to its capacity. The mark, if defined, is discarded.

The buffer's position is set to the number of bytes copied, rather than to zero, so that an invocation of this method can be followed immediately by an invocation of another relative put method.

Returns

a reference to this **ByteBuffer** (p. 690).

Exceptions

ReadOnlyBufferException (p. 2244)	if this buffer is read-only.
---	------------------------------

- virtual **ByteBuffer * duplicate ()**

Creates a new byte buffer that shares this buffer's content.

The content of the new buffer will be that of this buffer. Changes to this buffer's content will be visible in the new buffer, and vice versa; the two buffers' position, limit, and mark values will be independent.

The new buffer's capacity, limit, position, and mark values will be identical to those of this buffer. The new buffer will be read-only if, and only if, this buffer is read-only.

Returns

a new Byte **Buffer** (p. 582) which the caller owns.

- virtual unsigned char **get** () const

Relative get method.

Reads the byte at this buffer's current position, and then increments the position.

Returns

The byte at the buffer's current position.

Exceptions

BufferUnderflow-Exception (p. 611)	<i>if the buffer's current position is not smaller than its limit.</i>
--	--

- virtual unsigned char **get** (int index) const

Absolute get method.

Reads the byte at the given index.

Parameters

index	<i>The index in the Buffer (p. 582) where the byte is to be read.</i>
-------	--

Returns

the byte that is located at the given index.

Exceptions

IndexOutOfBoundsException	<i>if index is not smaller than the buffer's limit, or index is negative.</i>
---------------------------	---

- virtual char **getChar** ()

Reads the next byte at this buffer's current position, and then increments the position by one.

Returns

the next char in the buffer.

Exceptions

BufferUnderflow-Exception (p. 611)	<i>if there are no more bytes remaining in this buffer, meaning we have reached the set limit.</i>
--	--

- virtual char **getChar** (int index) const

Reads one byte at the given index and returns it.

Parameters

index	<i>The index in the Buffer (p. 582) where the byte is to be read.</i>
-------	--

Returns

the char at the given index in the buffer

Exceptions

IndexOutOfBoundsException	<i>if index is not smaller than the buffer's limit, or index is negative.</i>
---------------------------	---

- virtual double **getDouble** ()

Reads the next eight bytes at this buffer's current position, and then increments the

position by that amount.

Returns

the next double in the buffer.

Exceptions

BufferUnderflow-Exception (p. 611)	<i>if there are no more bytes remaining in this buffer, meaning we have reached the set limit.</i>
--	--

- virtual double **getDouble** (int index) const

Reads eight bytes at the given index and returns it.

Parameters

index	<i>The index in the Buffer (p. 582) where the bytes are to be read.</i>
-------	--

Returns

the double at the given index in the buffer.

Exceptions

IndexOutOfBoundsException	<i>if index is not smaller than the buffer's limit, or index is negative.</i>
---------------------------	---

- virtual float **getFloat** ()

Reads the next four bytes at this buffer's current position, and then increments the position by that amount.

Returns

the next float in the buffer.

Exceptions

BufferUnderflow-Exception (p. 611)	<i>if there are no more bytes remaining in this buffer, meaning we have reached the set limit.</i>
--	--

- virtual float **getFloat** (int index) const

Reads four bytes at the given index and returns it.

Parameters

index	<i>The index in the Buffer (p. 582) where the bytes are to be read.</i>
-------	--

Returns

the float at the given index in the buffer.

Exceptions

IndexOutOfBoundsException	<i>if there are not enough bytes remaining to fill the requested Data Type, or index is negative.</i>
---------------------------	---

- virtual long long **getLong** ()

Reads the next eight bytes at this buffer's current position, and then increments the position by that amount.

Returns

the next long long in the buffer.

Exceptions

BufferUnderflow-Exception (p. 611)	<i>if there are no more bytes remaining in this buffer, meaning we have reached the set limit.</i>
--	--

- virtual long long **getLong** (int index) const

Reads eight bytes at the given index and returns it.

Parameters

index	<i>The index in the Buffer (p. 582) where the bytes are to be read.</i>
-------	--

Returns

the long long at the given index in the buffer.

Exceptions

IndexOutOfBounds-Exception	<i>if there are not enough bytes remaining to fill the requested Data Type, or index is negative.</i>
----------------------------	---

- virtual int **getInt** ()

Reads the next four bytes at this buffer's current position, and then increments the position by that amount.

Returns

the next int in the buffer.

Exceptions

BufferUnderflow-Exception (p. 611)	<i>if there are no more bytes remaining in this buffer, meaning we have reached the set limit.</i>
--	--

- virtual int **getInt** (int index) const

Reads four bytes at the given index and returns it.

Parameters

index	<i>The index in the Buffer (p. 582) where the bytes are to be read.</i>
-------	--

Returns

the int at the given index in the buffer.

Exceptions

IndexOutOfBounds-Exception	<i>if there are not enough bytes remaining to fill the requested Data Type, or index is negative.</i>
----------------------------	---

- virtual short **getShort** ()

Reads the next two bytes at this buffer's current position, and then increments the position by that amount.

Returns

the next short in the buffer.

Exceptions

BufferUnderflow-Exception (p. 611)	<i>if there are no more bytes remaining in this buffer, meaning we have reached the set limit.</i>
--	--

- virtual short **getShort** (int index) const

Reads two bytes at the given index and returns it.

Parameters

index	<i>The index in the Buffer (p. 582) where the bytes are to be read.</i>
-------	--

Returns

the short at the given index in the buffer.

Exceptions

IndexOutOfBoundsException	<i>if there are not enough bytes remaining to fill the requested Data Type, or index is negative.</i>
---------------------------	---

- virtual **ByteBuffer** & **put** (unsigned char value)

Writes the given byte into this buffer at the current position, and then increments the position.

Parameters

value	<i>- the byte value to be written.</i>
-------	--

Returns

a reference to this buffer.

Exceptions

BufferOverflowException (p. 609)	<i>if this buffer's current position is not smaller than its limit.</i>
ReadOnlyBufferException (p. 2244)	<i>if this buffer is read-only.</i>

- virtual **ByteBuffer** & **put** (int index, unsigned char value)

Writes the given byte into this buffer at the given index.

Parameters

index	<i>- position in the Buffer (p. 582) to write the data</i>
value	<i>- the byte to write.</i>

Returns

a reference to this buffer.

Exceptions

IndexOutOfBoundsException	<i>if index greater than the buffer's limit minus the size of the type being written, or index is negative.</i>
ReadOnlyBufferException (p. 2244)	<i>if this buffer is read-only.</i>

- virtual **ByteBuffer** & **putChar** (char value)

Writes one byte containing the given value, into this buffer at the current position, and then increments the position by one.

Parameters

value	<i>The value to be written.</i>
-------	---------------------------------

Returns

a reference to this buffer.

Exceptions

BufferOverflow-Exception (p. 609)	<i>if there are fewer than bytes remaining in this buffer than the size of the data to be written</i>
ReadOnlyBuffer-Exception (p. 2244)	<i>if this buffer is read-only</i>

- virtual **ByteBuffer & putChar** (int index, char value)

Writes one byte containing the given value, into this buffer at the given index.

Parameters

index	<i>The position in the Buffer (p. 582) to write the data.</i>
value	<i>The value to write.</i>

Returns

a reference to this buffer.

Exceptions

IndexOutOfBoundsException	<i>if index greater than the buffer's limit minus the size of the type being written, or index is negative.</i>
ReadOnlyBuffer-Exception (p. 2244)	<i>if this buffer is read-only</i>

- virtual **ByteBuffer & putDouble** (double value)

Writes eight bytes containing the given value, into this buffer at the current position, and then increments the position by eight.

Parameters

value	<i>The value to be written.</i>
-------	---------------------------------

Returns

a reference to this buffer.

Exceptions

BufferOverflow-Exception (p. 609)	<i>if there are fewer than bytes remaining in this buffer than the size of the data to be written</i>
ReadOnlyBuffer-Exception (p. 2244)	<i>if this buffer is read-only</i>

- virtual **ByteBuffer & putDouble** (int index, double value)

Writes eight bytes containing the given value, into this buffer at the given index.

Parameters

index	<i>The position in the Buffer (p. 582) to write the data</i>
value	<i>The value to write.</i>

Returns

a reference to this buffer.

Exceptions

IndexOutOfBoundsException	<i>if index greater than the buffer's limit minus the size of the type being written, or index is negative.</i>
ReadOnlyBuffer-Exception (p. 2244)	<i>if this buffer is read-only.</i>

- virtual **ByteBuffer** & **putFloat** (float value)

Writes four bytes containing the given value, into this buffer at the current position, and then increments the position by eight.

Parameters

value	<i>The value to be written.</i>
-------	---------------------------------

Returns

a reference to this buffer.

Exceptions

BufferOverflow-Exception (p. 609)	<i>if there are fewer than bytes remaining in this buffer than the size of the data to be written.</i>
ReadOnlyBuffer-Exception (p. 2244)	<i>if this buffer is read-only.</i>

- virtual **ByteBuffer** & **putFloat** (int index, float value)

Writes four bytes containing the given value, into this buffer at the given index.

Parameters

index	<i>The position in the Buffer (p. 582) to write the data</i>
value	<i>The value to write.</i>

Returns

a reference to this buffer.

Exceptions

IndexOutOfBoundsException	<i>if index greater than the buffer's limit minus the size of the type being written, or index is negative.</i>
ReadOnlyBuffer-Exception (p. 2244)	<i>if this buffer is read-only.</i>

- virtual **ByteBuffer** & **putLong** (long long value)

Writes eight bytes containing the given value, into this buffer at the current position, and then increments the position by eight.

Parameters

value	<i>The value to be written.</i>
-------	---------------------------------

Returns

a reference to this buffer.

Exceptions

BufferOverflow-Exception (p. 609)	<i>if there are fewer than bytes remaining in this buffer than the size of the data to be written.</i>
ReadOnlyBuffer-Exception (p. 2244)	<i>if this buffer is read-only.</i>

- virtual **ByteBuffer & putLong** (int index, long long value)

Writes eight bytes containing the given value, into this buffer at the given index.

Parameters

index	<i>The position in the Buffer (p. 582) to write the data.</i>
value	<i>The value to write.</i>

Returns

a reference to this buffer.

Exceptions

IndexOutOfBoundsException	<i>if index greater than the buffer's limit minus the size of the type being written, or index is negative.</i>
ReadOnlyBuffer-Exception (p. 2244)	<i>if this buffer is read-only.</i>

- virtual **ByteBuffer & putInt** (int value)

Writes four bytes containing the given value, into this buffer at the current position, and then increments the position by eight.

Parameters

value	<i>The value to be written.</i>
-------	---------------------------------

Returns

a reference to this buffer.

Exceptions

BufferOverflow-Exception (p. 609)	<i>if there are fewer than bytes remaining in this buffer than the size of the data to be written</i>
ReadOnlyBuffer-Exception (p. 2244)	<i>if this buffer is read-only</i>

- virtual **ByteBuffer & putInt** (int index, int value)

Writes four bytes containing the given value, into this buffer at the given index.

Parameters

index	<i>The position in the Buffer (p. 582) to write the data.</i>
value	<i>The value to write.</i>

Returns

a reference to this buffer

Exceptions

IndexOutOfBoundsException	<i>if index greater than the buffer's limit minus the size of the type being written, or index is negative.</i>
ReadOnlyBuffer-Exception (p. 2244)	<i>if this buffer is read-only</i>

- virtual **ByteBuffer** & **putShort** (short value)

Writes two bytes containing the given value, into this buffer at the current position, and then increments the position by eight.

Parameters

value	<i>The value to be written.</i>
-------	---------------------------------

Returns

a reference to this buffer.

Exceptions

BufferOverflow-Exception (p. 609)	<i>if there are fewer than bytes remaining in this buffer than the size of the data to be written.</i>
ReadOnlyBuffer-Exception (p. 2244)	<i>if this buffer is read-only.</i>

- virtual **ByteBuffer** & **putShort** (int index, short value)

Writes two bytes containing the given value, into this buffer at the given index.

Parameters

index	<i>The position in the Buffer (p. 582) to write the data</i>
value	<i>The value to write.</i>

Returns

a reference to this buffer

Exceptions

IndexOutOfBoundsException	<i>if index greater than the buffer's limit minus the size of the type being written, or index is negative.</i>
ReadOnlyBuffer-Exception (p. 2244)	<i>if this buffer is read-only.</i>

- virtual **ByteBuffer** * **slice** () const

Creates a new byte buffer whose content is a shared subsequence of this buffer's content.

The content of the new buffer will start at this buffer's current position. Changes to this buffer's content will be visible in the new buffer, and vice versa; the two buffers'

position, limit, and mark values will be independent.

The new buffer's position will be zero, its capacity and its limit will be the number of bytes remaining in this buffer, and its mark will be undefined. The new buffer will be read-only if, and only if, this buffer is read-only.

Returns

*the newly create **ByteBuffer** (p. 690) which the caller owns.*

Protected Member Functions

- virtual void **setReadOnly** (bool value)

*Sets this **ByteBuffer** (p. 646) as Read-Only or not Read-Only.*

6.94.1 Detailed Description

This class defines six categories of operations upon byte buffers:

1. Absolute and relative get and put methods that read and write single bytes; 2. - Relative bulk get methods that transfer contiguous sequences of bytes from this buffer into an array; 3. Relative bulk put methods that transfer contiguous sequences of bytes from a byte array or some other byte buffer into this buffer; 4. Absolute and relative get and put methods that read and write values of other primitive types, translating them to and from sequences of bytes in a particular byte order; 5. Methods for creating view buffers, which allow a byte buffer to be viewed as a buffer containing values of some other primitive type; and 6. Methods for compacting, duplicating, and slicing a byte buffer.

Byte buffers can be created either by allocation, which allocates space for the buffer's content, or by wrapping an existing byte array into a buffer.

Access to binary data:

This class defines methods for reading and writing values of all other primitive types, except boolean. Primitive values are translated to (or from) sequences of bytes according to the buffer's current byte order.

For access to heterogeneous binary data, that is, sequences of values of different types, this class defines a family of absolute and relative get and put methods for each type. For 32-bit floating-point values, for example, this class defines:

float **getFloat()** (p. 668) float **getFloat(int index)** void **putFloat(float f)** (p. 675) void **putFloat(int index, float f)** (p. 675)

Corresponding methods are defined for the types char, short, int, long, and double. The index parameters of the absolute get and put methods are in terms of bytes rather than of the type being read or written.

For access to homogeneous binary data, that is, sequences of values of the same type, this class defines methods that can create views of a given byte buffer. A view buffer is simply another buffer whose content is backed by the byte buffer. Changes to the byte buffer's content will be visible in the view buffer, and vice versa; the two buffers' position, limit, and mark values are independent. The **asFloatBuffer** method, for example, creates an instance of the **FloatBuffer** class that is backed by the byte buffer

upon which the method is invoked. Corresponding view-creation methods are defined for the types `char`, `short`, `int`, `long`, and `double`.

View buffers have two important advantages over the families of type-specific `get` and `put` methods described above:

A view buffer is indexed not in terms of bytes but rather in terms of the type-specific size of its values;

A view buffer provides relative bulk `get` and `put` methods that can transfer contiguous sequences of values between a buffer and an array or some other buffer of the same type; and

Since

1.0

6.94.2 Constructor & Destructor Documentation

6.94.2.1 `decaf::internal::nio::ByteBuffer::ByteBuffer (int capacity, bool readOnly = false)`

Creates a **ByteBuffer** (p. 646) object that has its backing array allocated internally and is then owned and deleted when this object is deleted.

The array is initially created with all elements initialized to zero.

Parameters

<i>capacity</i>	The size of the array, this is the limit we read and write to.
<i>readOnly</i>	Should this buffer be read-only, default as <code>false</code>

Exceptions

<i>IllegalArgument-Exception</i>	if the capacity value is negative.
----------------------------------	------------------------------------

6.94.2.2 `decaf::internal::nio::ByteBuffer::ByteBuffer (unsigned char * array, int size, int offset, int length, bool readOnly = false)`

Creates a **ByteBuffer** (p. 646) object that wraps the given array.

Parameters

<i>array</i>	The array to wrap.
<i>size</i>	The size of the array passed.
<i>offset</i>	The position that is this buffers start position.
<i>length</i>	The size of the sub-array, this is the limit we read and write to.
<i>readOnly</i>	Should this buffer be read-only, default as <code>false</code> .

Exceptions

<i>NullPointerException</i>	if buffer is NULL
<i>IndexOutOfBoundsException</i>	if the preconditions of size, offset and length are violated.

6.94.2.3 `decaf::internal::nio::ByteBuffer::ByteBuffer (const decaf::lang::Pointer< ByteArrayAdapter > & array, int offset, int length, bool readOnly = false)`

Creates a byte buffer that wraps the passed `ByteArrayAdapter` and start at the given offset.

The capacity and limit of the new **ByteBuffer** (p. 646) will be that of the remaining capacity of the passed buffer.

Parameters

<i>array</i>	The <code>ByteArrayAdapter</code> to wrap
<i>offset</i>	The offset into array where the buffer starts
<i>length</i>	The length of the array we are wrapping or limit.
<i>readOnly</i>	Boolean indicating if this a readOnly buffer.

Exceptions

<i>NullPointerException</i>	if array is NULL
<i>IndexOutOfBoundsException</i>	if offset is greater than array capacity.

6.94.2.4 `decaf::internal::nio::ByteBuffer::ByteBuffer (const ByteBuffer & other)`

Create a **ByteBuffer** (p. 646) that mirrors this one, meaning it shares a reference to this buffers `ByteArrayAdapter` and when changes are made to that data it is reflected in both.

Parameters

<i>other</i>	The ByteBuffer (p. 646) this one is to mirror.
--------------	---

6.94.2.5 `virtual decaf::internal::nio::ByteBuffer::~~ByteBuffer ()`
[virtual]

6.94.3 Member Function Documentation

6.94.3.1 `virtual unsigned char* decaf::internal::nio::ByteBuffer::array ()`
`[virtual]`

Returns the byte array that backs this buffer.

Modifications to this buffer's content will cause the returned array's content to be modified, and vice versa.

Invoke the `hasArray` method before invoking this method in order to ensure that this buffer has an accessible backing array.

Returns

The array that backs this buffer

Exceptions

<i>ReadOnlyBufferException</i> (p. 2244)	if this buffer is backed by an array but is read-only
<i>UnsupportedOperationException</i>	if this buffer is not backed by an accessible array

Implements **decaf::nio::ByteBuffer** (p. 695).

6.94.3.2 `virtual int decaf::internal::nio::ByteBuffer::arrayOffset () const`
`[virtual]`

Returns the offset within this buffer's backing array of the first element of the buffer.

If this buffer is backed by an array then buffer position `p` corresponds to array index `p + arrayOffset()` (p. 696).

Invoke the `hasArray` method before invoking this method in order to ensure that this buffer has an accessible backing array.

Returns

The offset within this buffer's array of the first element of the buffer.

Exceptions

<i>ReadOnlyBufferException</i> (p. 2244)	if this buffer is backed by an array but is read-only.
<i>UnsupportedOperationException</i>	if this buffer is not backed by an accessible array.

Implements **decaf::nio::ByteBuffer** (p. 696).

```
6.94.3.3 virtual decaf::nio::CharBuffer* decaf::internal::nio::-
        ByteArrayBuffer::asCharBuffer ( ) const [inline,
        virtual]
```

Creates a view of this byte buffer as a char buffer.

The content of the new buffer will start at this buffer's current position. Changes to this buffer's content will be visible in the new buffer, and vice versa; the two buffers' position, limit, and mark values will be independent.

The new buffer's position will be zero, its capacity and its limit will be the number of bytes remaining in this buffer, and its mark will be undefined. The new buffer will be read-only if, and only if, this buffer is read-only.

Returns

the new Char **Buffer** (p. 582), which the caller then owns.

Implements **decaf::nio::ByteBuffer** (p. 696).

```
6.94.3.4 virtual decaf::nio::DoubleBuffer* decaf::internal::nio::-
        ByteArrayBuffer::asDoubleBuffer ( ) const [inline,
        virtual]
```

Creates a view of this byte buffer as a double buffer.

The content of the new buffer will start at this buffer's current position. Changes to this buffer's content will be visible in the new buffer, and vice versa; the two buffers' position, limit, and mark values will be independent.

The new buffer's position will be zero, its capacity and its limit will be the number of bytes remaining in this buffer divided by eight, and its mark will be undefined. The new buffer will be read-only if, and only if, this buffer is read-only.

Returns

the new double **Buffer** (p. 582), which the caller then owns.

Implements **decaf::nio::ByteBuffer** (p. 697).

```
6.94.3.5 virtual decaf::nio::FloatBuffer* decaf::internal::nio::-
        ByteArrayBuffer::asFloatBuffer ( ) const [inline,
        virtual]
```

Creates a view of this byte buffer as a float buffer.

The content of the new buffer will start at this buffer's current position. Changes to this buffer's content will be visible in the new buffer, and vice versa; the two buffers' position, limit, and mark values will be independent.

The new buffer's position will be zero, its capacity and its limit will be the number of bytes remaining in this buffer divided by four, and its mark will be undefined. The new buffer will be read-only if, and only if, this buffer is read-only.

Returns

the new float **Buffer** (p. 582), which the caller then owns.

Implements **decaf::nio::ByteBuffer** (p. 697).

6.94.3.6 `virtual decaf::nio::IntBuffer* decaf::internal::nio::-
ByteBuffer::asIntBuffer () const [inline,
virtual]`

Creates a view of this byte buffer as a int buffer.

The content of the new buffer will start at this buffer's current position. Changes to this buffer's content will be visible in the new buffer, and vice versa; the two buffers' position, limit, and mark values will be independent.

The new buffer's position will be zero, its capacity and its limit will be the number of bytes remaining in this buffer divided by four, and its mark will be undefined. The new buffer will be read-only if, and only if, this buffer is read-only.

Returns

the new int **Buffer** (p. 582), which the caller then owns.

Implements **decaf::nio::ByteBuffer** (p. 697).

6.94.3.7 `virtual decaf::nio::LongBuffer* decaf::internal::nio::-
ByteBuffer::asLongBuffer () const [inline,
virtual]`

Creates a view of this byte buffer as a long buffer.

The content of the new buffer will start at this buffer's current position. Changes to this buffer's content will be visible in the new buffer, and vice versa; the two buffers' position, limit, and mark values will be independent.

The new buffer's position will be zero, its capacity and its limit will be the number of bytes remaining in this buffer divided by eight, and its mark will be undefined. The new buffer will be read-only if, and only if, this buffer is read-only.

Returns

the new long **Buffer** (p. 582), which the caller then owns.

Implements **decaf::nio::ByteBuffer** (p. 698).

6.94.3.8 `virtual ByteBuffer* decaf::internal::nio::ByteBuffer::asRead-
OnlyBuffer () const [virtual]`

Creates a new, read-only byte buffer that shares this buffer's content.

The content of the new buffer will be that of this buffer. Changes to this buffer's content will be visible in the new buffer; the new buffer itself, however, will be read-only and will not allow the shared content to be modified. The two buffers' position, limit, and mark values will be independent.

If this buffer is itself read-only then this method behaves in exactly the same way as the `duplicate` method.

The new buffer's capacity, limit, position, and mark values will be identical to those of this buffer.

Returns

The new, read-only byte buffer which the caller then owns.

Implements **decaf::nio::ByteBuffer** (p. 698).

```
6.94.3.9 virtual decaf::nio::ShortBuffer* decaf::internal::nio::-
        ByteArrayBuffer::asShortBuffer ( ) const [inline,
        virtual]
```

Creates a view of this byte buffer as a short buffer.

The content of the new buffer will start at this buffer's current position. Changes to this buffer's content will be visible in the new buffer, and vice versa; the two buffers' position, limit, and mark values will be independent.

The new buffer's position will be zero, its capacity and its limit will be the number of bytes remaining in this buffer divided by two, and its mark will be undefined. The new buffer will be read-only if, and only if, this buffer is read-only.

Returns

the new short **Buffer** (p. 582), which the caller then owns.

Implements **decaf::nio::ByteBuffer** (p. 698).

```
6.94.3.10 virtual ByteArrayBuffer& decaf::internal::nio::ByteBuffer::compact (
        ) [virtual]
```

Compacts this buffer.

The bytes between the buffer's current position and its limit, if any, are copied to the beginning of the buffer. That is, the byte at index $p = \text{position}()$ (p. 587) is copied to index zero, the byte at index $p + 1$ is copied to index one, and so forth until the byte at index $\text{limit}()$ (p. 586) - 1 is copied to index $n = \text{limit}()$ (p. 586) - 1 - p . The buffer's position is then set to $n+1$ and its limit is set to its capacity. The mark, if defined, is discarded.

The buffer's position is set to the number of bytes copied, rather than to zero, so that an invocation of this method can be followed immediately by an invocation of another relative put method.

Returns

a reference to this **ByteBuffer** (p. 690).

Exceptions

<i>ReadOnlyBufferException</i> (p. 2244)	if this buffer is read-only.
--	------------------------------

Implements **decaf::nio::ByteBuffer** (p. 699).

6.94.3.11 virtual **ByteBuffer*** **decaf::internal::nio::ByteBuffer::duplicate**
() [virtual]

Creates a new byte buffer that shares this buffer's content.

The content of the new buffer will be that of this buffer. Changes to this buffer's content will be visible in the new buffer, and vice versa; the two buffers' position, limit, and mark values will be independent.

The new buffer's capacity, limit, position, and mark values will be identical to those of this buffer. The new buffer will be read-only if, and only if, this buffer is read-only.

Returns

a new Byte **Buffer** (p. 582) which the caller owns.

Implements **decaf::nio::ByteBuffer** (p. 699).

6.94.3.12 virtual unsigned char **decaf::internal::nio::ByteBuffer::get**() const
[virtual]

Relative get method.

Reads the byte at this buffer's current position, and then increments the position.

Returns

The byte at the buffer's current position.

Exceptions

<i>BufferUnderflowException</i> (p. 611)	if the buffer's current position is not smaller than its limit.
--	---

Implements **decaf::nio::ByteBuffer** (p. 701).

6.94.3.13 virtual unsigned char **decaf::internal::nio::ByteArrayBuffer::get** (int *index*)
const [virtual]

Absolute get method.

Reads the byte at the given index.

Parameters

<i>index</i>	The index in the Buffer (p. 582) where the byte is to be read.
--------------	---

Returns

the byte that is located at the given index.

Exceptions

<i>IndexOutOfBoundsException</i>	if index is not smaller than the buffer's limit, or index is negative.
----------------------------------	--

Implements **decaf::nio::ByteBuffer** (p. 701).

6.94.3.14 virtual char **decaf::internal::nio::ByteArrayBuffer::getChar** ()
[inline, virtual]

Reads the next byte at this buffer's current position, and then increments the position by one.

Returns

the next char in the buffer.

Exceptions

BufferUnderflow-Exception (p. 611)	if there are no more bytes remaining in this buffer, meaning we have reached the set limit.
--	---

Implements **decaf::nio::ByteBuffer** (p. 702).

6.94.3.15 virtual char **decaf::internal::nio::ByteArrayBuffer::getChar** (int *index*) const
[inline, virtual]

Reads one byte at the given index and returns it.

Parameters

<i>index</i>	The index in the Buffer (p. 582) where the byte is to be read.
--------------	---

Returns

the char at the given index in the buffer

Exceptions

<i>IndexOutOfBoundsException</i>	if index is not smaller than the buffer's limit, or index is negative.
----------------------------------	--

Implements **decaf::nio::ByteBuffer** (p. 702).

6.94.3.16 virtual double **decaf::internal::nio::ByteBuffer::getDouble** ()
[virtual]

Reads the next eight bytes at this buffer's current position, and then increments the position by that amount.

Returns

the next double in the buffer.

Exceptions

<i>BufferUnderflowException</i> (p. 611)	if there are no more bytes remaining in this buffer, meaning we have reached the set limit.
---	---

Implements **decaf::nio::ByteBuffer** (p. 703).

6.94.3.17 virtual double **decaf::internal::nio::ByteBuffer::getDouble** (int *index*)
const [virtual]

Reads eight bytes at the given index and returns it.

Parameters

<i>index</i>	The index in the Buffer (p. 582) where the bytes are to be read.
--------------	---

Returns

the double at the given index in the buffer.

Exceptions

<i>IndexOutOfBoundsException</i>	if index is not smaller than the buffer's limit, or index is negative.
----------------------------------	--

Implements **decaf::nio::ByteBuffer** (p. 703).

6.94.3.18 virtual float **decaf::internal::nio::ByteBuffer::getFloat** ()
[virtual]

Reads the next four bytes at this buffer's current position, and then increments the position by that amount.

Returns

the next float in the buffer.

Exceptions

<i>BufferUnderflowException</i> (p. 611)	if there are no more bytes remaining in this buffer, meaning we have reached the set limit.
--	---

Implements **decaf::nio::ByteBuffer** (p. 703).

6.94.3.19 virtual float **decaf::internal::nio::ByteBuffer::getFloat** (int *index*)
const [virtual]

Reads four bytes at the given index and returns it.

Parameters

<i>index</i>	The index in the Buffer (p. 582) where the bytes are to be read.
--------------	---

Returns

the float at the given index in the buffer.

Exceptions

<i>IndexOutOfBoundsException</i>	if there are not enough bytes remaining to fill the requested Data Type, or index is negative.
----------------------------------	--

Implements **decaf::nio::ByteBuffer** (p. 704).

6.94.3.20 virtual int **decaf::internal::nio::ByteBuffer::getInt** () [virtual]

Reads the next four bytes at this buffer's current position, and then increments the position by that amount.

Returns

the next int in the buffer.

Exceptions

<i>BufferUnderflow-Exception</i> (p. 611)	if there are no more bytes remaining in this buffer, meaning we have reached the set limit.
---	---

Implements **decaf::nio::ByteBuffer** (p. 704).

6.94.3.21 virtual int **decaf::internal::nio::ByteBuffer::getInt** (int *index*) const
[virtual]

Reads four bytes at the given index and returns it.

Parameters

<i>index</i>	The index in the Buffer (p. 582) where the bytes are to be read.
--------------	---

Returns

the int at the given index in the buffer.

Exceptions

<i>IndexOutOfBounds-Exception</i>	if there are not enough bytes remaining to fill the requested Data Type, or index is negative.
--	--

Implements **decaf::nio::ByteBuffer** (p. 705).

6.94.3.22 virtual long long **decaf::internal::nio::ByteBuffer::getLong** ()
[virtual]

Reads the next eight bytes at this buffer's current position, and then increments the position by that amount.

Returns

the next long long in the buffer.

Exceptions

<i>BufferUnderflow-Exception</i> (p. 611)	if there are no more bytes remaining in this buffer, meaning we have reached the set limit.
---	---

Implements **decaf::nio::ByteBuffer** (p. 705).

6.94.3.23 virtual long long decaf::internal::nio::ByteBuffer::getLong (int *index*)
const [virtual]

Reads eight bytes at the given index and returns it.

Parameters

<i>index</i>	The index in the Buffer (p. 582) where the bytes are to be read.
--------------	---

Returns

the long long at the given index in the buffer.

Exceptions

<i>IndexOutOfBoundsException</i>	if there are not enough bytes remaining to fill the requested Data Type, or index is negative.
---	--

Implements **decaf::nio::ByteBuffer** (p. 705).

6.94.3.24 virtual short decaf::internal::nio::ByteBuffer::getShort ()
[virtual]

Reads the next two bytes at this buffer's current position, and then increments the position by that amount.

Returns

the next short in the buffer.

Exceptions

<i>BufferUnderflow-Exception</i> (p. 611)	if there are no more bytes remaining in this buffer, meaning we have reached the set limit.
---	---

Implements **decaf::nio::ByteBuffer** (p. 706).

6.94.3.25 `virtual short decaf::internal::nio::ByteBuffer::getShort (int index)
const [virtual]`

Reads two bytes at the given index and returns it.

Parameters

<i>index</i>	The index in the Buffer (p. 582) where the bytes are to be read.
--------------	---

Returns

the short at the given index in the buffer.

Exceptions

<i>IndexOutOfBoundsException</i>	if there are not enough bytes remaining to fill the requested Data Type, or index is negative.
----------------------------------	--

Implements **decaf::nio::ByteBuffer** (p. 706).

6.94.3.26 `virtual bool decaf::internal::nio::ByteBuffer::hasArray () const
[inline, virtual]`

Tells whether or not this buffer is backed by an accessible byte array.

If this method returns true then the array and arrayOffset methods may safely be invoked. Subclasses should override this method if they do not have a backing array as this class always returns true.

Returns

true if, and only if, this buffer is backed by an array and is not read-only.

Implements **decaf::nio::ByteBuffer** (p. 707).

6.94.3.27 `virtual bool decaf::internal::nio::ByteBuffer::isReadOnly () const
[inline, virtual]`

Tells whether or not this buffer is read-only.

Returns

true if, and only if, this buffer is read-only

Implements **decaf::nio::ByteBuffer** (p. 707).

6.94.3.28 virtual **ByteBuffer&** **decaf::internal::nio::ByteBuffer::put** (
 unsigned char *value*) [virtual]

Writes the given byte into this buffer at the current position, and then increments the position.

Parameters

<i>value</i>	- the byte value to be written.
--------------	---------------------------------

Returns

a reference to this buffer.

Exceptions

BufferOverflow-Exception (p. 609)	if this buffer's current position is not smaller than its limit.
ReadOnlyBuffer-Exception (p. 2244)	if this buffer is read-only.

Implements **decaf::nio::ByteBuffer** (p. 709).

6.94.3.29 virtual **ByteBuffer&** **decaf::internal::nio::ByteBuffer::put** (int
 index, unsigned char *value*) [virtual]

Writes the given byte into this buffer at the given index.

Parameters

<i>index</i>	- position in the Buffer (p. 582) to write the data
<i>value</i>	- the byte to write.

Returns

a reference to this buffer.

Exceptions

IndexOutOfBoundsException	if index greater than the buffer's limit minus the size of the type being written, or index is negative.
ReadOnlyBuffer-Exception (p. 2244)	if this buffer is read-only.

Implements **decaf::nio::ByteBuffer** (p. 710).

6.94.3.30 virtual **ByteBuffer&** decaf::internal::nio::ByteBuffer::putChar (
 char *value*) [virtual]

Writes one byte containing the given value, into this buffer at the current position, and then increments the position by one.

Parameters

<i>value</i>	The value to be written.
--------------	--------------------------

Returns

a reference to this buffer.

Exceptions

BufferOverflow-Exception (p. 609)	if there are fewer than bytes remaining in this buffer than the size of the data to be written
ReadOnlyBuffer-Exception (p. 2244)	if this buffer is read-only

Implements **decaf::nio::ByteBuffer** (p. 710).

6.94.3.31 virtual **ByteBuffer&** decaf::internal::nio::ByteBuffer::putChar (
 int *index*, char *value*) [virtual]

Writes one byte containing the given value, into this buffer at the given index.

Parameters

<i>index</i>	The position in the Buffer (p. 582) to write the data.
<i>value</i>	The value to write.

Returns

a reference to this buffer.

Exceptions

IndexOutOfBounds-Exception	if index greater than the buffer's limit minus the size of the type being written, or index is negative.
ReadOnlyBuffer-Exception (p. 2244)	if this buffer is read-only

Implements **decaf::nio::ByteBuffer** (p. 711).

6.94.3.32 virtual ByteBuffer& decaf::internal::nio::ByteBuffer::put-Double (double *value*) [virtual]

Writes eight bytes containing the given value, into this buffer at the current position, and then increments the position by eight.

Parameters

<i>value</i>	The value to be written.
--------------	--------------------------

Returns

a reference to this buffer.

Exceptions

BufferOverflow-Exception (p. 609)	if there are fewer than bytes remaining in this buffer than the size of the data to be written
ReadOnlyBuffer-Exception (p. 2244)	if this buffer is read-only

Implements **decaf::nio::ByteBuffer** (p. 711).

6.94.3.33 virtual ByteBuffer& decaf::internal::nio::ByteBuffer::putDouble (int *index*, double *value*) [virtual]

Writes eight bytes containing the given value, into this buffer at the given index.

Parameters

<i>index</i>	The position in the Buffer (p. 582) to write the data
<i>value</i>	The value to write.

Returns

a reference to this buffer.

Exceptions

IndexOutOfBoundsException	if index greater than the buffer's limit minus the size of the type being written, or index is negative.
ReadOnlyBuffer-Exception (p. 2244)	if this buffer is read-only.

Implements **decaf::nio::ByteBuffer** (p. 712).

6.94.3.34 virtual **ByteBuffer& decaf::internal::nio::ByteBuffer::putFloat** (
float *value*) [virtual]

Writes four bytes containing the given value, into this buffer at the current position, and then increments the position by eight.

Parameters

<i>value</i>	The value to be written.
--------------	--------------------------

Returns

a reference to this buffer.

Exceptions

BufferOverflow-Exception (p. 609)	if there are fewer than bytes remaining in this buffer than the size of the data to be written.
ReadOnlyBuffer-Exception (p. 2244)	if this buffer is read-only.

Implements **decaf::nio::ByteBuffer** (p. 712).

6.94.3.35 virtual **ByteBuffer& decaf::internal::nio::ByteBuffer::putFloat** (
int *index*, float *value*) [virtual]

Writes four bytes containing the given value, into this buffer at the given index.

Parameters

<i>index</i>	The position in the Buffer (p. 582) to write the data
<i>value</i>	The value to write.

Returns

a reference to this buffer.

Exceptions

IndexOutOfBounds-Exception	if index greater than the buffer's limit minus the size of the type being written, or index is negative.
ReadOnlyBuffer-Exception (p. 2244)	if this buffer is read-only.

Implements **decaf::nio::ByteBuffer** (p. 713).

6.94.3.36 **virtual ByteBuffer& decaf::internal::nio::ByteBuffer::putInt (int value)** [virtual]

Writes four bytes containing the given value, into this buffer at the current position, and then increments the position by eight.

Parameters

<i>value</i>	The value to be written.
--------------	--------------------------

Returns

a reference to this buffer.

Exceptions

BufferOverflow-Exception (p. 609)	if there are fewer than bytes remaining in this buffer than the size of the data to be written
ReadOnlyBuffer-Exception (p. 2244)	if this buffer is read-only

Implements **decaf::nio::ByteBuffer** (p. 713).

6.94.3.37 **virtual ByteBuffer& decaf::internal::nio::ByteBuffer::putInt (int index, int value)** [virtual]

Writes four bytes containing the given value, into this buffer at the given index.

Parameters

<i>index</i>	The position in the Buffer (p. 582) to write the data.
<i>value</i>	The value to write.

Returns

a reference to this buffer

Exceptions

IndexOutOfBoundsException	if index greater than the buffer's limit minus the size of the type being written, or index is negative.
ReadOnlyBuffer-Exception (p. 2244)	if this buffer is read-only

Implements **decaf::nio::ByteBuffer** (p. 714).

6.94.3.38 virtual **ByteBuffer& decaf::internal::nio::ByteBuffer::putLong** (
long long *value*) [virtual]

Writes eight bytes containing the given value, into this buffer at the current position, and then increments the position by eight.

Parameters

<i>value</i>	The value to be written.
--------------	--------------------------

Returns

a reference to this buffer.

Exceptions

BufferOverflow-Exception (p. 609)	if there are fewer than bytes remaining in this buffer than the size of the data to be written.
ReadOnlyBuffer-Exception (p. 2244)	if this buffer is read-only.

Implements **decaf::nio::ByteBuffer** (p. 714).

6.94.3.39 virtual **ByteBuffer& decaf::internal::nio::ByteBuffer::putLong** (
int *index*, long long *value*) [virtual]

Writes eight bytes containing the given value, into this buffer at the given index.

Parameters

<i>index</i>	The position in the Buffer (p. 582) to write the data.
<i>value</i>	The value to write.

Returns

a reference to this buffer.

Exceptions

IndexOutOfBounds-Exception	if index greater than the buffer's limit minus the size of the type being written, or index is negative.
ReadOnlyBuffer-Exception (p. 2244)	if this buffer is read-only.

Implements **decaf::nio::ByteBuffer** (p. 715).

6.94.3.40 virtual ByteBuffer& decaf::internal::nio::ByteBuffer::putShort (short value) [virtual]

Writes two bytes containing the given value, into this buffer at the current position, and then increments the position by eight.

Parameters

<i>value</i>	The value to be written.
--------------	--------------------------

Returns

a reference to this buffer.

Exceptions

BufferOverflow-Exception (p. 609)	if there are fewer than bytes remaining in this buffer than the size of the data to be written.
ReadOnlyBuffer-Exception (p. 2244)	if this buffer is read-only.

Implements **decaf::nio::ByteBuffer** (p. 715).

6.94.3.41 virtual ByteBuffer& decaf::internal::nio::ByteBuffer::putShort (int index, short value) [virtual]

Writes two bytes containing the given value, into this buffer at the given index.

Parameters

<i>index</i>	The position in the Buffer (p. 582) to write the data
<i>value</i>	The value to write.

Returns

a reference to this buffer

Exceptions

IndexOutOfBoundsException	if index greater than the buffer's limit minus the size of the type being written, or index is negative.
ReadOnlyBuffer-Exception (p. 2244)	if this buffer is read-only.

Implements **decaf::nio::ByteBuffer** (p. 716).

6.94.3.42 `virtual void decaf::internal::nio::ByteBuffer::setReadOnly (bool value) [inline, protected, virtual]`

Sets this **ByteBuffer** (p. 646) as Read-Only or not Read-Only.

Parameters

<i>value</i>	Boolean value, true if this buffer is to be read-only, false otherwise.
--------------	---

6.94.3.43 `virtual ByteBuffer* decaf::internal::nio::ByteBuffer::slice () const [virtual]`

Creates a new byte buffer whose content is a shared subsequence of this buffer's content.

The content of the new buffer will start at this buffer's current position. Changes to this buffer's content will be visible in the new buffer, and vice versa; the two buffers' position, limit, and mark values will be independent.

The new buffer's position will be zero, its capacity and its limit will be the number of bytes remaining in this buffer, and its mark will be undefined. The new buffer will be read-only if, and only if, this buffer is read-only.

Returns

the newly create **ByteBuffer** (p. 690) which the caller owns.

Implements **decaf::nio::ByteBuffer** (p. 716).

The documentation for this class was generated from the following file:

- `src/main/decaf/internal/nio/ByteBuffer.h`

6.95 decaf::io::ByteArrayInputStream Class Reference

A **ByteArrayInputStream** (p. 679) contains an internal buffer that contains bytes that may be read from the stream.

```
#include <src/main/decaf/io/ByteArrayInputStream.h>
```

Inheritance diagram for **decaf::io::ByteArrayInputStream**:

Public Member Functions

- **ByteArrayInputStream** ()

Creates an **ByteArrayInputStream** (p. 679) with an empty input buffer, the buffer can be initialized with a call to `setByteArray`.

- **ByteArrayInputStream** (const std::vector< unsigned char > &buffer)

Creates the input stream and calls `setBuffer` with the specified buffer object.

- **ByteArrayInputStream** (const unsigned char *buffer, int bufferSize, bool own=false)

Create an instance of the **ByteArrayInputStream** (p. 679) with the given buffer as the source of input for all read operations.

- **ByteArrayInputStream** (const unsigned char *buffer, int bufferSize, int offset, int length, bool own=false)

Create an instance of the **ByteArrayInputStream** (p. 679) with the given buffer as the source of input for all read operations.

- virtual ~**ByteArrayInputStream** ()

- virtual void **setByteArray** (const std::vector< unsigned char > &buffer)

Sets the internal buffer.

- virtual void **setByteArray** (const unsigned char *buffer, int bufferSize)

Sets the data that this reader uses, replaces any existing data and resets to beginning of the buffer.

- virtual void **setByteArray** (const unsigned char *buffer, int bufferSize, int offset, int length)

Sets the data that this reader uses, replaces any existing data and resets to beginning of the buffer.

- virtual int **available** () const

Indicates the number of bytes available.

The default implementation of this methods returns 0. Classes that override this method may return the total number of bytes that are currently available for reading and others may simply return a value of one indicating that there is some data available. The caller should view the result of this method as an absolute.

The default implementation of this method returns zero.

Returns

the number of bytes available on this input stream.

Exceptions

IOException (p. 1545)	if an I/O error occurs.
------------------------------	-------------------------

- virtual long long **skip** (long long num)

Skips over and discards *n* bytes of data from this input stream.

The skip method may, for a variety of reasons, end up skipping over some smaller number of bytes, possibly 0. This may result from any of a number of conditions; reaching end of file before *n* bytes have been skipped is only one possibility. The actual number of bytes skipped is returned.

The skip method of **InputStream** (p. 1464) creates a byte array and then repeatedly reads into it until *num* bytes have been read or the end of the stream has been reached. Subclasses are encouraged to provide a more efficient implementation of this method.

Parameters

num	The number of bytes to skip.
-----	------------------------------

Returns

total bytes skipped

Exceptions

IOException (p. 1545)	<i>if an I/O error occurs.</i>
UnsupportedOperation-Exception	<i>if the concrete stream class does not support skipping bytes.</i>

- virtual void **mark** (int readLimit)

Marks the current position in the stream. A subsequent call to the reset method repositions this stream at the last marked position so that subsequent reads re-read the same bytes.

If a stream instance reports that marks are supported then the stream will ensure that the same bytes can be read again after the reset method is called so long the readLimit is not reached.

Calling mark on a closed stream instance should have no effect.

The default implementation of this method does nothing.

Parameters

readLimit	<i>The max bytes read before marked position is invalid.</i>
-----------	--

- virtual void **reset** ()

Repositions this stream to the position at the time the mark method was last called on this input stream.

*If the method markSupported returns true, then: * If the method mark has not been called since the stream was created, or the number of bytes read from the stream since mark was last called is larger than the argument to mark at that last call, then an **IOException** (p. 1545) might be thrown. * If such an **IOException** (p. 1545) is not thrown, then the stream is reset to a state such that all the bytes read since the most recent call to mark (or since the start of the file, if mark has not been called) will be resupplied to subsequent callers of the read method, followed by any bytes that otherwise would have been the next input data as of the time of the call to reset.*

*If the method markSupported returns false, then: * The call to reset may throw an **IOException** (p. 1545). * If an **IOException** (p. 1545) is not thrown, then the stream is reset to a fixed state that depends on the particular type of the input stream and how it was created. The bytes that will be supplied to subsequent callers of the read method depend on the particular type of the input stream.*

*The default implementation of this method throws an **IOException** (p. 1545).*

Exceptions

IOException (p. 1545)	<i>if an I/O error occurs.</i>
------------------------------	--------------------------------

- virtual bool **markSupported** () const

Determines if this input stream supports the mark and reset methods.

Whether or not mark and reset are supported is an invariant property of a particular input stream instance.

The default implementation of this method returns false.

Returns

true if this stream instance supports marks

Protected Member Functions

- virtual int **doReadByte** ()

- virtual int **doReadArrayBounded** (unsigned char *buffer, int size, int offset, int length)

6.95.1 Detailed Description

A **ByteArrayInputStream** (p. 679) contains an internal buffer that contains bytes that may be read from the stream.

An internal counter keeps track of the next byte to be supplied by the read method. The **ByteArrayInputStream** (p. 679) never copies the supplied buffers, only points to them, therefore the caller must ensure that the supplied buffer remain in scope, or is not deleted before this **ByteArrayInputStream** (p. 679) is freed. If the own argument of one of the constructors that accepts an array pointer is set to true than the **ByteArrayInputStream** (p. 679) instance will take ownership of the supplied pointer and delete it when that instance is destroyed.

Closing a **ByteArrayInputStream** (p. 679) has no effect. The methods in this class can be called after the stream has been closed without generating an **IOException** (p. 1545).

Since

1.0

6.95.2 Constructor & Destructor Documentation

6.95.2.1 decaf::io::ByteArrayInputStream::ByteArrayInputStream ()

Creates an **ByteArrayInputStream** (p. 679) with an empty input buffer, the buffer can be initialized with a call to `setByteArray`.

6.95.2.2 decaf::io::ByteArrayInputStream::ByteArrayInputStream (const std::vector< unsigned char > & buffer)

Creates the input stream and calls `setBuffer` with the specified buffer object.

Parameters

<i>buffer</i>	The buffer to be used.
---------------	------------------------

6.95.2.3 decaf::io::ByteArrayInputStream::ByteArrayInputStream (const unsigned char * buffer, int bufferSize, bool own = false)

Create an instance of the **ByteArrayInputStream** (p. 679) with the given buffer as the source of input for all read operations.

Parameters

<i>buffer</i>	The initial byte array to use to read from.
<i>bufferSize</i>	The size of the buffer.
<i>own</i>	Indicates if this object should take ownership of the array, default is false.

Exceptions

<i>NullPointerException</i>	if the buffer is Null.
<i>IllegalArgument-Exception</i>	if the bufferSize is negative.

6.95.2.4 `decaf::io::ByteArrayInputStream::ByteArrayInputStream (const unsigned char * buffer, int bufferSize, int offset, int length, bool own = false)`

Create an instance of the **ByteArrayInputStream** (p. 679) with the given buffer as the source of input for all read operations.

Parameters

<i>buffer</i>	The initial byte array to use to read from.
<i>bufferSize</i>	The size of the buffer.
<i>offset</i>	The offset into the buffer to begin reading from.
<i>length</i>	The number of bytes to read past the offset.
<i>own</i>	Indicates if this object should take ownership of the array, default is false.

Exceptions

<i>NullPointerException</i>	if the buffer is Null.
<i>IllegalArgument-Exception</i>	if the bufferSize is negative.

6.95.2.5 `virtual decaf::io::ByteArrayInputStream::~~ByteArrayInputStream ()`
[virtual]

6.95.3 Member Function Documentation

6.95.3.1 `virtual int decaf::io::ByteArrayInputStream::available () const`
[virtual]

Indicates the number of bytes available.

The default implementation of this methods returns 0. Classes that override this method may return the total number of bytes that are currently available for reading and others may simply return a value of one indicating that there is some data available. The caller

should view the result of this method as an absolute.

The default implementation of this method returns zero.

Returns

the number of bytes available on this input stream.

Exceptions

<i>IOException</i> (p. 1545)	if an I/O error occurs.
--	-------------------------

Reimplemented from **decaf::io::InputStream** (p. 1466).

6.95.3.2 `virtual int decaf::io::ByteArrayInputStream::doReadArrayBounded
(unsigned char * buffer, int size, int offset, int length)` [protected,
virtual]

Reimplemented from **decaf::io::InputStream** (p. 1467).

6.95.3.3 `virtual int decaf::io::ByteArrayInputStream::doReadByte ()`
[protected, virtual]

Implements **decaf::io::InputStream** (p. 1467).

6.95.3.4 `virtual void decaf::io::ByteArrayInputStream::mark (int readLimit)`
[virtual]

Marks the current position in the stream. A subsequent call to the reset method repositions this stream at the last marked position so that subsequent reads re-read the same bytes.

If a stream instance reports that marks are supported then the stream will ensure that the same bytes can be read again after the reset method is called so long the readLimit is not reached.

Calling mark on a closed stream instance should have no effect.

The default implementation of this method does nothing.

Parameters

<i>readLimit</i>	The max bytes read before marked position is invalid.
------------------	---

Reimplemented from **decaf::io::InputStream** (p. 1468).

6.95.3.5 `virtual bool decaf::io::ByteArrayInputStream::markSupported () const`
`[inline, virtual]`

Determines if this input stream supports the mark and reset methods.

Whether or not mark and reset are supported is an invariant property of a particular input stream instance.

The default implementation of this method returns false.

Returns

true if this stream instance supports marks

Reimplemented from **decaf::io::InputStream** (p. 1468).

6.95.3.6 `virtual void decaf::io::ByteArrayInputStream::reset ()` `[virtual]`

Repositions this stream to the position at the time the mark method was last called on this input stream.

If the method `markSupported` returns true, then: * If the method `mark` has not been called since the stream was created, or the number of bytes read from the stream since mark was last called is larger than the argument to `mark` at that last call, then an **IOException** (p. 1545) might be thrown. * If such an **IOException** (p. 1545) is not thrown, then the stream is reset to a state such that all the bytes read since the most recent call to `mark` (or since the start of the file, if `mark` has not been called) will be resupplied to subsequent callers of the `read` method, followed by any bytes that otherwise would have been the next input data as of the time of the call to `reset`.

If the method `markSupported` returns false, then: * The call to `reset` may throw an **IOException** (p. 1545). * If an **IOException** (p. 1545) is not thrown, then the stream is reset to a fixed state that depends on the particular type of the input stream and how it was created. The bytes that will be supplied to subsequent callers of the `read` method depend on the particular type of the input stream.

The default implementation of this method throws an **IOException** (p. 1545).

Exceptions

IOException (p. 1545)	if an I/O error occurs.
---------------------------------	-------------------------

Reimplemented from **decaf::io::InputStream** (p. 1471).

6.95.3.7 `virtual void decaf::io::ByteArrayInputStream::setByteArray (const`
`std::vector< unsigned char > & buffer)` `[virtual]`

Sets the internal buffer.

The input stream will wrap around this buffer and will perform all read operations on it. The position will be reinitialized to the beginning of the specified buffer. This class will

not own the given buffer - it is the caller's responsibility to free the memory of the given buffer as appropriate.

Parameters

<i>buffer</i>	The buffer to be used.
---------------	------------------------

6.95.3.8 virtual void decaf::io::ByteArrayInputStream::setByteArray (const unsigned char * *buffer*, int *bufferSize*) [virtual]

Sets the data that this reader uses, replaces any existing data and resets to beginning of the buffer.

Parameters

<i>buffer</i>	The initial byte array to use to read from.
<i>bufferSize</i>	The size of the buffer.

Exceptions

<i>NullPointerException</i>	if the buffer is Null.
<i>IllegalArgument-Exception</i>	if the bufferSize is negative.

6.95.3.9 virtual void decaf::io::ByteArrayInputStream::setByteArray (const unsigned char * *buffer*, int *bufferSize*, int *offset*, int *length*) [virtual]

Sets the data that this reader uses, replaces any existing data and resets to beginning of the buffer.

Parameters

<i>buffer</i>	The initial byte array to use to read from.
<i>bufferSize</i>	The size of the buffer.
<i>offset</i>	The offset into the buffer to begin reading from.
<i>length</i>	The number of bytes to read past the offset.

Exceptions

<i>NullPointerException</i>	if the buffer is Null.
<i>IllegalArgument-Exception</i>	if the bufferSize is negative.

6.95.3.10 virtual long long decaf::io::ByteArrayInputStream::skip (long long *num*)
[virtual]

Skips over and discards *n* bytes of data from this input stream.

The skip method may, for a variety of reasons, end up skipping over some smaller number of bytes, possibly 0. This may result from any of a number of conditions; reaching end of file before *n* bytes have been skipped is only one possibility. The actual number of bytes skipped is returned.

The skip method of **InputStream** (p. 1464) creates a byte array and then repeatedly reads into it until *num* bytes have been read or the end of the stream has been reached. Subclasses are encouraged to provide a more efficient implementation of this method.

Parameters

<i>num</i>	The number of bytes to skip.
------------	------------------------------

Returns

total bytes skipped

Exceptions

<i>IOException</i> (p. 1545)	if an I/O error occurs.
<i>Unsupported-Operation-Exception</i>	if the concrete stream class does not support skipping bytes.

Reimplemented from **decaf::io::InputStream** (p. 1472).

The documentation for this class was generated from the following file:

- src/main/decaf/io/**ByteArrayInputStream.h**

6.96 decaf::io::ByteArrayOutputStream Class Reference

```
#include <src/main/decaf/io/ByteArrayOutputStream.h>
```

Inheritance diagram for decaf::io::ByteArrayOutputStream:

Public Member Functions

- **ByteArrayOutputStream ()**
Default Constructor - uses a default internal buffer of 32 bytes, the size increases as the need for more room arises.

- **ByteArrayOutputStream** (int bufferSize)
Creates a **ByteArrayOutputStream** (p. 687) with an internal buffer allocated with the given size.
- virtual ~**ByteArrayOutputStream** ()
- std::pair< unsigned char *, int > **toByteArray** () const
Creates a newly allocated byte array.
- long long **size** () const
Gets the current count of bytes written into this **ByteArrayOutputStream** (p. 687).
- virtual void **reset** ()
Clear current Stream contents.
- virtual std::string **toString** () const
Converts the bytes in the buffer into a standard C++ string.
- void **writeTo** (**OutputStream** *out) const
Writes the complete contents of this byte array output stream to the specified output stream argument, as if by calling the output stream's write method using out.write(buf, 0, count).

Protected Member Functions

- virtual void **doWriteByte** (unsigned char value)
- virtual void **doWriteArrayBounded** (const unsigned char *buffer, int size, int offset, int length)

6.96.1 Constructor & Destructor Documentation

6.96.1.1 decaf::io::ByteArrayOutputStream::ByteArrayOutputStream ()

Default Constructor - uses a default internal buffer of 32 bytes, the size increases as the need for more room arises.

6.96.1.2 decaf::io::ByteArrayOutputStream::ByteArrayOutputStream (int bufferSize)

Creates a **ByteArrayOutputStream** (p. 687) with an internal buffer allocated with the given size.

Parameters

<i>bufferSize</i>	The size to use for the internal buffer.
-------------------	--

Exceptions

<i>IllegalArgumentException</i>	if the size is less than or equal to zero.
---------------------------------	--

6.96.1.3 virtual **decaf::io::ByteArrayOutputStream::~~ByteArrayOutputStream** ()
[virtual]

6.96.2 Member Function Documentation

6.96.2.1 virtual void **decaf::io::ByteArrayOutputStream::doWriteArrayBounded** (
const unsigned char * *buffer*, int *size*, int *offset*, int *length*) [protected,
virtual]

Reimplemented from **decaf::io::OutputStream** (p. 2068).

6.96.2.2 virtual void **decaf::io::ByteArrayOutputStream::doWriteByte** (unsigned char
value) [protected, virtual]

Implements **decaf::io::OutputStream** (p. 2069).

6.96.2.3 virtual void **decaf::io::ByteArrayOutputStream::reset** () [virtual]

Clear current Stream contents.

Exceptions

<i>IOException</i> (p. 1545)	
--	--

6.96.2.4 long long **decaf::io::ByteArrayOutputStream::size** () const

Gets the current count of bytes written into this **ByteArrayOutputStream** (p. 687).

Returns

the number of valid bytes contained in the **ByteArrayOutputStream** (p. 687).

6.96.2.5 std::pair<unsigned char*, int> **decaf::io::ByteArrayOutputStream::toByte-
Array** () const

Creates a newly allocated byte array.

Its size is the current size of this output stream and the valid contents of the buffer have been copied into it. The newly allocated array and its size are returned inside an STL pair structure, the caller is responsible for freeing the returned array.

Returns

an STL pair containing the copied array and its size.

6.96.2.6 `virtual std::string decaf::io::ByteArrayOutputStream::toString () const`
`[virtual]`

Converts the bytes in the buffer into a standard C++ string.

Returns

a string containing the bytes in the buffer

Reimplemented from **decaf::io::OutputStream** (p. 2070).

6.96.2.7 `void decaf::io::ByteArrayOutputStream::writeTo (OutputStream * out)`
`const`

Writes the complete contents of this byte array output stream to the specified output stream argument, as if by calling the output stream's write method using `out.write(buf, 0, count)`.

The documentation for this class was generated from the following file:

- `src/main/decaf/io/ByteArrayOutputStream.h`

6.97 decaf::nio::ByteBuffer Class Reference

This class defines six categories of operations upon byte buffers:

```
#include <src/main/decaf/nio/ByteBuffer.h>
```

Inheritance diagram for `decaf::nio::ByteBuffer`:

Public Member Functions

- `virtual ~ByteBuffer ()`
- `virtual std::string toString () const`
- **ByteBuffer & get** (std::vector< unsigned char > buffer)
Relative bulk get method.
- **ByteBuffer & get** (unsigned char *buffer, int size, int offset, int length)
Relative bulk get method.
- **ByteBuffer & put (ByteBuffer &src)**
This method transfers the bytes remaining in the given source buffer into this buffer.
- **ByteBuffer & put** (const unsigned char *buffer, int size, int offset, int length)
This method transfers bytes into this buffer from the given source array.
- **ByteBuffer & put** (std::vector< unsigned char > &buffer)
This method transfers the entire content of the given source byte array into this buffer.

- virtual bool **isReadOnly** () const =0
Tells whether or not this buffer is read-only.
- virtual unsigned char * **array** ()=0
Returns the byte array that backs this buffer.
- virtual int **arrayOffset** () const =0
Returns the offset within this buffer's backing array of the first element of the buffer.
- virtual bool **hasArray** () const =0
Tells whether or not this buffer is backed by an accessible byte array.
- virtual **CharBuffer** * **asCharBuffer** () const =0
Creates a view of this byte buffer as a char buffer.
- virtual **DoubleBuffer** * **asDoubleBuffer** () const =0
Creates a view of this byte buffer as a double buffer.
- virtual **FloatBuffer** * **asFloatBuffer** () const =0
Creates a view of this byte buffer as a float buffer.
- virtual **IntBuffer** * **asIntBuffer** () const =0
Creates a view of this byte buffer as a int buffer.
- virtual **LongBuffer** * **asLongBuffer** () const =0
Creates a view of this byte buffer as a long buffer.
- virtual **ShortBuffer** * **asShortBuffer** () const =0
Creates a view of this byte buffer as a short buffer.
- virtual **ByteBuffer** * **asReadOnlyBuffer** () const =0
Creates a new, read-only byte buffer that shares this buffer's content.
- virtual **ByteBuffer** & **compact** ()=0
Compacts this buffer.
- virtual **ByteBuffer** * **duplicate** ()=0
Creates a new byte buffer that shares this buffer's content.
- virtual unsigned char **get** () const =0
Relative get method.
- virtual unsigned char **get** (int index) const =0
Absolute get method.
- virtual char **getChar** ()=0
Reads the next byte at this buffer's current position, and then increments the position by one.
- virtual char **getChar** (int index) const =0
Reads one byte at the given index and returns it.
- virtual double **getDouble** ()=0
Reads the next eight bytes at this buffer's current position, and then increments the position by that amount.
- virtual double **getDouble** (int index) const =0
Reads eight bytes at the given index and returns it.
- virtual float **getFloat** ()=0
Reads the next four bytes at this buffer's current position, and then increments the position by that amount.
- virtual float **getFloat** (int index) const =0

Reads four bytes at the given index and returns it.

- virtual long long **getLong** ()=0
Reads the next eight bytes at this buffer's current position, and then increments the position by that amount.
- virtual long long **getLong** (int index) const =0
Reads eight bytes at the given index and returns it.
- virtual int **getInt** ()=0
Reads the next four bytes at this buffer's current position, and then increments the position by that amount.
- virtual int **getInt** (int index) const =0
Reads four bytes at the given index and returns it.
- virtual short **getShort** ()=0
Reads the next two bytes at this buffer's current position, and then increments the position by that amount.
- virtual short **getShort** (int index) const =0
Reads two bytes at the given index and returns it.
- virtual **ByteBuffer & put** (unsigned char value)=0
Writes the given byte into this buffer at the current position, and then increments the position.
- virtual **ByteBuffer & put** (int index, unsigned char value)=0
Writes the given byte into this buffer at the given index.
- virtual **ByteBuffer & putChar** (char value)=0
Writes one byte containing the given value, into this buffer at the current position, and then increments the position by one.
- virtual **ByteBuffer & putChar** (int index, char value)=0
Writes one byte containing the given value, into this buffer at the given index.
- virtual **ByteBuffer & putDouble** (double value)=0
Writes eight bytes containing the given value, into this buffer at the current position, and then increments the position by eight.
- virtual **ByteBuffer & putDouble** (int index, double value)=0
Writes eight bytes containing the given value, into this buffer at the given index.
- virtual **ByteBuffer & putFloat** (float value)=0
Writes four bytes containing the given value, into this buffer at the current position, and then increments the position by eight.
- virtual **ByteBuffer & putFloat** (int index, float value)=0
Writes four bytes containing the given value, into this buffer at the given index.
- virtual **ByteBuffer & putLong** (long long value)=0
Writes eight bytes containing the given value, into this buffer at the current position, and then increments the position by eight.
- virtual **ByteBuffer & putLong** (int index, long long value)=0
Writes eight bytes containing the given value, into this buffer at the given index.
- virtual **ByteBuffer & putInt** (int value)=0
Writes four bytes containing the given value, into this buffer at the current position, and then increments the position by eight.
- virtual **ByteBuffer & putInt** (int index, int value)=0

Writes four bytes containing the given value, into this buffer at the given index.

- virtual **ByteBuffer** & **putShort** (short value)=0

Writes two bytes containing the given value, into this buffer at the current position, and then increments the position by eight.

- virtual **ByteBuffer** & **putShort** (int index, short value)=0

Writes two bytes containing the given value, into this buffer at the given index.

- virtual **ByteBuffer** * **slice** () const =0

Creates a new byte buffer whose content is a shared subsequence of this buffer's content.

- virtual int **compareTo** (const **ByteBuffer** &value) const

- virtual bool **equals** (const **ByteBuffer** &value) const

- virtual bool **operator==** (const **ByteBuffer** &value) const

- virtual bool **operator<** (const **ByteBuffer** &value) const

Static Public Member Functions

- static **ByteBuffer** * **allocate** (int capacity)

Allocates a new byte buffer whose position will be zero its limit will be its capacity and its mark is not set.

- static **ByteBuffer** * **wrap** (unsigned char *array, int size, int offset, int length)

*Wraps the passed buffer with a new **ByteBuffer** (p. 690).*

- static **ByteBuffer** * **wrap** (std::vector< unsigned char > &buffer)

*Wraps the passed STL Byte Vector in a **ByteBuffer** (p. 690).*

Protected Member Functions

- **ByteBuffer** (int capacity)

*Creates a **ByteBuffer** (p. 690) object that has its backing array allocated internally and is then owned and deleted when this object is deleted.*

6.97.1 Detailed Description

This class defines six categories of operations upon byte buffers:

1. Absolute and relative get and put methods that read and write single bytes;
2. - Relative bulk get methods that transfer contiguous sequences of bytes from this buffer into an array;
3. Relative bulk put methods that transfer contiguous sequences of bytes from a byte array or some other byte buffer into this buffer;
4. Absolute and relative get and put methods that read and write values of other primitive types, translating them to and from sequences of bytes in a particular byte order;
5. Methods for creating view buffers, which allow a byte buffer to be viewed as a buffer containing values of some

other primitive type; and 6. Methods for compacting, duplicating, and slicing a byte buffer.

Byte buffers can be created either by allocation, which allocates space for the buffer's content, or by wrapping an existing byte array into a buffer.

Access to binary data:

This class defines methods for reading and writing values of all other primitive types, except boolean. Primitive values are translated to (or from) sequences of bytes according to the buffer's current byte order.

For access to heterogeneous binary data, that is, sequences of values of different types, this class defines a family of absolute and relative get and put methods for each type. For 32-bit floating-point values, for example, this class defines:

float **getFloat()** (p. 703) float **getFloat(int index)** void **putFloat(float f)** (p. 712) void **putFloat(int index, float f)** (p. 713)

Corresponding methods are defined for the types char, short, int, long, and double. The index parameters of the absolute get and put methods are in terms of bytes rather than of the type being read or written.

For access to homogeneous binary data, that is, sequences of values of the same type, this class defines methods that can create views of a given byte buffer. A view buffer is simply another buffer whose content is backed by the byte buffer. Changes to the byte buffer's content will be visible in the view buffer, and vice versa; the two buffers' position, limit, and mark values are independent. The **asFloatBuffer** method, for example, creates an instance of the **FloatBuffer** (p. 1368) class that is backed by the byte buffer upon which the method is invoked. Corresponding view-creation methods are defined for the types char, short, int, long, and double.

View buffers have two important advantages over the families of type-specific get and put methods described above:

A view buffer is indexed not in terms of bytes but rather in terms of the type-specific size of its values;

A view buffer provides relative bulk get and put methods that can transfer contiguous sequences of values between a buffer and an array or some other buffer of the same type; and

6.97.2 Constructor & Destructor Documentation

6.97.2.1 `decaf::nio::ByteBuffer::ByteBuffer (int capacity)` [protected]

Creates a **ByteBuffer** (p. 690) object that has its backing array allocated internally and is then owned and deleted when this object is deleted.

The array is initially created with all elements initialized to zero.

Parameters

<i>capacity</i>	The size of the array, this is the limit we read and write to.
-----------------	--

Exceptions

<i>IllegalArgument-Exception</i>	if capacity is negative.
----------------------------------	--------------------------

6.97.2.2 virtual decaf::nio::ByteBuffer::~ByteBuffer () [inline, virtual]

6.97.3 Member Function Documentation

6.97.3.1 static ByteBuffer* decaf::nio::ByteBuffer::allocate (int capacity)
[static]

Allocates a new byte buffer whose position will be zero its limit will be its capacity and its mark is not set.

Parameters

<i>capacity</i>	The internal buffer's capacity.
-----------------	---------------------------------

Returns

a newly allocated **ByteBuffer** (p. 690) which the caller owns.

Exceptions

<i>IllegalArgument-Exception</i>	if capacity is negative.
----------------------------------	--------------------------

6.97.3.2 virtual unsigned char* decaf::nio::ByteBuffer::array () [pure virtual]

Returns the byte array that backs this buffer.

Modifications to this buffer's content will cause the returned array's content to be modified, and vice versa.

Invoke the `hasArray` method before invoking this method in order to ensure that this buffer has an accessible backing array.

Returns

The array that backs this buffer

Exceptions

<i>ReadOnlyBuffer-Exception</i> (p. 2244)	if this buffer is backed by an array but is read-only
--	---

<i>Unsupported- OperationException</i>	if this buffer is not backed by an accessible array
--	---

Implemented in **decaf::internal::nio::ByteBuffer** (p. 661).

6.97.3.3 `virtual int decaf::nio::ByteBuffer::arrayOffset () const` `[pure virtual]`

Returns the offset within this buffer's backing array of the first element of the buffer.

If this buffer is backed by an array then buffer position *p* corresponds to array index *p* + **arrayOffset()** (p. 696).

Invoke the `hasArray` method before invoking this method in order to ensure that this buffer has an accessible backing array.

Returns

The offset within this buffer's array of the first element of the buffer.

Exceptions

<i>ReadOnlyBuffer- Exception</i> (p. 2244)	if this buffer is backed by an array but is read-only.
<i>Unsupported- OperationException</i>	if this buffer is not backed by an accessible array.

Implemented in **decaf::internal::nio::ByteBuffer** (p. 661).

6.97.3.4 `virtual CharBuffer* decaf::nio::ByteBuffer::asCharBuffer () const` `[pure virtual]`

Creates a view of this byte buffer as a char buffer.

The content of the new buffer will start at this buffer's current position. Changes to this buffer's content will be visible in the new buffer, and vice versa; the two buffers' position, limit, and mark values will be independent.

The new buffer's position will be zero, its capacity and its limit will be the number of bytes remaining in this buffer, and its mark will be undefined. The new buffer will be read-only if, and only if, this buffer is read-only.

Returns

the new Char **Buffer** (p. 582), which the caller then owns.

Implemented in **decaf::internal::nio::ByteBuffer** (p. 662).

6.97.3.5 `virtual DoubleBuffer* decaf::nio::ByteBuffer::asDoubleBuffer () const`
[pure virtual]

Creates a view of this byte buffer as a double buffer.

The content of the new buffer will start at this buffer's current position. Changes to this buffer's content will be visible in the new buffer, and vice versa; the two buffers' position, limit, and mark values will be independent.

The new buffer's position will be zero, its capacity and its limit will be the number of bytes remaining in this buffer divided by eight, and its mark will be undefined. The new buffer will be read-only if, and only if, this buffer is read-only.

Returns

the new double **Buffer** (p. 582), which the caller then owns.

Implemented in **decaf::internal::nio::ByteArrayBuffer** (p. 662).

6.97.3.6 `virtual FloatBuffer* decaf::nio::ByteBuffer::asFloatBuffer () const`
[pure virtual]

Creates a view of this byte buffer as a float buffer.

The content of the new buffer will start at this buffer's current position. Changes to this buffer's content will be visible in the new buffer, and vice versa; the two buffers' position, limit, and mark values will be independent.

The new buffer's position will be zero, its capacity and its limit will be the number of bytes remaining in this buffer divided by four, and its mark will be undefined. The new buffer will be read-only if, and only if, this buffer is read-only.

Returns

the new float **Buffer** (p. 582), which the caller then owns.

Implemented in **decaf::internal::nio::ByteArrayBuffer** (p. 662).

6.97.3.7 `virtual IntBuffer* decaf::nio::ByteBuffer::asIntBuffer () const` [pure virtual]

Creates a view of this byte buffer as a int buffer.

The content of the new buffer will start at this buffer's current position. Changes to this buffer's content will be visible in the new buffer, and vice versa; the two buffers' position, limit, and mark values will be independent.

The new buffer's position will be zero, its capacity and its limit will be the number of bytes remaining in this buffer divided by four, and its mark will be undefined. The new buffer will be read-only if, and only if, this buffer is read-only.

Returns

the new int **Buffer** (p. 582), which the caller then owns.

Implemented in **decaf::internal::nio::ByteBuffer** (p. 663).

6.97.3.8 `virtual LongBuffer* decaf::nio::ByteBuffer::asLongBuffer () const`
`[pure virtual]`

Creates a view of this byte buffer as a long buffer.

The content of the new buffer will start at this buffer's current position. Changes to this buffer's content will be visible in the new buffer, and vice versa; the two buffers' position, limit, and mark values will be independent.

The new buffer's position will be zero, its capacity and its limit will be the number of bytes remaining in this buffer divided by eight, and its mark will be undefined. The new buffer will be read-only if, and only if, this buffer is read-only.

Returns

the new long **Buffer** (p. 582), which the caller then owns.

Implemented in **decaf::internal::nio::ByteBuffer** (p. 663).

6.97.3.9 `virtual ByteBuffer* decaf::nio::ByteBuffer::asReadOnlyBuffer () const`
`[pure virtual]`

Creates a new, read-only byte buffer that shares this buffer's content.

The content of the new buffer will be that of this buffer. Changes to this buffer's content will be visible in the new buffer; the new buffer itself, however, will be read-only and will not allow the shared content to be modified. The two buffers' position, limit, and mark values will be independent.

If this buffer is itself read-only then this method behaves in exactly the same way as the duplicate method.

The new buffer's capacity, limit, position, and mark values will be identical to those of this buffer.

Returns

The new, read-only byte buffer which the caller then owns.

Implemented in **decaf::internal::nio::ByteBuffer** (p. 663).

6.97.3.10 `virtual ShortBuffer* decaf::nio::ByteBuffer::asShortBuffer () const`
`[pure virtual]`

Creates a view of this byte buffer as a short buffer.

The content of the new buffer will start at this buffer's current position. Changes to this buffer's content will be visible in the new buffer, and vice versa; the two buffers' position, limit, and mark values will be independent.

The new buffer's position will be zero, its capacity and its limit will be the number of bytes remaining in this buffer divided by two, and its mark will be undefined. The new buffer will be read-only if, and only if, this buffer is read-only.

Returns

the new short **Buffer** (p. 582), which the caller then owns.

Implemented in **decaf::internal::nio::ByteBuffer** (p. 664).

6.97.3.11 virtual **ByteBuffer&** **decaf::nio::ByteBuffer::compact** () [pure virtual]

Compacts this buffer.

The bytes between the buffer's current position and its limit, if any, are copied to the beginning of the buffer. That is, the byte at index $p = \text{position}()$ (p. 587) is copied to index zero, the byte at index $p + 1$ is copied to index one, and so forth until the byte at index $\text{limit}()$ (p. 586) - 1 is copied to index $n = \text{limit}()$ (p. 586) - 1 - p . The buffer's position is then set to $n+1$ and its limit is set to its capacity. The mark, if defined, is discarded.

The buffer's position is set to the number of bytes copied, rather than to zero, so that an invocation of this method can be followed immediately by an invocation of another relative put method.

Returns

a reference to this **ByteBuffer** (p. 690).

Exceptions

ReadOnlyBufferException (p. 2244)	if this buffer is read-only.
---	------------------------------

Implemented in **decaf::internal::nio::ByteBuffer** (p. 664).

6.97.3.12 virtual int **decaf::nio::ByteBuffer::compareTo** (const **ByteBuffer** & *value*) const [virtual]

6.97.3.13 virtual **ByteBuffer*** **decaf::nio::ByteBuffer::duplicate** () [pure virtual]

Creates a new byte buffer that shares this buffer's content.

The content of the new buffer will be that of this buffer. Changes to this buffer's content

will be visible in the new buffer, and vice versa; the two buffers' position, limit, and mark values will be independent.

The new buffer's capacity, limit, position, and mark values will be identical to those of this buffer. The new buffer will be read-only if, and only if, this buffer is read-only.

Returns

a new **Byte Buffer** (p. 582) which the caller owns.

Implemented in **decaf::internal::nio::ByteBuffer** (p. 665).

6.97.3.14 `virtual bool decaf::nio::ByteBuffer::equals (const ByteBuffer & value) const`
[virtual]

6.97.3.15 `ByteBuffer& decaf::nio::ByteBuffer::get (std::vector< unsigned char >`
`buffer)`

Relative bulk get method.

This method transfers bytes from this buffer into the given destination vector. An invocation of this method of the form `src.get(a)` behaves in exactly the same way as the invocation. The vector must be sized to the amount of data that is to be read, that is to say, the caller should call `buffer.resize(N)` before calling this get method.

Returns

a reference to this **Byte Buffer** (p. 582).

Exceptions

BufferUnderflow-Exception (p. 611)	if there are fewer than length bytes remaining in this buffer
--	---

6.97.3.16 `ByteBuffer& decaf::nio::ByteBuffer::get (unsigned char * buffer, int size, int`
`offset, int length)`

Relative bulk get method.

This method transfers bytes from this buffer into the given destination array. If there are fewer bytes remaining in the buffer than are required to satisfy the request, that is, if `length > remaining()` (p. 588), then no bytes are transferred and a **BufferUnderflow-Exception** (p. 611) is thrown.

Otherwise, this method copies length bytes from this buffer into the given array, starting at the current position of this buffer and at the given offset in the array. The position of this buffer is then incremented by length.

Parameters

<i>buffer</i>	The pointer to an allocated buffer to fill.
<i>size</i>	The size of the passed in Buffer (p. 582).
<i>offset</i>	The position in the buffer to start filling.
<i>length</i>	The amount of data to put in the passed buffer.

Returns

a reference to this **Buffer** (p. 582).

Exceptions

<i>IndexOutOfBoundsException</i>	if the preconditions of size, offset, or length are not met.
BufferUnderflow-Exception (p. 611)	if there are fewer than length bytes remaining in this buffer.
<i>NullPointerException</i>	if the passed buffer is null.

6.97.3.17 virtual unsigned char decaf::nio::ByteBuffer::get () const [pure virtual]

Relative get method.

Reads the byte at this buffer's current position, and then increments the position.

Returns

The byte at the buffer's current position.

Exceptions

BufferUnderflow-Exception (p. 611)	if the buffer's current position is not smaller than its limit.
--	---

Implemented in **decaf::internal::nio::ByteArrayBuffer** (p. 665).

6.97.3.18 virtual unsigned char decaf::nio::ByteBuffer::get (int *index*) const [pure virtual]

Absolute get method.

Reads the byte at the given index.

Parameters

<i>index</i>	The index in the Buffer (p. 582) where the byte is to be read.
--------------	---

Returns

the byte that is located at the given index.

Exceptions

<i>IndexOutOfBoundsException</i>	if index is not smaller than the buffer's limit, or index is negative.
---	--

Implemented in **decaf::internal::nio::ByteArrayBuffer** (p. 666).

6.97.3.19 `virtual char decaf::nio::ByteBuffer::getChar () [pure virtual]`

Reads the next byte at this buffer's current position, and then increments the position by one.

Returns

the next char in the buffer.

Exceptions

<i>BufferUnderflowException</i> (p. 611)	if there are no more bytes remaining in this buffer, meaning we have reached the set limit.
--	---

Implemented in **decaf::internal::nio::ByteArrayBuffer** (p. 666).

6.97.3.20 `virtual char decaf::nio::ByteBuffer::getChar (int index) const [pure virtual]`

Reads one byte at the given index and returns it.

Parameters

<i>index</i>	The index in the Buffer (p. 582) where the byte is to be read.
--------------	---

Returns

the char at the given index in the buffer

Exceptions

<i>IndexOutOfBoundsException</i>	if index is not smaller than the buffer's limit, or index is negative.
---	--

Implemented in **decaf::internal::nio::ByteBuffer** (p. 666).

6.97.3.21 virtual double **decaf::nio::ByteBuffer::getDouble** () [pure virtual]

Reads the next eight bytes at this buffer's current position, and then increments the position by that amount.

Returns

the next double in the buffer.

Exceptions

<i>BufferUnderflow-Exception</i> (p. 611)	if there are no more bytes remaining in this buffer, meaning we have reached the set limit.
---	---

Implemented in **decaf::internal::nio::ByteBuffer** (p. 667).

6.97.3.22 virtual double **decaf::nio::ByteBuffer::getDouble** (int *index*) const [pure virtual]

Reads eight bytes at the given index and returns it.

Parameters

<i>index</i>	The index in the Buffer (p. 582) where the bytes are to be read.
--------------	---

Returns

the double at the given index in the buffer.

Exceptions

<i>IndexOutOfBounds-Exception</i>	if index is not smaller than the buffer's limit, or index is negative.
--	--

Implemented in **decaf::internal::nio::ByteBuffer** (p. 667).

6.97.3.23 virtual float **decaf::nio::ByteBuffer::getFloat** () [pure virtual]

Reads the next four bytes at this buffer's current position, and then increments the position by that amount.

Returns

the next float in the buffer.

Exceptions

<i>BufferUnderflow-Exception</i> (p. 611)	if there are no more bytes remaining in this buffer, meaning we have reached the set limit.
---	---

Implemented in **decaf::internal::nio::ByteArrayBuffer** (p. 668).

6.97.3.24 `virtual float decaf::nio::ByteBuffer::getFloat (int index) const` [pure virtual]

Reads four bytes at the given index and returns it.

Parameters

<i>index</i>	The index in the Buffer (p. 582) where the bytes are to be read.
--------------	---

Returns

the float at the given index in the buffer.

Exceptions

<i>IndexOutOfBoundsException</i>	if there are not enough bytes remaining to fill the requested Data Type, or index is negative.
---	--

Implemented in **decaf::internal::nio::ByteArrayBuffer** (p. 668).

6.97.3.25 `virtual int decaf::nio::ByteBuffer::getInt ()` [pure virtual]

Reads the next four bytes at this buffer's current position, and then increments the position by that amount.

Returns

the next int in the buffer.

Exceptions

<i>BufferUnderflow-Exception</i> (p. 611)	if there are no more bytes remaining in this buffer, meaning we have reached the set limit.
---	---

Implemented in **decaf::internal::nio::ByteBuffer** (p. 669).

6.97.3.26 `virtual int decaf::nio::ByteBuffer::getInt (int index) const` `[pure virtual]`

Reads four bytes at the given index and returns it.

Parameters

<i>index</i>	The index in the Buffer (p. 582) where the bytes are to be read.
--------------	---

Returns

the int at the given index in the buffer.

Exceptions

<i>IndexOutOfBoundsException</i>	if there are not enough bytes remaining to fill the requested Data Type, or index is negative.
----------------------------------	--

Implemented in **decaf::internal::nio::ByteBuffer** (p. 669).

6.97.3.27 `virtual long long decaf::nio::ByteBuffer::getLong ()` `[pure virtual]`

Reads the next eight bytes at this buffer's current position, and then increments the position by that amount.

Returns

the next long long in the buffer.

Exceptions

<i>BufferUnderflow-Exception</i> (p. 611)	if there are no more bytes remaining in this buffer, meaning we have reached the set limit.
--	---

Implemented in **decaf::internal::nio::ByteBuffer** (p. 669).

6.97.3.28 `virtual long long decaf::nio::ByteBuffer::getLong (int index) const` `[pure virtual]`

Reads eight bytes at the given index and returns it.

Parameters

<i>index</i>	The index in the Buffer (p. 582) where the bytes are to be read.
--------------	---

Returns

the long long at the given index in the buffer.

Exceptions

<i>IndexOutOfBoundsException</i>	if there are not enough bytes remaining to fill the requested Data Type, or index is negative.
---	--

Implemented in **decaf::internal::nio::ByteArrayBuffer** (p. 670).

6.97.3.29 virtual short **decaf::nio::ByteBuffer::getShort** () [pure virtual]

Reads the next two bytes at this buffer's current position, and then increments the position by that amount.

Returns

the next short in the buffer.

Exceptions

<i>BufferUnderflowException</i> (p. 611)	if there are no more bytes remaining in this buffer, meaning we have reached the set limit.
--	---

Implemented in **decaf::internal::nio::ByteArrayBuffer** (p. 670).

6.97.3.30 virtual short **decaf::nio::ByteBuffer::getShort** (int *index*) const [pure virtual]

Reads two bytes at the given index and returns it.

Parameters

<i>index</i>	The index in the Buffer (p. 582) where the bytes are to be read.
--------------	---

Returns

the short at the given index in the buffer.

Exceptions

<i>IndexOutOfBoundsException</i>	if there are not enough bytes remaining to fill the requested Data Type, or index is negative.
---	--

Implemented in **decaf::internal::nio::ByteBuffer** (p. 671).

6.97.3.31 `virtual bool decaf::nio::ByteBuffer::hasArray () const [pure virtual]`

Tells whether or not this buffer is backed by an accessible byte array.

If this method returns true then the array and arrayOffset methods may safely be invoked. Subclasses should override this method if they do not have a backing array as this class always returns true.

Returns

true if, and only if, this buffer is backed by an array and is not read-only.

Implemented in **decaf::internal::nio::ByteBuffer** (p. 671).

6.97.3.32 `virtual bool decaf::nio::ByteBuffer::isReadOnly () const [pure virtual]`

Tells whether or not this buffer is read-only.

Returns

true if, and only if, this buffer is read-only

Implements **decaf::nio::Buffer** (p. 586).

Implemented in **decaf::internal::nio::ByteBuffer** (p. 671).

6.97.3.33 `virtual bool decaf::nio::ByteBuffer::operator< (const ByteBuffer & value) const [virtual]`

6.97.3.34 `virtual bool decaf::nio::ByteBuffer::operator== (const ByteBuffer & value) const [virtual]`

6.97.3.35 `ByteBuffer& decaf::nio::ByteBuffer::put (ByteBuffer & src)`

This method transfers the bytes remaining in the given source buffer into this buffer.

If there are more bytes remaining in the source buffer than in this buffer, that is, if `src.remaining() > remaining()` (p. 588), then no bytes are transferred and a **BufferOverflowException** (p. 609) is thrown.

Otherwise, this method copies `n = src.remaining()` bytes from the given buffer into this buffer, starting at each buffer's current position. The positions of both buffers are then incremented by `n`.

Parameters

<code>src</code>	The buffer to take bytes from an place in this one.
------------------	---

Returns

a reference to this buffer

Exceptions

<i>BufferOverflow-Exception</i> (p. 609)	if there is insufficient space in this buffer for the remaining bytes in the source buffer
<i>IllegalArgumentException</i>	if the source buffer is this buffer
<i>ReadOnlyBuffer-Exception</i> (p. 2244)	if this buffer is read-only

6.97.3.36 ByteBuffer& decaf::nio::ByteBuffer::put (const unsigned char * *buffer*, int *size*, int *offset*, int *length*)

This method transfers bytes into this buffer from the given source array.

If there are more bytes to be copied from the array than remain in this buffer, that is, if `length > remaining()` (p. 588), then no bytes are transferred and a **BufferOverflow-Exception** (p. 609) is thrown.

Otherwise, this method copies `length` bytes from the given array into this buffer, starting at the given `offset` in the array and at the current position of this buffer. The position of this buffer is then incremented by `length`.

Parameters

<i>buffer</i>	The array from which bytes are to be read.
<i>size</i>	The size of the given array.
<i>offset</i>	The offset within the array of the first byte to be read.
<i>length</i>	The number of bytes to be read from the given array.

Returns

a reference to this buffer.

Exceptions

<i>BufferOverflow-Exception</i> (p. 609)	if there is insufficient space in this buffer.
<i>ReadOnlyBuffer-Exception</i> (p. 2244)	if this buffer is read-only.
<i>NullPointerException</i>	if the passed buffer is null.
<i>IndexOutOfBoundsException</i>	if the preconditions of <code>size</code> , <code>offset</code> , or <code>length</code> are not met.

6.97.3.37 ByteBuffer& decaf::nio::ByteBuffer::put (std::vector< unsigned char > & *buffer*)

This method transfers the entire content of the given source byte array into this buffer.

This is the same as calling put(&buffer[0], buffer.size(), 0, buffer.size())

Parameters

<i>buffer</i>	The buffer whose contents are copied to this ByteBuffer (p. 690).
---------------	--

Returns

a reference to this buffer.

Exceptions

BufferOverflow-Exception (p. 609)	if there is insufficient space in this buffer.
ReadOnlyBuffer-Exception (p. 2244)	if this buffer is read-only.

6.97.3.38 virtual ByteBuffer& decaf::nio::ByteBuffer::put (unsigned char *value*) [pure virtual]

Writes the given byte into this buffer at the current position, and then increments the position.

Parameters

<i>value</i>	- the byte value to be written.
--------------	---------------------------------

Returns

a reference to this buffer.

Exceptions

BufferOverflow-Exception (p. 609)	if this buffer's current position is not smaller than its limit.
ReadOnlyBuffer-Exception (p. 2244)	if this buffer is read-only.

Implemented in **decaf::internal::nio::ByteArrayBuffer** (p. 672).

6.97.3.39 `virtual ByteBuffer& decaf::nio::ByteBuffer::put (int index, unsigned char value)` [pure virtual]

Writes the given byte into this buffer at the given index.

Parameters

<i>index</i>	- position in the Buffer (p. 582) to write the data
<i>value</i>	- the byte to write.

Returns

a reference to this buffer.

Exceptions

<i>IndexOutOfBoundsException</i>	if index greater than the buffer's limit minus the size of the type being written, or index is negative.
ReadOnlyBufferException (p. 2244)	if this buffer is read-only.

Implemented in **decaf::internal::nio::ByteArrayBuffer** (p. 672).

6.97.3.40 `virtual ByteBuffer& decaf::nio::ByteBuffer::putChar (char value)` [pure virtual]

Writes one byte containing the given value, into this buffer at the current position, and then increments the position by one.

Parameters

<i>value</i>	The value to be written.
--------------	--------------------------

Returns

a reference to this buffer.

Exceptions

BufferOverflowException (p. 609)	if there are fewer than bytes remaining in this buffer than the size of the data to be written
ReadOnlyBufferException (p. 2244)	if this buffer is read-only

Implemented in **decaf::internal::nio::ByteArrayBuffer** (p. 673).

6.97.3.41 virtual ByteBuffer& decaf::nio::ByteBuffer::putChar (int *index*, char *value*)
[pure virtual]

Writes one byte containing the given value, into this buffer at the given index.

Parameters

<i>index</i>	The position in the Buffer (p. 582) to write the data.
<i>value</i>	The value to write.

Returns

a reference to this buffer.

Exceptions

<i>IndexOutOfBoundsException</i>	if index greater than the buffer's limit minus the size of the type being written, or index is negative.
ReadOnlyBufferException (p. 2244)	if this buffer is read-only

Implemented in **decaf::internal::nio::ByteBuffer** (p. 673).

6.97.3.42 virtual ByteBuffer& decaf::nio::ByteBuffer::putDouble (double *value*)
[pure virtual]

Writes eight bytes containing the given value, into this buffer at the current position, and then increments the position by eight.

Parameters

<i>value</i>	The value to be written.
--------------	--------------------------

Returns

a reference to this buffer.

Exceptions

BufferOverflowException (p. 609)	if there are fewer than bytes remaining in this buffer than the size of the data to be written
ReadOnlyBufferException (p. 2244)	if this buffer is read-only

Implemented in **decaf::internal::nio::ByteBuffer** (p. 674).

6.97.3.43 `virtual ByteBuffer& decaf::nio::ByteBuffer::putDouble (int index, double value)` [pure virtual]

Writes eight bytes containing the given value, into this buffer at the given index.

Parameters

<i>index</i>	The position in the Buffer (p. 582) to write the data
<i>value</i>	The value to write.

Returns

a reference to this buffer.

Exceptions

<i>IndexOutOfBoundsException</i>	if index greater than the buffer's limit minus the size of the type being written, or index is negative.
<i>ReadOnlyBufferException</i> (p. 2244)	if this buffer is read-only.

Implemented in **decaf::internal::nio::ByteArrayBuffer** (p. 674).

6.97.3.44 `virtual ByteBuffer& decaf::nio::ByteBuffer::putFloat (float value)` [pure virtual]

Writes four bytes containing the given value, into this buffer at the current position, and then increments the position by eight.

Parameters

<i>value</i>	The value to be written.
--------------	--------------------------

Returns

a reference to this buffer.

Exceptions

<i>BufferOverflowException</i> (p. 609)	if there are fewer than bytes remaining in this buffer than the size of the data to be written.
<i>ReadOnlyBufferException</i> (p. 2244)	if this buffer is read-only.

Implemented in **decaf::internal::nio::ByteArrayBuffer** (p. 675).

6.97.3.45 virtual ByteBuffer& decaf::nio::ByteBuffer::putFloat (int *index*, float *value*)
[pure virtual]

Writes four bytes containing the given value, into this buffer at the given index.

Parameters

<i>index</i>	The position in the Buffer (p. 582) to write the data
<i>value</i>	The value to write.

Returns

a reference to this buffer.

Exceptions

<i>IndexOutOfBoundsException</i>	if index greater than the buffer's limit minus the size of the type being written, or index is negative.
ReadOnlyBufferException (p. 2244)	if this buffer is read-only.

Implemented in **decaf::internal::nio::ByteBuffer** (p. 675).

6.97.3.46 virtual ByteBuffer& decaf::nio::ByteBuffer::putInt (int *value*) [pure virtual]

Writes four bytes containing the given value, into this buffer at the current position, and then increments the position by eight.

Parameters

<i>value</i>	The value to be written.
--------------	--------------------------

Returns

a reference to this buffer.

Exceptions

BufferOverflowException (p. 609)	if there are fewer than bytes remaining in this buffer than the size of the data to be written
ReadOnlyBufferException (p. 2244)	if this buffer is read-only

Implemented in **decaf::internal::nio::ByteBuffer** (p. 676).

6.97.3.47 `virtual ByteBuffer& decaf::nio::ByteBuffer::putInt (int index, int value)`
`[pure virtual]`

Writes four bytes containing the given value, into this buffer at the given index.

Parameters

<i>index</i>	The position in the Buffer (p. 582) to write the data.
<i>value</i>	The value to write.

Returns

a reference to this buffer

Exceptions

<i>IndexOutOfBoundsException</i>	if index greater than the buffer's limit minus the size of the type being written, or index is negative.
<i>ReadOnlyBufferException</i> (p. 2244)	if this buffer is read-only

Implemented in **decaf::internal::nio::ByteArrayBuffer** (p. 676).

6.97.3.48 `virtual ByteBuffer& decaf::nio::ByteBuffer::putLong (long long value)`
`[pure virtual]`

Writes eight bytes containing the given value, into this buffer at the current position, and then increments the position by eight.

Parameters

<i>value</i>	The value to be written.
--------------	--------------------------

Returns

a reference to this buffer.

Exceptions

<i>BufferOverflowException</i> (p. 609)	if there are fewer than bytes remaining in this buffer than the size of the data to be written.
<i>ReadOnlyBufferException</i> (p. 2244)	if this buffer is read-only.

Implemented in **decaf::internal::nio::ByteArrayBuffer** (p. 677).

6.97.3.49 virtual **ByteBuffer& decaf::nio::ByteBuffer::putLong** (int *index*, long long *value*) [pure virtual]

Writes eight bytes containing the given value, into this buffer at the given index.

Parameters

<i>index</i>	The position in the Buffer (p. 582) to write the data.
<i>value</i>	The value to write.

Returns

a reference to this buffer.

Exceptions

<i>IndexOutOfBoundsException</i>	if index greater than the buffer's limit minus the size of the type being written, or index is negative.
ReadOnlyBufferException (p. 2244)	if this buffer is read-only.

Implemented in **decaf::internal::nio::ByteBuffer** (p. 677).

6.97.3.50 virtual **ByteBuffer& decaf::nio::ByteBuffer::putShort** (short *value*) [pure virtual]

Writes two bytes containing the given value, into this buffer at the current position, and then increments the position by eight.

Parameters

<i>value</i>	The value to be written.
--------------	--------------------------

Returns

a reference to this buffer.

Exceptions

BufferOverflowException (p. 609)	if there are fewer than bytes remaining in this buffer than the size of the data to be written.
ReadOnlyBufferException (p. 2244)	if this buffer is read-only.

Implemented in **decaf::internal::nio::ByteBuffer** (p. 678).

6.97.3.51 `virtual ByteBuffer& decaf::nio::ByteBuffer::putShort (int index, short value)` [pure virtual]

Writes two bytes containing the given value, into this buffer at the given index.

Parameters

<i>index</i>	The position in the Buffer (p. 582) to write the data
<i>value</i>	The value to write.

Returns

a reference to this buffer

Exceptions

<i>IndexOutOfBoundsException</i>	if index greater than the buffer's limit minus the size of the type being written, or index is negative.
ReadOnlyBufferException (p. 2244)	if this buffer is read-only.

Implemented in **decaf::internal::nio::ByteArrayBuffer** (p. 678).

6.97.3.52 `virtual ByteBuffer* decaf::nio::ByteBuffer::slice () const` [pure virtual]

Creates a new byte buffer whose content is a shared subsequence of this buffer's content.

The content of the new buffer will start at this buffer's current position. Changes to this buffer's content will be visible in the new buffer, and vice versa; the two buffers' position, limit, and mark values will be independent.

The new buffer's position will be zero, its capacity and its limit will be the number of bytes remaining in this buffer, and its mark will be undefined. The new buffer will be read-only if, and only if, this buffer is read-only.

Returns

the newly create **ByteBuffer** (p. 690) which the caller owns.

Implemented in **decaf::internal::nio::ByteArrayBuffer** (p. 679).

6.97.3.53 `virtual std::string decaf::nio::ByteBuffer::toString () const` [virtual]

Returns

a std::string describing this object

6.97.3.54 static **ByteBuffer*** **decaf::nio::ByteBuffer::wrap** (unsigned char * *array*, int *size*, int *offset*, int *length*) [static]

Wraps the passed buffer with a new **ByteBuffer** (p. 690).

The new buffer will be backed by the given byte array; that is, modifications to the buffer will cause the array to be modified and vice versa. The new buffer's capacity will be `array.length`, its position will be `offset`, its limit will be `offset + length`, and its mark will be undefined. Its backing array will be the given array, and its array offset will be zero.

Parameters

<i>array</i>	The array that will back the new buffer.
<i>size</i>	The size of the provided array.
<i>offset</i>	The offset of the subarray to be used.
<i>length</i>	The length of the subarray to be used.

Returns

a new **ByteBuffer** (p. 690) that is backed by `buffer`, caller owns.

Exceptions

<i>NullPointerException</i>	if the array passed in is NULL.
<i>IndexOutOfBoundsException</i>	if the preconditions of <code>size</code> , <code>offset</code> , or <code>length</code> are not met.

6.97.3.55 static **ByteBuffer*** **decaf::nio::ByteBuffer::wrap** (std::vector< unsigned char > & *buffer*) [static]

Wraps the passed STL Byte Vector in a **ByteBuffer** (p. 690).

The new buffer will be backed by the given byte array; modifications to the buffer will cause the array to be modified and vice versa. The new buffer's capacity and limit will be `buffer.size()`, its position will be zero, and its mark will be undefined. Its backing array will be the given array, and its array offset will be zero.

Parameters

<i>buffer</i>	The vector that will back the new buffer, the vector must have been sized to the desired size already by calling <code>vector.resize(N)</code> .
---------------	--

Returns

a new **ByteBuffer** (p. 690) that is backed by `buffer`, caller owns.

The documentation for this class was generated from the following file:

- `src/main/decaf/nio/ByteBuffer.h`

6.98 cms::BytesMessage Class Reference

A **BytesMessage** (p.718) object is used to send a message containing a stream of unsigned bytes.

```
#include <src/main/cms/BytesMessage.h>
```

Inheritance diagram for cms::BytesMessage:

Public Member Functions

- virtual **~BytesMessage** () throw ()
- virtual void **setBodyBytes** (const unsigned char *buffer, int numBytes)=0
sets the bytes given to the message body.
- virtual unsigned char * **getBodyBytes** () const =0
Gets the bytes that are contained in this message and returns them in a newly allocated array that becomes the property of the caller.
- virtual int **getBodyLength** () const =0
Returns the number of bytes contained in the body of this message.
- virtual void **reset** ()=0
Puts the message body in read-only mode and repositions the stream of bytes to the beginning.
- virtual bool **readBoolean** () const =0
Reads a Boolean from the Bytes message stream.
- virtual void **writeBoolean** (bool value)=0
Writes a boolean to the bytes message stream as a 1-byte value.
- virtual unsigned char **readByte** () const =0
Reads a Byte from the Bytes message stream.
- virtual void **writeByte** (unsigned char value)=0
Writes a byte to the bytes message stream as a 1-byte value.
- virtual int **readBytes** (std::vector< unsigned char > &value) const =0
Reads a byte array from the bytes message stream.
- virtual void **writeBytes** (const std::vector< unsigned char > &value)=0
Writes a byte array to the bytes message stream using the vector size as the number of bytes to write.
- virtual int **readBytes** (unsigned char *buffer, int length) const =0
Reads a portion of the bytes message stream.
- virtual void **writeBytes** (const unsigned char *value, int offset, int length)=0
Writes a portion of a byte array to the bytes message stream.
- virtual char **readChar** () const =0
Reads a Char from the Bytes message stream.
- virtual void **writeChar** (char value)=0
Writes a char to the bytes message stream as a 1-byte value.

- virtual float **readFloat** () const =0
Reads a 32 bit float from the Bytes message stream.
- virtual void **writeFloat** (float value)=0
Writes a float to the bytes message stream as a 4 byte value.
- virtual double **readDouble** () const =0
Reads a 64 bit double from the Bytes message stream.
- virtual void **writeDouble** (double value)=0
Writes a double to the bytes message stream as a 8 byte value.
- virtual short **readShort** () const =0
Reads a 16 bit signed short from the Bytes message stream.
- virtual void **writeShort** (short value)=0
Writes a signed short to the bytes message stream as a 2 byte value.
- virtual unsigned short **readUnsignedShort** () const =0
Reads a 16 bit unsigned short from the Bytes message stream.
- virtual void **writeUnsignedShort** (unsigned short value)=0
Writes a unsigned short to the bytes message stream as a 2 byte value.
- virtual int **readInt** () const =0
Reads a 32 bit signed integer from the Bytes message stream.
- virtual void **writeInt** (int value)=0
Writes a signed int to the bytes message stream as a 4 byte value.
- virtual long long **readLong** () const =0
Reads a 64 bit long from the Bytes message stream.
- virtual void **writeLong** (long long value)=0
Writes a long long to the bytes message stream as a 8 byte value.
- virtual std::string **readString** () const =0
Reads an ASCII String from the Bytes message stream.
- virtual void **writeString** (const std::string &value)=0
Writes an ASCII String to the Bytes message stream.
- virtual std::string **readUTF** () const =0
*Reads an UTF String from the **BytesMessage** (p. 718) stream.*
- virtual void **writeUTF** (const std::string &value)=0
*Writes an UTF String to the **BytesMessage** (p. 718) stream.*
- virtual **BytesMessage** * **clone** () const =0
Clones this message.

6.98.1 Detailed Description

A **BytesMessage** (p. 718) object is used to send a message containing a stream of unsigned bytes.

It inherits from the **Message** (p. 1839) interface and adds a bytes message body. -
The receiver of the message supplies the interpretation of the bytes using the methods added by the **BytesMessage** (p. 718) interface.

The **BytesMessage** (p. 718) methods are based largely on those found in **decaf.io.DataInputStream** (p. 1094) and **decaf.io.DataOutputStream** (p. 1109).

Although the CMS API allows the use of message properties with byte messages, they are typically not used, since the inclusion of properties may affect the format.

The primitive types can be written explicitly using methods for each type. Because the C++ language is more limited when dealing with primitive types the JMS equivalent generic read and write methods that take Java objects cannot be provided in the CMS API.

When the message is first created, and when `clearBody` is called, the body of the message is in write-only mode. After the first call to `reset` has been made, the message body is in read-only mode. After a message has been sent, the client that sent it can retain and modify it without affecting the message that has been sent. The same message object can be sent multiple times. When a message has been received, the provider has called `reset` so that the message body is in read-only mode for the client.

If `clearBody` is called on a message in read-only mode, the message body is cleared and the message is in write-only mode.

If a client attempts to read a message in write-only mode, a **MessageNotReadableException** (p. 1922) is thrown.

If a client attempts to write a message in read-only mode, a **MessageNotWriteableException** (p. 1923) is thrown.

Since

1.0

6.98.2 Constructor & Destructor Documentation

6.98.2.1 `virtual cms::BytesMessage::~BytesMessage () throw () [virtual]`

6.98.3 Member Function Documentation

6.98.3.1 `virtual BytesMessage* cms::BytesMessage::clone () const [pure virtual]`

Clones this message.

Returns

a deep copy of this message.

Exceptions

CMSException (p. 826)	- if an internal error occurs while cloning the Message (p. 1839).
---------------------------------	---

Implements **cms::Message** (p. 1844).

Implemented in **activemq::commands::ActiveMQBytesMessage** (p. 168).

6.98.3.2 `virtual unsigned char* cms::BytesMessage::getBodyBytes () const` [pure virtual]

Gets the bytes that are contained in this message and returns them in a newly allocated array that becomes the property of the caller.

This is a copy of the data contained in this message, changing the value contained in this array has no effect on the data contained in this message.

Returns

pointer to a byte buffer that the call owns upon completion of this method.

Exceptions

<i>CMSEException</i> (p. 826)	- If an internal error occurs.
<i>MessageNotReadableException</i> (p. 1922)	- If the message is in Write Only Mode.

Implemented in **activemq::commands::ActiveMQBytesMessage** (p. 170).

6.98.3.3 `virtual int cms::BytesMessage::getBodyLength () const` [pure virtual]

Returns the number of bytes contained in the body of this message.

Returns

number of bytes.

Exceptions

<i>CMSEException</i> (p. 826)	- If an internal error occurs.
<i>MessageNotReadableException</i> (p. 1922)	- If the message is in Write Only Mode.

Implemented in **activemq::commands::ActiveMQBytesMessage** (p. 170).

6.98.3.4 `virtual bool cms::BytesMessage::readBoolean () const` [pure virtual]

Reads a Boolean from the Bytes message stream.

Returns

boolean value from stream

Exceptions

<i>CMSException</i> (p. 826)	- if the CMS provider fails to read the message due to some internal error.
<i>MessageEOF-Exception</i> (p. 1905)	- if unexpected end of bytes stream has been reached.
<i>MessageNot-ReadableException</i> (p. 1922)	- if the message is in write-only mode.

Implemented in `activemq::commands::ActiveMQBytesMessage` (p. 171).

6.98.3.5 `virtual unsigned char cms::BytesMessage::readByte () const` [pure virtual]

Reads a Byte from the Bytes message stream.

Returns

unsigned char value from stream

Exceptions

<i>CMSException</i> (p. 826)	- if the CMS provider fails to read the message due to some internal error.
<i>MessageEOF-Exception</i> (p. 1905)	- if unexpected end of bytes stream has been reached.
<i>MessageNot-ReadableException</i> (p. 1922)	- if the message is in write-only mode.

Implemented in `activemq::commands::ActiveMQBytesMessage` (p. 171).

6.98.3.6 `virtual int cms::BytesMessage::readBytes (std::vector< unsigned char > & value) const [pure virtual]`

Reads a byte array from the bytes message stream.

If the length of vector value is less than the number of bytes remaining to be read from the stream, the vector should be filled. A subsequent call reads the next increment, and so on.

If the number of bytes remaining in the stream is less than the length of vector value, the bytes should be read into the vector. The return value of the total number of bytes read will be less than the length of the vector, indicating that there are no more bytes left to be read from the stream. The next read of the stream returns -1.

Parameters

<i>value</i>	buffer to place data in
--------------	-------------------------

Returns

the total number of bytes read into the buffer, or -1 if there is no more data because the end of the stream has been reached

Exceptions

<i>CMSException</i> (p. 826)	- if the CMS provider fails to read the message due to some internal error.
<i>MessageEOF-Exception</i> (p. 1905)	- if unexpected end of bytes stream has been reached.
<i>MessageNot-ReadableException</i> (p. 1922)	- if the message is in write-only mode.

Implemented in **activemq::commands::ActiveMQBytesMessage** (p. 172).

6.98.3.7 `virtual int cms::BytesMessage::readBytes (unsigned char * buffer, int length) const [pure virtual]`

Reads a portion of the bytes message stream.

If the length of array value is less than the number of bytes remaining to be read from the stream, the array should be filled. A subsequent call reads the next increment, and so on.

If the number of bytes remaining in the stream is less than the length of array value, the bytes should be read into the array. The return value of the total number of bytes read will be less than the length of the array, indicating that there are no more bytes left to be read from the stream. The next read of the stream returns -1.

If length is negative, or length is greater than the length of the array value, then an -

`IndexOutOfBoundsException` is thrown. No bytes will be read from the stream for this exception case.

Parameters

<i>buffer</i>	the buffer into which the data is read
<i>length</i>	the number of bytes to read; must be less than or equal to <code>value.length</code>

Returns

the total number of bytes read into the buffer, or -1 if there is no more data because the end of the stream has been reached

Exceptions

<i>CMSException</i> (p. 826)	- if the CMS provider fails to read the message due to some internal error.
<i>MessageEOF-Exception</i> (p. 1905)	- if unexpected end of bytes stream has been reached.
<i>MessageNot-ReadableException</i> (p. 1922)	- if the message is in write-only mode.

Implemented in `activemq::commands::ActiveMQBytesMessage` (p. 172).

6.98.3.8 `virtual char cms::BytesMessage::readChar() const` [pure virtual]

Reads a Char from the Bytes message stream.

Returns

char value from stream

Exceptions

<i>CMSException</i> (p. 826)	- if the CMS provider fails to read the message due to some internal error.
<i>MessageEOF-Exception</i> (p. 1905)	- if unexpected end of bytes stream has been reached.
<i>MessageNot-ReadableException</i> (p. 1922)	- if the message is in write-only mode.

Implemented in `activemq::commands::ActiveMQBytesMessage` (p. 173).

6.98.3.9 virtual double cms::BytesMessage::readDouble () const [pure virtual]

Reads a 64 bit double from the Bytes message stream.

Returns

double value from stream

Exceptions

CMSException (p. 826)	- if the CMS provider fails to read the message due to some internal error.
MessageEOF-Exception (p. 1905)	- if unexpected end of bytes stream has been reached.
MessageNot-ReadableException (p. 1922)	- if the message is in write-only mode.

Implemented in **activemq::commands::ActiveMQBytesMessage** (p. 173).

6.98.3.10 virtual float cms::BytesMessage::readFloat () const [pure virtual]

Reads a 32 bit float from the Bytes message stream.

Returns

double value from stream

Exceptions

CMSException (p. 826)	- if the CMS provider fails to read the message due to some internal error.
MessageEOF-Exception (p. 1905)	- if unexpected end of bytes stream has been reached.
MessageNot-ReadableException (p. 1922)	- if the message is in write-only mode.

Implemented in **activemq::commands::ActiveMQBytesMessage** (p. 174).

6.98.3.11 virtual int cms::BytesMessage::readInt () const [pure virtual]

Reads a 32 bit signed integer from the Bytes message stream.

Returns

int value from stream

Exceptions

<i>CMSException</i> (p. 826)	- if the CMS provider fails to read the message due to some internal error.
<i>MessageEOF-Exception</i> (p. 1905)	- if unexpected end of bytes stream has been reached.
<i>MessageNot-ReadableException</i> (p. 1922)	- if the message is in write-only mode.

Implemented in **activemq::commands::ActiveMQBytesMessage** (p. 174).

6.98.3.12 `virtual long long cms::BytesMessage::readLong () const` [pure virtual]

Reads a 64 bit long from the Bytes message stream.

Returns

long long value from stream

Exceptions

<i>CMSException</i> (p. 826)	- if the CMS provider fails to read the message due to some internal error.
<i>MessageEOF-Exception</i> (p. 1905)	- if unexpected end of bytes stream has been reached.
<i>MessageNot-ReadableException</i> (p. 1922)	- if the message is in write-only mode.

Implemented in **activemq::commands::ActiveMQBytesMessage** (p. 175).

6.98.3.13 `virtual short cms::BytesMessage::readShort () const` [pure virtual]

Reads a 16 bit signed short from the Bytes message stream.

Returns

short value from stream

Exceptions

<i>CMSException</i> (p. 826)	- if the CMS provider fails to read the message due to some internal error.
<i>MessageEOF-Exception</i> (p. 1905)	- if unexpected end of bytes stream has been reached.
<i>MessageNot-ReadableException</i> (p. 1922)	- if the message is in write-only mode.

Implemented in **activemq::commands::ActiveMQBytesMessage** (p. 175).

6.98.3.14 `virtual std::string cms::BytesMessage::readString () const` [pure virtual]

Reads an ASCII String from the Bytes message stream.

Returns

String from stream

Exceptions

<i>CMSException</i> (p. 826)	- if the CMS provider fails to read the message due to some internal error.
<i>MessageEOF-Exception</i> (p. 1905)	- if unexpected end of bytes stream has been reached.
<i>MessageNot-ReadableException</i> (p. 1922)	- if the message is in write-only mode.

Implemented in **activemq::commands::ActiveMQBytesMessage** (p. 176).

6.98.3.15 `virtual unsigned short cms::BytesMessage::readUnsignedShort () const` [pure virtual]

Reads a 16 bit unsigned short from the Bytes message stream.

Returns

unsigned short value from stream

Exceptions

<i>CMSException</i> (p. 826)	- if the CMS provider fails to read the message due to some internal error.
--	---

<i>MessageEOF-Exception</i> (p. 1905)	- if unexpected end of bytes stream has been reached.
<i>MessageNot-ReadableException</i> (p. 1922)	- if the message is in write-only mode.

Implemented in **activemq::commands::ActiveMQBytesMessage** (p. 176).

6.98.3.16 `virtual std::string cms::BytesMessage::readUTF () const` [pure virtual]

Reads an UTF String from the **BytesMessage** (p. 718) stream.

Returns

String from stream

Exceptions

<i>CMSException</i> (p. 826)	- if the CMS provider fails to read the message due to some internal error.
<i>MessageEOF-Exception</i> (p. 1905)	- if unexpected end of bytes stream has been reached.
<i>MessageNot-ReadableException</i> (p. 1922)	- if the message is in write-only mode.

Implemented in **activemq::commands::ActiveMQBytesMessage** (p. 176).

6.98.3.17 `virtual void cms::BytesMessage::reset ()` [pure virtual]

Puts the message body in read-only mode and repositions the stream of bytes to the beginning.

Exceptions

<i>CMSException</i> (p. 826)	- If the provider fails to perform the reset operation.
<i>MessageFormatException</i> (p. 1906)	- If the Message (p. 1839) has an invalid format.

Implemented in **activemq::commands::ActiveMQBytesMessage** (p. 177).

6.98.3.18 virtual void cms::BytesMessage::setBodyBytes (const unsigned char * *buffer*, int *numBytes*) [pure virtual]

sets the bytes given to the message body.

Parameters

<i>buffer</i>	Byte Buffer to copy
<i>numBytes</i>	Number of bytes in Buffer to copy

Exceptions

CMSException (p. 826)	- If an internal error occurs.
MessageNot-WriteableException (p. 1923)	- if in Read Only Mode.

Implemented in **activemq::commands::ActiveMQBytesMessage** (p. 177).

6.98.3.19 virtual void cms::BytesMessage::writeBoolean (bool *value*) [pure virtual]

Writes a boolean to the bytes message stream as a 1-byte value.

The value true is written as the value (byte)1; the value false is written as the value (byte)0.

Parameters

<i>value</i>	boolean to write to the stream
--------------	--------------------------------

Exceptions

CMSException (p. 826)	- if the CMS provider fails to write the message due to some internal error.
MessageNot-WriteableException (p. 1923)	- if the message is in read-only mode.

Implemented in **activemq::commands::ActiveMQBytesMessage** (p. 178).

6.98.3.20 virtual void cms::BytesMessage::writeByte (unsigned char *value*) [pure virtual]

Writes a byte to the bytes message stream as a 1-byte value.

Parameters

<i>value</i>	byte to write to the stream
--------------	-----------------------------

Exceptions

<i>CMSException</i> (p. 826)	- if the CMS provider fails to write the message due to some internal error.
<i>MessageNot-WriteableException</i> (p. 1923)	- if the message is in read-only mode.

Implemented in **activemq::commands::ActiveMQBytesMessage** (p. 178).

6.98.3.21 `virtual void cms::BytesMessage::writeBytes (const std::vector< unsigned char > & value) [pure virtual]`

Writes a byte array to the bytes message stream using the vector size as the number of bytes to write.

Parameters

<i>value</i>	bytes to write to the stream
--------------	------------------------------

Exceptions

<i>CMSException</i> (p. 826)	- if the CMS provider fails to write the message due to some internal error.
<i>MessageNot-WriteableException</i> (p. 1923)	- if the message is in read-only mode.

Implemented in **activemq::commands::ActiveMQBytesMessage** (p. 179).

6.98.3.22 `virtual void cms::BytesMessage::writeBytes (const unsigned char * value, int offset, int length) [pure virtual]`

Writes a portion of a byte array to the bytes message stream.
size as the number of bytes to write.

Parameters

<i>value</i>	bytes to write to the stream
<i>offset</i>	the initial offset within the byte array
<i>length</i>	the number of bytes to use

Exceptions

<i>CMSEException</i> (p. 826)	- if the CMS provider fails to write the message due to some internal error.
<i>MessageNotWriteableException</i> (p. 1923)	- if the message is in read-only mode.

Implemented in **activemq::commands::ActiveMQBytesMessage** (p. 179).

6.98.3.23 virtual void cms::BytesMessage::writeChar (char *value*) [pure virtual]

Writes a char to the bytes message stream as a 1-byte value.

Parameters

<i>value</i>	char to write to the stream
--------------	-----------------------------

Exceptions

<i>CMSEException</i> (p. 826)	- if the CMS provider fails to write the message due to some internal error.
<i>MessageNotWriteableException</i> (p. 1923)	- if the message is in read-only mode.

Implemented in **activemq::commands::ActiveMQBytesMessage** (p. 179).

6.98.3.24 virtual void cms::BytesMessage::writeDouble (double *value*) [pure virtual]

Writes a double to the bytes message stream as a 8 byte value.

Parameters

<i>value</i>	double to write to the stream
--------------	-------------------------------

Exceptions

<i>CMSEException</i> (p. 826)	- if the CMS provider fails to write the message due to some internal error.
<i>MessageNotWriteableException</i> (p. 1923)	- if the message is in read-only mode.

Implemented in **activemq::commands::ActiveMQBytesMessage** (p. 180).

6.98.3.25 `virtual void cms::BytesMessage::writeFloat (float value)` [pure virtual]

Writes a float to the bytes message stream as a 4 byte value.

Parameters

<i>value</i>	float to write to the stream
--------------	------------------------------

Exceptions

<i>CMSException</i> (p. 826)	- if the CMS provider fails to write the message due to some internal error.
<i>MessageNot-WriteableException</i> (p. 1923)	- if the message is in read-only mode.

Implemented in `activemq::commands::ActiveMQBytesMessage` (p. 180).

6.98.3.26 `virtual void cms::BytesMessage::writeInt (int value)` [pure virtual]

Writes a signed int to the bytes message stream as a 4 byte value.

Parameters

<i>value</i>	signed int to write to the stream
--------------	-----------------------------------

Exceptions

<i>CMSException</i> (p. 826)	- if the CMS provider fails to write the message due to some internal error.
<i>MessageNot-WriteableException</i> (p. 1923)	- if the message is in read-only mode.

Implemented in `activemq::commands::ActiveMQBytesMessage` (p. 181).

6.98.3.27 `virtual void cms::BytesMessage::writeLong (long long value)` [pure virtual]

Writes a long long to the bytes message stream as a 8 byte value.

Parameters

<i>value</i>	signed long long to write to the stream
--------------	---

Exceptions

<i>CMSEException</i> (p. 826)	- if the CMS provider fails to write the message due to some internal error.
<i>MessageNotWriteableException</i> (p. 1923)	- if the message is in read-only mode.

Implemented in **activemq::commands::ActiveMQBytesMessage** (p. 181).

6.98.3.28 virtual void cms::BytesMessage::writeShort (short *value*) [pure virtual]

Writes a signed short to the bytes message stream as a 2 byte value.

Parameters

<i>value</i>	signed short to write to the stream
--------------	-------------------------------------

Exceptions

<i>CMSEException</i> (p. 826)	- if the CMS provider fails to write the message due to some internal error.
<i>MessageNotWriteableException</i> (p. 1923)	- if the message is in read-only mode.

Implemented in **activemq::commands::ActiveMQBytesMessage** (p. 181).

6.98.3.29 virtual void cms::BytesMessage::writeString (const std::string & *value*) [pure virtual]

Writes an ASCII String to the Bytes message stream.

Parameters

<i>value</i>	String to write to the stream
--------------	-------------------------------

Exceptions

<i>CMSEException</i> (p. 826)	- if the CMS provider fails to write the message due to some internal error.
<i>MessageNotWriteableException</i> (p. 1923)	- if the message is in read-only mode.

Implemented in **activemq::commands::ActiveMQBytesMessage** (p. 182).

6.98.3.30 `virtual void cms::BytesMessage::writeUnsignedShort (unsigned short value)` [pure virtual]

Writes a unsigned short to the bytes message stream as a 2 byte value.

Parameters

<i>value</i>	unsigned short to write to the stream
--------------	---------------------------------------

Exceptions

<i>CMSEException</i> (p. 826)	- if the CMS provider fails to write the message due to some internal error.
<i>MessageNot-WriteableException</i> (p. 1923)	- if the message is in read-only mode.

Implemented in **activemq::commands::ActiveMQBytesMessage** (p. 182).

6.98.3.31 `virtual void cms::BytesMessage::writeUTF (const std::string & value)` [pure virtual]

Writes an UTF String to the **BytesMessage** (p. 718) stream.

Parameters

<i>value</i>	String to write to the stream
--------------	-------------------------------

Exceptions

<i>CMSEException</i> (p. 826)	- if the CMS provider fails to read the message due to some internal error.
<i>MessageNot-WriteableException</i> (p. 1923)	- if the message is in read-only mode.

Implemented in **activemq::commands::ActiveMQBytesMessage** (p. 183).

The documentation for this class was generated from the following file:

- `src/main/cms/BytesMessage.h`

6.99 activemq::cmsutil::CachedConsumer Class Reference

A cached message consumer contained within a pooled session.

```
#include <src/main/activemq/cmsutil/CachedConsumer.h>
```

Inheritance diagram for activemq::cmsutil::CachedConsumer:

Public Member Functions

- **CachedConsumer** (**cms::MessageConsumer** *consumer)
- virtual **~CachedConsumer** () throw ()
- virtual void **close** ()
Does nothing - the real producer resource will be closed by the lifecycle manager.
- virtual void **start** ()
Starts the service.
- virtual void **stop** ()
Stops this service.
- virtual **cms::Message** * **receive** ()
Synchronously Receive a Message.
- virtual **cms::Message** * **receive** (int millisecs)
Synchronously Receive a Message, time out after defined interval.
- virtual **cms::Message** * **receiveNoWait** ()
Receive a Message, does not wait if there isn't a new message to read, returns NULL if nothing read.
- virtual void **setMessageListener** (**cms::MessageListener** *listener)
Sets the MessageListener that this class will send notifs on.
- virtual **cms::MessageListener** * **getMessageListener** () const
Gets the MessageListener that this class will send new Message notification events to.
- virtual std::string **getMessageSelector** () const
Gets this message consumer's message selector expression.

6.99.1 Detailed Description

A cached message consumer contained within a pooled session.

6.99.2 Constructor & Destructor Documentation

6.99.2.1 **activemq::cmsutil::CachedConsumer::CachedConsumer** (**cms::MessageConsumer** * consumer) [inline]

6.99.2.2 **virtual activemq::cmsutil::CachedConsumer::~~CachedConsumer** () throw () [inline, virtual]

6.99.3 Member Function Documentation

6.99.3.1 `virtual void activemq::cmsutil::CachedConsumer::close () [inline, virtual]`

Does nothing - the real producer resource will be closed by the lifecycle manager.

Implements **cms::Closeable** (p. 816).

6.99.3.2 `virtual cms::MessageListener* activemq::cmsutil::CachedConsumer::getMessageListener () const [inline, virtual]`

Gets the MessageListener that this class will send new Message notification events to.

Returns

The listener of messages received by this consumer

Exceptions

<i>CMSEException</i> - If an internal error occurs.

Implements **cms::MessageConsumer** (p. 1878).

6.99.3.3 `virtual std::string activemq::cmsutil::CachedConsumer::getMessageSelector () const [inline, virtual]`

Gets this message consumer's message selector expression.

Returns

This Consumer's selector expression or "".

Exceptions

<i>CMSEException</i> - If an internal error occurs.

Implements **cms::MessageConsumer** (p. 1878).

6.99.3.4 `virtual cms::Message* activemq::cmsutil::CachedConsumer::receive () [inline, virtual]`

Synchronously Receive a Message.

Returns

new message which the caller owns and must delete.

Exceptions

<i>CMSEException</i>	- If an internal error occurs.
----------------------	--------------------------------

Implements **cms::MessageConsumer** (p. 1879).

6.99.3.5 virtual **cms::Message*** **activemq::cmsutil::CachedConsumer::receive** (int *millisecs*) [inline, virtual]

Synchronously Receive a Message, time out after defined interval.

Returns null if nothing read.

Returns

new message which the caller owns and must delete.

Exceptions

<i>CMSEException</i>	- If an internal error occurs.
----------------------	--------------------------------

Implements **cms::MessageConsumer** (p. 1879).

6.99.3.6 virtual **cms::Message*** **activemq::cmsutil::CachedConsumer::receiveNoWait** () [inline, virtual]

Receive a Message, does not wait if there isn't a new message to read, returns NULL if nothing read.

Returns

new message which the caller owns and must delete.

Exceptions

<i>CMSEException</i>	- If an internal error occurs.
----------------------	--------------------------------

Implements **cms::MessageConsumer** (p. 1880).

6.99.3.7 virtual void **activemq::cmsutil::CachedConsumer::setMessageListener** (**cms::MessageListener** * *listener*) [inline, virtual]

Sets the MessageListener that this class will send notifs on.

Parameters

<i>listener</i>	The listener of messages received by this consumer.
-----------------	---

Exceptions

<i>CMSEException</i> - If an internal error occurs.

Implements **cms::MessageConsumer** (p. 1880).

6.99.3.8 `virtual void activemq::cmsutil::CachedConsumer::start () [inline, virtual]`

Starts the service.

Exceptions

<i>CMSEException</i> if an internal error occurs while starting.
--

Implements **cms::Startable** (p. 2534).

6.99.3.9 `virtual void activemq::cmsutil::CachedConsumer::stop () [inline, virtual]`

Stops this service.

Exceptions

<i>CMSEException</i> - if an internal error occurs while stopping the Service.
--

Implements **cms::Stoppable** (p. 2602).

The documentation for this class was generated from the following file:

- `src/main/activemq/cmsutil/CachedConsumer.h`

6.100 activemq::cmsutil::CachedProducer Class Reference

A cached message producer contained within a pooled session.

```
#include <src/main/activemq/cmsutil/CachedProducer.h>
```

Inheritance diagram for `activemq::cmsutil::CachedProducer`:

Public Member Functions

- **CachedProducer** (**cms::MessageProducer** *producer)
- virtual **~CachedProducer** () throw ()
- virtual void **close** ()

Does nothing - the real producer resource will be closed by the lifecycle manager.

- virtual void **send** (cms::Message *message)

Sends the message to the default producer destination, but does not take ownership of the message, caller must still destroy it.

- virtual void **send** (cms::Message *message, int deliveryMode, int priority, long long timeToLive)

Sends the message to the default producer destination, but does not take ownership of the message, caller must still destroy it.

- virtual void **send** (const cms::Destination *destination, cms::Message *message)

Sends the message to the designated destination, but does not take ownership of the message, caller must still destroy it.

- virtual void **send** (const cms::Destination *destination, cms::Message *message, int deliveryMode, int priority, long long timeToLive)

Sends the message to the designated destination, but does not take ownership of the message, caller must still destroy it.

- virtual void **setDeliveryMode** (int mode)

Sets the delivery mode for this Producer.

- virtual int **getDeliveryMode** () const

Gets the delivery mode for this Producer.

- virtual void **setDisableMessageID** (bool value)

Sets if Message Ids are disabled for this Producer.

- virtual bool **getDisableMessageID** () const

Gets if Message Ids are disabled for this Producer.

- virtual void **setDisableMessageTimeStamp** (bool value)

Sets if Message Time Stamps are disabled for this Producer.

- virtual bool **getDisableMessageTimeStamp** () const

Gets if Message Time Stamps are disabled for this Producer.

- virtual void **setPriority** (int priority)

Sets the Priority that this Producers sends messages at.

- virtual int **getPriority** () const

Gets the Priority level that this producer sends messages at.

- virtual void **setTimeToLive** (long long time)

Sets the Time to Live that this Producers sends messages with.

- virtual long long **getTimeToLive** () const

Gets the Time to Live that this producer sends messages with.

6.100.1 Detailed Description

A cached message producer contained within a pooled session.

6.100.2 Constructor & Destructor Documentation

6.100.2.1 `activemq::cmsutil::CachedProducer::CachedProducer (cms::MessageProducer * producer) [inline]`

6.100.2.2 `virtual activemq::cmsutil::CachedProducer::~~CachedProducer () throw () [inline, virtual]`

6.100.3 Member Function Documentation

6.100.3.1 `virtual void activemq::cmsutil::CachedProducer::close () [inline, virtual]`

Does nothing - the real producer resource will be closed by the lifecycle manager.

Implements **cms::Closeable** (p. 816).

6.100.3.2 `virtual int activemq::cmsutil::CachedProducer::getDeliveryMode () const [inline, virtual]`

Gets the delivery mode for this Producer.

Returns

The DeliveryMode

Exceptions

<i>CMSEException</i> - if an internal error occurs.

Implements **cms::MessageProducer** (p. 1926).

6.100.3.3 `virtual bool activemq::cmsutil::CachedProducer::getDisableMessageID () const [inline, virtual]`

Gets if Message Ids are disabled for this Producer.

Returns

boolean indicating enable / disable (true / false)

Exceptions

<i>CMSEException</i> - if an internal error occurs.

Implements **cms::MessageProducer** (p. 1927).

6.100.3.4 `virtual bool activemq::cmsutil::CachedProducer::get-DisableMessageTimeStamp () const [inline, virtual]`

Gets if Message Time Stamps are disabled for this Producer.

Returns

boolean indicating enable / disable (true / false)

Exceptions

<i>CMSEException</i>	- if an internal error occurs.
----------------------	--------------------------------

Implements **cms::MessageProducer** (p. 1927).

6.100.3.5 `virtual int activemq::cmsutil::CachedProducer::getPriority () const [inline, virtual]`

Gets the Priority level that this producer sends messages at.

Returns

int based priority level

Exceptions

<i>CMSEException</i>	- if an internal error occurs.
----------------------	--------------------------------

Implements **cms::MessageProducer** (p. 1927).

6.100.3.6 `virtual long long activemq::cmsutil::CachedProducer::getTimeToLive () const [inline, virtual]`

Gets the Time to Live that this producer sends messages with.

Returns

Time to live value in milliseconds

Exceptions

<i>CMSEException</i>	- if an internal error occurs.
----------------------	--------------------------------

Implements **cms::MessageProducer** (p. 1928).

6.100.3.7 `virtual void activemq::cmsutil::CachedProducer::send (cms::Message * message) [inline, virtual]`

Sends the message to the default producer destination, but does not take ownership of the message, caller must still destroy it.

Uses default values for deliveryMode, priority, and time to live.

Parameters

<i>message</i>	The message to be sent.
----------------	-------------------------

Exceptions

<i>CMSException</i>	- if an internal error occurs while sending the message.
<i>MessageFormatException</i>	- if an Invalid Message is given.
<i>InvalidDestinationException</i>	- if a client uses this method with a MessageProducer with an invalid destination.
<i>UnsupportedOperationException</i>	- if a client uses this method with a MessageProducer that did not specify a destination at creation time.

Implements **cms::MessageProducer** (p. 1928).

6.100.3.8 `virtual void activemq::cmsutil::CachedProducer::send (cms::Message * message, int deliveryMode, int priority, long long timeToLive) [inline, virtual]`

Sends the message to the default producer destination, but does not take ownership of the message, caller must still destroy it.

Parameters

<i>message</i>	The message to be sent.
<i>delivery-Mode</i>	The delivery mode to be used.
<i>priority</i>	The priority for this message.
<i>timeToLive</i>	The time to live value for this message in milliseconds.

Exceptions

<i>CMSException</i>	- if an internal error occurs while sending the message.
<i>MessageFormatException</i>	- if an Invalid Message is given.
<i>InvalidDestinationException</i>	- if a client uses this method with a MessageProducer with an invalid destination.
<i>UnsupportedOperationException</i>	- if a client uses this method with a MessageProducer that did not specify a destination at creation time.

Implements **cms::MessageProducer** (p. 1929).

6.100.3.9 `virtual void activemq::cmsutil::CachedProducer::send (const
cms::Destination * destination, cms::Message * message) [inline,
virtual]`

Sends the message to the designated destination, but does not take ownership of the message, caller must still destroy it.

Uses default values for deliveryMode, priority, and time to live.

Parameters

<i>destination</i>	The destination on which to send the message
<i>message</i>	the message to be sent.

Exceptions

<i>CMSEException</i>	- if an internal error occurs while sending the message.
<i>MessageFormat-Exception</i>	- if an Invalid Message is given.
<i>InvalidDestination-Exception</i>	- if a client uses this method with a MessageProducer with an invalid destination.
<i>Unsupported-OperationException</i>	- if a client uses this method with a MessageProducer that did not specify a destination at creation time.

Implements **cms::MessageProducer** (p. 1929).

6.100.3.10 `virtual void activemq::cmsutil::CachedProducer::send (const
cms::Destination * destination, cms::Message * message, int deliveryMode,
int priority, long long timeToLive) [inline, virtual]`

Sends the message to the designated destination, but does not take ownership of the message, caller must still destroy it.

Parameters

<i>destination</i>	The destination on which to send the message
<i>message</i>	The message to be sent.
<i>delivery-Mode</i>	The delivery mode to be used.
<i>priority</i>	The priority for this message.
<i>timeToLive</i>	The time to live value for this message in milliseconds.

Exceptions

<i>CMSEException</i>	- if an internal error occurs while sending the message.
<i>MessageFormat-Exception</i>	- if an Invalid Message is given.

<i>InvalidDestinationException</i>	- if a client uses this method with a MessageProducer with an invalid destination.
<i>UnsupportedOperationException</i>	- if a client uses this method with a MessageProducer that did not specify a destination at creation time.

Implements **cms::MessageProducer** (p. 1930).

6.100.3.11 **virtual void activemq::cmsutil::CachedProducer::setDeliveryMode (int mode)** [inline, virtual]

Sets the delivery mode for this Producer.

Parameters

<i>mode</i>	The DeliveryMode
-------------	------------------

Exceptions

<i>CMSEException</i>	- if an internal error occurs.
----------------------	--------------------------------

Implements **cms::MessageProducer** (p. 1931).

6.100.3.12 **virtual void activemq::cmsutil::CachedProducer::setDisableMessageID (bool value)** [inline, virtual]

Sets if Message Ids are disabled for this Producer.

Parameters

<i>value</i>	boolean indicating enable / disable (true / false)
--------------	--

Exceptions

<i>CMSEException</i>	- if an internal error occurs.
----------------------	--------------------------------

Implements **cms::MessageProducer** (p. 1931).

6.100.3.13 **virtual void activemq::cmsutil::CachedProducer::setDisableMessageTimeStamp (bool value)** [inline, virtual]

Sets if Message Time Stamps are disabled for this Producer.

Parameters

<i>value</i>	- boolean indicating enable / disable (true / false)
--------------	--

Exceptions

<i>CMSEException</i>	- if an internal error occurs.
----------------------	--------------------------------

Implements **cms::MessageProducer** (p. 1931).

6.100.3.14 `virtual void activemq::cmsutil::CachedProducer::setPriority (int priority)
[inline, virtual]`

Sets the Priority that this Producers sends messages at.

Parameters

<i>priority</i>	int value for Priority level
-----------------	------------------------------

Exceptions

<i>CMSEException</i>	- if an internal error occurs.
----------------------	--------------------------------

Implements **cms::MessageProducer** (p. 1932).

6.100.3.15 `virtual void activemq::cmsutil::CachedProducer::setTimeToLive (long
long time) [inline, virtual]`

Sets the Time to Live that this Producers sends messages with.

This value will be used if the time to live is not specified via the send method.

Parameters

<i>time</i>	default time to live value in milliseconds
-------------	--

Exceptions

<i>CMSEException</i>	- if an internal error occurs.
----------------------	--------------------------------

Implements **cms::MessageProducer** (p. 1932).

The documentation for this class was generated from the following file:

- src/main/activemq/cmsutil/**CachedProducer.h**

6.101 decaf::util::concurrent::Callable< V > Class Template - Reference

A task that returns a result and may throw an exception.

```
#include <src/main/decaf/util/concurrent/Callable.h>
```

Public Member Functions

- virtual **~Callable** ()
- virtual V **call** ()=0

Computes a result, or throws an exception if unable to do so.

6.101.1 Detailed Description

```
template<typename V>class decaf::util::concurrent::Callable< V >
```

A task that returns a result and may throw an exception.

Implementors define a single method with no arguments called **call**. This interface differs from the **Runnable** interface in that a **Callable** (p. 746) object can return a result and is allowed to throw an exceptions from its **call** method.

The **Executors** (p. 1299) class contains utility methods to convert from other common forms to **Callable** (p. 746) classes.

Since

1.0

6.101.2 Constructor & Destructor Documentation

6.101.2.1 `template<typename V > virtual decaf::util::concurrent::Callable< V >::~~Callable () [inline, virtual]`

6.101.3 Member Function Documentation

6.101.3.1 `template<typename V > virtual V decaf::util::concurrent::Callable< V >::call () [pure virtual]`

Computes a result, or throws an exception if unable to do so.

Returns

Computed Result.

6.102 decaf::util::concurrent::ThreadPoolExecutor::CallerRunsPolicy Class

Reference

747

Exceptions

<i>Exception</i>	If unable to compute a result.
------------------	--------------------------------

The documentation for this class was generated from the following file:

- src/main/decaf/util/concurrent/**Callable.h**

6.102 decaf::util::concurrent::ThreadPoolExecutor::CallerRuns-Policy Class Reference

Handler policy for tasks that are rejected upon a call to **ThreadPoolExecutor::execute** (p.2724) this class will attempt to run the task in the Thread that called the execute method unless the executor is shutdown in which case the task is not run and is destroyed.

```
#include <src/main/decaf/util/concurrent/ThreadPoolExecutor.h>
```

Inheritance diagram for decaf::util::concurrent::ThreadPoolExecutor::CallerRunsPolicy:

Data Structures

- class **DiscardOldestPolicy**
*Handler policy for tasks that are rejected upon a call to **ThreadPoolExecutor::execute** (p. 2724) this class always destroys the oldest unexecuted task in the **Queue** (p. 2222) and then attempts to execute the rejected task using the passed in executor.*
- class **DiscardPolicy**
*Handler policy for tasks that are rejected upon a call to **ThreadPoolExecutor::execute** (p. 2724) this class always destroys the rejected task and returns quietly.*

Public Member Functions

- **CallerRunsPolicy** ()
- virtual **~CallerRunsPolicy** ()
- virtual void **rejectedExecution** (decaf::lang::Runnable *task, **ThreadPool-Executor** *executor **DECAF_UNUSED**)

6.102.1 Detailed Description

Handler policy for tasks that are rejected upon a call to **ThreadPoolExecutor::execute** (p.2724) this class will attempt to run the task in the Thread that called the execute method unless the executor is shutdown in which case the task is not run and is destroyed.

Since

1.0

6.102.2 Constructor & Destructor Documentation

6.102.2.1 **decaf::util::concurrent::ThreadPoolExecutor::CallerRunsPolicy::CallerRunsPolicy ()** [inline]

6.102.2.2 **virtual decaf::util::concurrent::ThreadPoolExecutor::~CallerRunsPolicy::~~CallerRunsPolicy ()** [inline, virtual]

6.102.3 Member Function Documentation

6.102.3.1 **virtual void decaf::util::concurrent::ThreadPoolExecutor::CallerRunsPolicy::rejectedExecution (decaf::lang::Runnable * task, ThreadPoolExecutor *executer *DECAF_UNUSED*)** [inline, virtual]

References decaf::lang::Runnable::run().

The documentation for this class was generated from the following file:

- src/main/decaf/util/concurrent/**ThreadPoolExecutor.h**

6.103 decaf::util::concurrent::CancellationException Class - Reference

```
#include <src/main/decaf/util/concurrent/Cancellation-Exception.h>
```

Inheritance diagram for decaf::util::concurrent::CancellationException:

Public Member Functions

- **CancellationException ()** throw ()
Default Constructor.
- **CancellationException (const decaf::lang::Exception &ex)** throw ()
Conversion Constructor from some other Exception.
- **CancellationException (const CancellationException &ex)** throw ()
Copy Constructor.
- **CancellationException (const std::exception *cause)** throw ()
Constructor.

- **CancellationException** (const char *file, const int lineNumber, const char *msg,...) throw ()
Constructor - Initializes the file name and line number where this message occurred.
- **CancellationException** (const char *file, const int lineNumber, const std::exception *cause, const char *msg,...) throw ()
Constructor - Initializes the file name and line number where this message occurred.
- virtual **CancellationException** * **clone** () const
Clones this exception.
- virtual ~**CancellationException** () throw ()

6.103.1 Constructor & Destructor Documentation

6.103.1.1 **decaf::util::concurrent::CancellationException::CancellationException**
 () throw () [inline]

Default Constructor.

6.103.1.2 **decaf::util::concurrent::CancellationException::CancellationException**
 (const decaf::lang::Exception & ex) throw () [inline]

Conversion Constructor from some other Exception.

Parameters

<i>ex</i>	An exception that should become this type of Exception
-----------	--

6.103.1.3 **decaf::util::concurrent::CancellationException::CancellationException**
 (const CancellationException & ex) throw () [inline]

Copy Constructor.

Parameters

<i>ex</i>	- The Exception to copy in this new instance.
-----------	---

6.103.1.4 **decaf::util::concurrent::CancellationException::CancellationException**
 (const std::exception * *cause*) throw () [inline]

Constructor.

Parameters

<i>cause</i>	Pointer to the exception that caused this one to be thrown, the object is cloned caller retains ownership.
--------------	--

6.103.1.5 `decaf::util::concurrent::CancellationException::CancellationException`
`(const char * file, const int lineNumber, const char * msg, ...) throw ()`
`[inline]`

Constructor - Initializes the file name and line number where this message occurred.

Sets the message to report, using an optional list of arguments to parse into the message

Parameters

<i>file</i>	- The file name where exception occurs
<i>lineNumber</i>	- The line number where the exception occurred.
<i>msg</i>	- The message to report
...	- list of primitives that are formatted into the message

6.103.1.6 `decaf::util::concurrent::CancellationException::CancellationException`
`(const char * file, const int lineNumber, const std::exception * cause, const char * msg, ...) throw ()` `[inline]`

Constructor - Initializes the file name and line number where this message occurred.

Sets the message to report, using an optional list of arguments to parse into the message

Parameters

<i>file</i>	- The file name where exception occurs
<i>lineNumber</i>	- The line number where the exception occurred.
<i>cause</i>	- The exception that was the cause for this one to be thrown.
<i>msg</i>	- The message to report
...	- list of primitives that are formatted into the message

6.103.1.7 `virtual decaf::util::concurrent::CancellationException-`
`::~CancellationException () throw ()` `[inline,`
`virtual]`

6.103.2 Member Function Documentation

6.103.2.1 `virtual CancellationException* decaf::util::concurrent-`
`::CancellationException::clone () const` `[inline,`
`virtual]`

Clones this exception.

This is useful for cases where you need to preserve the type of the original exception as well as the message. All subclasses should override.

Returns

a new instance of an exception that is a clone of this one.

Reimplemented from **decaf::lang::Exception** (p. 1282).

The documentation for this class was generated from the following file:

- src/main/decaf/util/concurrent/**CancellationException.h**

6.104 decaf::security::cert::Certificate Class Reference

Base interface for all identity certificates.

```
#include <src/main/decaf/security/cert/Certificate.h>
```

Inheritance diagram for decaf::security::cert::Certificate:

Public Member Functions

- virtual **~Certificate** ()
- virtual bool **equals** (const **Certificate** &cert) const =0
Compares the encoded form of the two certificates.
- virtual void **getEncoded** (std::vector< unsigned char > &output) const =0
Provides the encoded form of this certificate.
- virtual std::string **getType** () const =0
Returns the type of this certificate.
- virtual **PublicKey** * **getPublicKey** ()=0
Gets the public key of this certificate.
- virtual const **PublicKey** * **getPublicKey** () const =0
Gets the public key of this certificate.
- virtual void **verify** (const **PublicKey** &publicKey) const =0
Verifies that this certificate was signed with the private key that corresponds to the specified public key.
- virtual void **verify** (const **PublicKey** &publicKey, const std::string &sigProvider) const =0
Verifies that this certificate was signed with the private key that corresponds to the specified public key.
- virtual std::string **toString** () const =0
Returns a string representation of this certificate.

6.104.1 Detailed Description

Base interface for all identity certificates.

6.104.2 Constructor & Destructor Documentation

6.104.2.1 `virtual decaf::security::cert::Certificate::~~Certificate () [inline, virtual]`

6.104.3 Member Function Documentation

6.104.3.1 `virtual bool decaf::security::cert::Certificate::equals (const Certificate & cert) const [pure virtual]`

Compares the encoded form of the two certificates.

Parameters

<i>cert</i>	The certificate to be tested for equality with this certificate.
-------------	--

Returns

true if the given certificate is equal to this certificate.

6.104.3.2 `virtual void decaf::security::cert::Certificate::getEncoded (std::vector< unsigned char > & output) const [pure virtual]`

Provides the encoded form of this certificate.

Parameters

<i>output</i>	Receives the encoded form of this certificate.
---------------	--

Exceptions

Certificate- EncodingException (p. 755)	if an encoding error occurs
---	-----------------------------

6.104.3.3 `virtual PublicKey* decaf::security::cert::Certificate::getPublicKey () [pure virtual]`

Gets the public key of this certificate.

Returns

the public key

6.104.3.4 `virtual const PublicKey* decaf::security::cert::Certificate::getPublicKey () const [pure virtual]`

Gets the public key of this certificate.

Returns

the public key

6.104.3.5 `virtual std::string decaf::security::cert::Certificate::getType () const [pure virtual]`

Returns the type of this certificate.

Returns

the type of this certificate

6.104.3.6 `virtual std::string decaf::security::cert::Certificate::toString () const [pure virtual]`

Returns a string representation of this certificate.

Returns

a string representation of this certificate

6.104.3.7 `virtual void decaf::security::cert::Certificate::verify (const PublicKey & publicKey) const [pure virtual]`

Verifies that this certificate was signed with the private key that corresponds to the specified public key.

Parameters

<i>publicKey</i>	The public key used to carry out the validation.
------------------	--

Exceptions

NoSuchAlgorithm-Exception (p. 1981)	- on unsupported signature algorithms.
InvalidKey-Exception (p. 1536)	- on incorrect key.

NoSuchProvider-Exception (p. 1986)	- if there's no default provider.
SignatureException (p. 2438)	- on signature errors.
Certificate-Exception (p. 757)	- on encoding errors.

6.104.3.8 `virtual void decaf::security::cert::Certificate::verify (const PublicKey & publicKey, const std::string & sigProvider) const [pure virtual]`

Verifies that this certificate was signed with the private key that corresponds to the specified public key.

Uses the verification engine of the specified provider.

Parameters

<i>publicKey</i>	The public key used to carry out the validation.
<i>sigProvider</i>	The name of the signature provider

Exceptions

NoSuchAlgorithm-Exception (p. 1981)	- on unsupported signature algorithms.
InvalidKey-Exception (p. 1536)	- on incorrect key.
NoSuchProvider-Exception (p. 1986)	- if there's no default provider.
SignatureException (p. 2438)	- on signature errors.
Certificate-Exception (p. 757)	- on encoding errors.

The documentation for this class was generated from the following file:

- src/main/decaf/security/cert/**Certificate.h**

6.105 decaf::security::cert::CertificateEncodingException Class - Reference

```
#include <src/main/decaf/security/cert/CertificateEncodingException.h>
```

Inheritance diagram for decaf::security::cert::CertificateEncodingException:

Public Member Functions

- **CertificateEncodingException** () throw ()
Default Constructor.
- **CertificateEncodingException** (const **Exception** &ex) throw ()
Conversion Constructor from some other Exception.
- **CertificateEncodingException** (const **CertificateEncodingException** &ex) throw ()
Copy Constructor.
- **CertificateEncodingException** (const char *file, const int lineNumber, const char *msg,...) throw ()
Constructor - Initializes the file name and line number where this message occurred.
- virtual **CertificateEncodingException** * **clone** () const
Clones this exception.
- virtual ~**CertificateEncodingException** () throw ()

6.105.1 Constructor & Destructor Documentation

6.105.1.1 decaf::security::cert::CertificateEncodingException::CertificateEncodingException () throw () [inline]

Default Constructor.

6.105.1.2 decaf::security::cert::CertificateEncodingException::~CertificateEncodingException (const Exception & ex) throw () [inline]

Conversion Constructor from some other Exception.

Parameters

ex	An exception that should become this type of Exception
----	--

6.105.1.3 **decaf::security::cert::CertificateEncodingException::CertificateEncodingException** (const CertificateEncodingException & ex) throw ()
[inline]

Copy Constructor.

Parameters

<i>ex</i>	An exception that should become this type of Exception
-----------	--

6.105.1.4 **decaf::security::cert::CertificateEncodingException::CertificateEncodingException** (const char * *file*, const int *lineNumber*, const char * *msg*, ...) throw () [inline]

Constructor - Initializes the file name and line number where this message occurred.

Sets the message to report, using an optional list of arguments to parse into the message

Parameters

<i>file</i>	name where exception occurs
<i>lineNumber</i>	line number where the exception occurred.
<i>msg</i>	message to report
...	list of primitives that are formatted into the message

6.105.1.5 **virtual decaf::security::cert::CertificateEncodingException::~CertificateEncodingException** () throw () [inline, virtual]

6.105.2 Member Function Documentation

6.105.2.1 **virtual CertificateEncodingException* decaf::security::cert::CertificateEncodingException::clone** () const [inline, virtual]

Clones this exception.

This is useful for cases where you need to preserve the type of the original exception as well as the message. All subclasses should override.

Returns

A deep copy of this exception.

Reimplemented from **decaf::security::cert::CertificateException** (p. 758).

The documentation for this class was generated from the following file:

- src/main/decaf/security/cert/**CertificateEncodingException.h**

6.106 decaf::security::cert::CertificateException Class Reference

```
#include <src/main/decaf/security/cert/CertificateException.h>
```

Inheritance diagram for decaf::security::cert::CertificateException:

Public Member Functions

- **CertificateException** () throw ()
Default Constructor.
- **CertificateException** (const **Exception** &ex) throw ()
Conversion Constructor from some other Exception.
- **CertificateException** (const **CertificateException** &ex) throw ()
Copy Constructor.
- **CertificateException** (const char *file, const int lineNumber, const char *msg,...) throw ()
Constructor - Initializes the file name and line number where this message occurred.
- virtual **CertificateException** * **clone** () const
Clones this exception.
- virtual ~**CertificateException** () throw ()

6.106.1 Constructor & Destructor Documentation

6.106.1.1 decaf::security::cert::CertificateException::CertificateException () throw () [inline]

Default Constructor.

6.106.1.2 decaf::security::cert::CertificateException::CertificateException (const Exception & ex) throw () [inline]

Conversion Constructor from some other Exception.

Parameters

<i>ex</i>	An exception that should become this type of Exception
-----------	--

6.106.1.3 decaf::security::cert::CertificateException::CertificateException (const CertificateException & ex) throw () [inline]

Copy Constructor.

Parameters

<i>ex</i>	An exception that should become this type of Exception
-----------	--

6.106.1.4 `decaf::security::cert::CertificateException::CertificateException (const char * file, const int lineNumber, const char * msg, ...) throw () [inline]`

Constructor - Initializes the file name and line number where this message occurred.

Sets the message to report, using an optional list of arguments to parse into the message

Parameters

<i>file</i>	name where exception occurs
<i>lineNumber</i>	line number where the exception occurred.
<i>msg</i>	message to report
<i>...</i>	list of primitives that are formatted into the message

6.106.1.5 `virtual decaf::security::cert::CertificateException::~CertificateException () throw () [inline, virtual]`

6.106.2 Member Function Documentation

6.106.2.1 `virtual CertificateException* decaf::security::cert::CertificateException::clone () const [inline, virtual]`

Clones this exception.

This is useful for cases where you need to preserve the type of the original exception as well as the message. All subclasses should override.

Returns

A deep copy of this exception.

Reimplemented from **decaf::security::GeneralSecurityException** (p. 1397).

Reimplemented in **decaf::security::cert::CertificateExpiredException** (p. 760), **decaf::security::cert::CertificateNotYetValidException** (p. 762), **decaf::security::cert::CertificateParsingException** (p. 764), and **decaf::security::cert::CertificateEncodingException** (p. 756).

The documentation for this class was generated from the following file:

- `src/main/decaf/security/cert/CertificateException.h`

6.107 decaf::security::cert::CertificateExpiredException Class - Reference

```
#include <src/main/decaf/security/cert/CertificateExpired-Exception.h>
```

Inheritance diagram for decaf::security::cert::CertificateExpiredException:

Public Member Functions

- **CertificateExpiredException** () throw ()
Default Constructor.
- **CertificateExpiredException** (const **Exception** &ex) throw ()
Conversion Constructor from some other Exception.
- **CertificateExpiredException** (const **CertificateExpiredException** &ex) throw ()
Copy Constructor.
- **CertificateExpiredException** (const char *file, const int lineNumber, const char *msg,...) throw ()
Constructor - Initializes the file name and line number where this message occurred.
- virtual **CertificateExpiredException** * **clone** () const
Clones this exception.
- virtual ~**CertificateExpiredException** () throw ()

6.107.1 Constructor & Destructor Documentation

6.107.1.1 decaf::security::cert::CertificateExpiredException::CertificateExpiredException () throw () [inline]

Default Constructor.

6.107.1.2 decaf::security::cert::CertificateExpiredException::~CertificateExpiredException (const Exception & ex) throw () [inline]

Conversion Constructor from some other Exception.

Parameters

ex	An exception that should become this type of Exception
----	--

6.107.1.3 **decaf::security::cert::CertificateExpiredException::CertificateExpiredException** (const CertificateExpiredException & *ex*) throw ()
[inline]

Copy Constructor.

Parameters

<i>ex</i>	An exception that should become this type of Exception
-----------	--

6.107.1.4 **decaf::security::cert::CertificateExpiredException::CertificateExpiredException** (const char * *file*, const int *lineNumber*, const char * *msg*, ...) throw ()
[inline]

Constructor - Initializes the file name and line number where this message occurred.

Sets the message to report, using an optional list of arguments to parse into the message

Parameters

<i>file</i>	name where exception occurs
<i>lineNumber</i>	line number where the exception occurred.
<i>msg</i>	message to report
...	list of primitives that are formatted into the message

6.107.1.5 **virtual decaf::security::cert::CertificateExpiredException::~CertificateExpiredException** () throw () [inline, virtual]

6.107.2 Member Function Documentation

6.107.2.1 **virtual CertificateExpiredException* decaf::security::cert::CertificateExpiredException::clone** () const [inline, virtual]

Clones this exception.

This is useful for cases where you need to preserve the type of the original exception as well as the message. All subclasses should override.

Returns

A deep copy of this exception.

Reimplemented from **decaf::security::cert::CertificateException** (p. 758).

The documentation for this class was generated from the following file:

- src/main/decaf/security/cert/CertificateExpiredException.h

6.108 decaf::security::cert::CertificateNotYetValidException Class Reference 761

6.108 decaf::security::cert::CertificateNotYetValidException Class Reference

```
#include <src/main/decaf/security/cert/CertificateNotYetValidException.h>
```

Inheritance diagram for decaf::security::cert::CertificateNotYetValidException:

Public Member Functions

- **CertificateNotYetValidException** () throw ()
Default Constructor.
- **CertificateNotYetValidException** (const **Exception** &ex) throw ()
Conversion Constructor from some other Exception.
- **CertificateNotYetValidException** (const **CertificateNotYetValidException** &ex) throw ()
Copy Constructor.
- **CertificateNotYetValidException** (const char *file, const int lineNumber, const char *msg,...) throw ()
Constructor - Initializes the file name and line number where this message occurred.
- virtual **CertificateNotYetValidException** * **clone** () const
Clones this exception.
- virtual ~**CertificateNotYetValidException** () throw ()

6.108.1 Constructor & Destructor Documentation

6.108.1.1 decaf::security::cert::CertificateNotYetValidException-
::CertificateNotYetValidException () throw ()
[inline]

Default Constructor.

6.108.1.2 decaf::security::cert::CertificateNotYetValidException::-
CertificateNotYetValidException (const Exception & ex) throw ()
[inline]

Conversion Constructor from some other Exception.

Parameters

ex	An exception that should become this type of Exception
----	--

6.108.1.3 **decaf::security::cert::CertificateNotYetValidException::CertificateNotYetValidException** (const **CertificateNotYetValidException** & *ex*) throw () [inline]

Copy Constructor.

Parameters

<i>ex</i>	An exception that should become this type of Exception
-----------	--

6.108.1.4 **decaf::security::cert::CertificateNotYetValidException::CertificateNotYetValidException** (const char * *file*, const int *lineNumber*, const char * *msg*, ...) throw () [inline]

Constructor - Initializes the file name and line number where this message occurred.

Sets the message to report, using an optional list of arguments to parse into the message

Parameters

<i>file</i>	name where exception occurs
<i>lineNumber</i>	line number where the exception occurred.
<i>msg</i>	message to report
...	list of primitives that are formatted into the message

6.108.1.5 **virtual decaf::security::cert::CertificateNotYetValidException::~CertificateNotYetValidException** () throw () [inline, virtual]

6.108.2 Member Function Documentation

6.108.2.1 **virtual CertificateNotYetValidException* decaf::security::cert::CertificateNotYetValidException::clone** () const [inline, virtual]

Clones this exception.

This is useful for cases where you need to preserve the type of the original exception as well as the message. All subclasses should override.

Returns

A deep copy of this exception.

Reimplemented from **decaf::security::cert::CertificateException** (p. 758).

The documentation for this class was generated from the following file:

- src/main/decaf/security/cert/**CertificateNotYetValidException.h**

6.109 decaf::security::cert::CertificateParsingException Class - Reference

```
#include <src/main/decaf/security/cert/CertificateParsing-Exception.h>
```

Inheritance diagram for decaf::security::cert::CertificateParsingException:

Public Member Functions

- **CertificateParsingException** () throw ()
Default Constructor.
- **CertificateParsingException** (const **Exception** &ex) throw ()
Conversion Constructor from some other Exception.
- **CertificateParsingException** (const **CertificateParsingException** &ex) throw ()
Copy Constructor.
- **CertificateParsingException** (const char *file, const int lineNumber, const char *msg,...) throw ()
Constructor - Initializes the file name and line number where this message occurred.
- virtual **CertificateParsingException** * **clone** () const
Clones this exception.
- virtual ~**CertificateParsingException** () throw ()

6.109.1 Constructor & Destructor Documentation

6.109.1.1 decaf::security::cert::CertificateParsingException::CertificateParsingException () throw () [inline]

Default Constructor.

6.109.1.2 decaf::security::cert::CertificateParsingException::~CertificateParsingException (const Exception & ex) throw () [inline]

Conversion Constructor from some other Exception.

Parameters

ex	An exception that should become this type of Exception
----	--

6.109.1.3 **decaf::security::cert::CertificateParsingException::CertificateParsingException** (const CertificateParsingException & ex) throw ()
[inline]

Copy Constructor.

Parameters

<i>ex</i>	An exception that should become this type of Exception
-----------	--

6.109.1.4 **decaf::security::cert::CertificateParsingException::CertificateParsingException** (const char * file, const int lineNumber, const char * msg, ...) throw ()
[inline]

Constructor - Initializes the file name and line number where this message occurred.

Sets the message to report, using an optional list of arguments to parse into the message

Parameters

<i>file</i>	name where exception occurs
<i>lineNumber</i>	line number where the exception occurred.
<i>msg</i>	message to report
<i>...</i>	list of primitives that are formatted into the message

6.109.1.5 **virtual decaf::security::cert::CertificateParsingException::~CertificateParsingException** () throw () [inline, virtual]

6.109.2 Member Function Documentation

6.109.2.1 **virtual CertificateParsingException* decaf::security::cert::CertificateParsingException::clone** () const [inline, virtual]

Clones this exception.

This is useful for cases where you need to preserve the type of the original exception as well as the message. All subclasses should override.

Returns

A deep copy of this exception.

Reimplemented from **decaf::security::cert::CertificateException** (p. 758).

The documentation for this class was generated from the following file:

- src/main/decaf/security/cert/**CertificateParsingException.h**

6.110 decaf::lang::Character Class Reference

```
#include <src/main/decaf/lang/Character.h>
```

Inheritance diagram for decaf::lang::Character:

Public Member Functions

- **Character** (char value)
- virtual int **compareTo** (const **Character** &c) const
*Compares this **Character** (p. 765) instance with another.*
- virtual bool **operator==** (const **Character** &c) const
Compares equality between this object and the one passed.
- virtual bool **operator<** (const **Character** &c) const
Compares this object to another and returns true if this object is considered to be less than the one passed.
- virtual int **compareTo** (const char &c) const
*Compares this **Character** (p. 765) instance with a char type.*
- virtual bool **operator==** (const char &c) const
Compares equality between this object and the one passed.
- virtual bool **operator<** (const char &c) const
Compares this object to another and returns true if this object is considered to be less than the one passed.
- bool **equals** (const **Character** &c) const
- bool **equals** (const char &c) const
- std::string **toString** () const
- virtual double **doubleValue** () const
Answers the double value which the receiver represents.
- virtual float **floatValue** () const
Answers the float value which the receiver represents.
- virtual unsigned char **byteValue** () const
Answers the byte value which the receiver represents.
- virtual short **shortValue** () const
Answers the short value which the receiver represents.
- virtual int **intValue** () const
Answers the int value which the receiver represents.
- virtual long long **longValue** () const
Answers the long value which the receiver represents.

Static Public Member Functions

- static **Character** **valueOf** (char value)
*Returns a **Character** (p. 765) instance representing the specified char value.*
- static bool **isWhitespace** (char c)
Indicates whether or not the given character is considered whitespace.
- static bool **isDigit** (char c)
Indicates whether or not the given character is a digit.
- static bool **isLowerCase** (char c)
Indicates whether or not the given character is a lower case character.
- static bool **isUpperCase** (char c)
Indicates whether or not the given character is a upper case character.
- static bool **isLetter** (char c)
Indicates whether or not the given character is a letter.
- static bool **isLetterOrDigit** (char c)
Indicates whether or not the given character is either a letter or a digit.
- static bool **isISOControl** (char c)
Answers whether the character is an ISO control character, which is a char that lays in the range of 0 to 1f and 7f to 9f.
- static int **digit** (char c, int radix)
Returns the numeric value of the character ch in the specified radix.

Static Public Attributes

- static const int **MIN_RADIX** = 2
The minimum radix available for conversion to and from strings.
- static const int **MAX_RADIX** = 36
The maximum radix available for conversion to and from strings.
- static const char **MIN_VALUE** = (char)0x7F
The minimum value that a signed char can take on.
- static const char **MAX_VALUE** = (char)0x80
The maximum value that a signed char can take on.
- static const int **SIZE** = 8
The size of the primitive charactor in bits.

6.110.1 Constructor & Destructor Documentation

6.110.1.1 `decaf::lang::Character::Character (char value)`

Parameters

<i>value</i>	- char to wrap.
--------------	-----------------

6.110.2 Member Function Documentation

6.110.2.1 virtual unsigned char **decaf::lang::Character::byteValue** () const
[inline, virtual]

Answers the byte value which the receiver represents.

Returns

int the value of the receiver.

Reimplemented from **decaf::lang::Number** (p. 1993).

6.110.2.2 virtual int **decaf::lang::Character::compareTo** (const **Character** & *c*) const
[inline, virtual]

Compares this **Character** (p. 765) instance with another.

Parameters

<i>c</i>	- the Character (p. 765) instance to be compared
----------	---

Returns

zero if this object represents the same char value as the argument; a positive value if this object represents a value greater than the passed in value, and -1 if this object represents a value less than the passed in value.

Implements **decaf::lang::Comparable**< **Character** > (p. 886).

6.110.2.3 virtual int **decaf::lang::Character::compareTo** (const char & *c*) const
[inline, virtual]

Compares this **Character** (p. 765) instance with a char type.

Parameters

<i>c</i>	- the char instance to be compared
----------	------------------------------------

Returns

zero if this object represents the same char value as the argument; a positive value if this object represents a value greater than the passed in value, and -1 if this object represents a value less than the passed in value.

Implements **decaf::lang::Comparable**< **char** > (p. 886).

6.110.2.4 `static int decaf::lang::Character::digit (char c, int radix)` `[static]`

Returns the numeric value of the character *ch* in the specified radix.

If the radix is not in the range `MIN_RADIX <= radix <= MAX_RADIX` or if the value of *ch* is not a valid digit in the specified radix, -1 is returned. A character is a valid digit if at least one of the following is true:

- * The method `isDigit` is true of the character and the single-character decomposition is less than the specified radix. In this case the decimal digit value is returned.
- * The character is one of the uppercase Latin letters 'A' through 'Z' and its code is less than `radix + 'A' - 10`. In this case, `ch - 'A' + 10` is returned.
- * The character is one of the lowercase Latin letters 'a' through 'z' and its code is less than `radix + 'a' - 10`. In this case, `ch - 'a' + 10` is returned.

Parameters

<i>c</i>	- the char to be converted
<i>radix</i>	- the radix of the number

Returns

the numeric value of the number represented in the given radix

6.110.2.5 `virtual double decaf::lang::Character::doubleValue () const` `[inline, virtual]`

Answers the double value which the receiver represents.

Returns

double the value of the receiver.

Implements `decaf::lang::Number` (p. 1993).

6.110.2.6 `bool decaf::lang::Character::equals (const Character & c) const` `[inline, virtual]`

Returns

true if the two **Character** (p. 765) Objects have the same value.

Implements `decaf::lang::Comparable< Character >` (p. 887).

6.110.2.7 `bool decaf::lang::Character::equals (const char & c) const` `[inline, virtual]`

Returns

true if the two Characters have the same value.

Implements `decaf::lang::Comparable< char >` (p. 887).

6.110.2.8 `virtual float decaf::lang::Character::floatValue () const [inline, virtual]`

Answers the float value which the receiver represents.

Returns

float the value of the receiver.

Implements **decaf::lang::Number** (p. 1993).

6.110.2.9 `virtual int decaf::lang::Character::intValue () const [inline, virtual]`

Answers the int value which the receiver represents.

Returns

int the value of the receiver.

Implements **decaf::lang::Number** (p. 1993).

6.110.2.10 `static bool decaf::lang::Character::isDigit (char c) [inline, static]`

Indicates whether or not the given character is a digit.

6.110.2.11 `static bool decaf::lang::Character::isISOControl (char c) [inline, static]`

Answers whether the character is an ISO control character, which is a char that lays in the range of 0 to 1f and 7f to 9f.

Parameters

<code>c</code>	- the character, including supplementary characters
----------------	---

Returns

true if the char is an ISO control character

6.110.2.12 `static bool decaf::lang::Character::isLetter (char c) [inline, static]`

Indicates whether or not the given character is a letter.

6.110.2.13 `static bool decaf::lang::Character::isLetterOrDigit (char c)` `[inline, static]`

Indicates whether or not the given character is either a letter or a digit.

6.110.2.14 `static bool decaf::lang::Character::isLowerCase (char c)` `[inline, static]`

Indicates whether or not the given character is a lower case character.

6.110.2.15 `static bool decaf::lang::Character::isUpperCase (char c)` `[inline, static]`

Indicates whether or not the given character is a upper case character.

6.110.2.16 `static bool decaf::lang::Character::isWhitespace (char c)` `[inline, static]`

Indicates whether or not the given character is considered whitespace.

6.110.2.17 `virtual long long decaf::lang::Character::longValue () const` `[inline, virtual]`

Answers the long value which the receiver represents.

Returns

long the value of the receiver.

Implements **decaf::lang::Number** (p. 1994).

6.110.2.18 `virtual bool decaf::lang::Character::operator< (const Character & c) const` `[inline, virtual]`

Compares this object to another and returns true if this object is considered to be less than the one passed.

This

Parameters

<i>c</i>	- the value to be compared to this one.
----------	---

Returns

true if this object is equal to the one passed.

Implements **decaf::lang::Comparable**< **Character** > (p. 887).

6.110.2.19 `virtual bool decaf::lang::Character::operator< (const char & c) const`
[inline, virtual]

Compares this object to another and returns true if this object is considered to be less than the one passed.

This

Parameters

<code>c</code>	- the value to be compared to this one.
----------------	---

Returns

true if this object is equal to the one passed.

Implements **decaf::lang::Comparable**< **char** > (p. 887).

6.110.2.20 `virtual bool decaf::lang::Character::operator== (const Character & c) const`
[inline, virtual]

Compares equality between this object and the one passed.

Parameters

<code>c</code>	- the value to be compared to this one.
----------------	---

Returns

true if this object is equal to the one passed.

Implements **decaf::lang::Comparable**< **Character** > (p. 888).

6.110.2.21 `virtual bool decaf::lang::Character::operator== (const char & c) const`
[inline, virtual]

Compares equality between this object and the one passed.

Parameters

<code>c</code>	- the value to be compared to this one.
----------------	---

Returns

true if this object is equal to the one passed.

Implements **decaf::lang::Comparable**< **char** > (p. 888).

6.110.2.22 `virtual short decaf::lang::Character::shortValue () const` `[inline, virtual]`

Answers the short value which the receiver represents.

Returns

int the value of the receiver.

Reimplemented from **decaf::lang::Number** (p. 1994).

6.110.2.23 `std::string decaf::lang::Character::toString () const`

Returns

this **Character** (p. 765) Object as a **String** (p. 2620) Representation

6.110.2.24 `static Character decaf::lang::Character::valueOf (char value)` `[inline, static]`

Returns a **Character** (p. 765) instance representing the specified char value.

Parameters

<i>value</i>	- the primitive char to wrap.
--------------	-------------------------------

Returns

a new Charactor instance that wraps this value.

6.110.3 Field Documentation

6.110.3.1 `const int decaf::lang::Character::MAX_RADIX = 36` `[static]`

The maximum radix available for conversion to and from strings.

6.110.3.2 `const char decaf::lang::Character::MAX_VALUE = (char)0x80` `[static]`

The maximum value that a signed char can take on.

6.110.3.3 `const int decaf::lang::Character::MIN_RADIX = 2` `[static]`

The minimum radix available for conversion to and from strings.

6.110.3.4 `const char decaf::lang::Character::MIN_VALUE = (char)0x7F` `[static]`

The minimum value that a signed char can take on.

6.110.3.5 `const int decaf::lang::Character::SIZE = 8` `[static]`

The size of the primitive character in bits.

The documentation for this class was generated from the following file:

- `src/main/decaf/lang/Character.h`

6.111 decaf::internal::nio::CharArrayBuffer Class Reference

```
#include <src/main/decaf/internal/nio/CharArrayBuffer.h>
```

Inheritance diagram for `decaf::internal::nio::CharArrayBuffer`:

Public Member Functions

- **CharArrayBuffer** (int size, bool **readOnly**=false)
*Creates a **CharArrayBuffer** (p. 773) object that has its backing array allocated internally and is then owned and deleted when this object is deleted.*
- **CharArrayBuffer** (char *array, int size, int **offset**, int **length**, bool **readOnly**=false)
*Creates a **CharArrayBuffer** (p. 773) object that wraps the given array.*
- **CharArrayBuffer** (const `decaf::lang::Pointer< ByteArrayAdapter >` &array, int **offset**, int **length**, bool **readOnly**=false)
Creates a byte buffer that wraps the passed ByteArrayAdapter and start at the given offset.
- **CharArrayBuffer** (const **CharArrayBuffer** &other)
*Create a **CharArrayBuffer** (p. 773) that mirrors this one, meaning it shares a reference to this buffers ByteArrayAdapter and when changes are made to that data it is reflected in both.*
- virtual `~CharArrayBuffer ()`
- virtual char * **array** ()
*Returns the character array that backs this buffer (optional operation).
Modifications to this buffer's content will cause the returned array's content to be modified, and vice versa.
Invoke the `hasArray` method before invoking this method in order to ensure that this buffer has an accessible backing array.*

Returns

the array that backs this **Buffer** (p. 582).

Exceptions

ReadOnlyBuffer-Exception (p. 2244)	if this Buffer (p. 582) is read only.
UnsupportedOperation-Exception	if the underlying store has no array.

- virtual int **arrayOffset** ()

Returns the offset within this buffer's backing array of the first element of the buffer (optional operation).

Invoke the `hasArray` method before invoking this method in order to ensure that this buffer has an accessible backing array.

Returns

The offset into the backing array where index zero starts.

Exceptions

ReadOnlyBuffer-Exception (p. 2244)	if this Buffer (p. 582) is read only.
UnsupportedOperation-Exception	if the underlying store has no array.

- virtual **CharBuffer * asReadOnlyBuffer** () const

Creates a new, read-only char buffer that shares this buffer's content.

The content of the new buffer will be that of this buffer. Changes to this buffer's content will be visible in the new buffer; the new buffer itself, however, will be read-only and will not allow the shared content to be modified. The two buffers' position, limit, and mark values will be independent.

If this buffer is itself read-only then this method behaves in exactly the same way as the `duplicate` method.

The new buffer's capacity, limit, position, and mark values will be identical to those of this buffer.

Returns

The new, read-only char buffer which the caller then owns.

- virtual **CharBuffer & compact** ()

Compacts this buffer.

The bytes between the buffer's current position and its limit, if any, are copied to the beginning of the buffer. That is, the byte at index $p = \text{position}()$ (p. 587) is copied to index zero, the byte at index $p + 1$ is copied to index one, and so forth until the byte at index $\text{limit}()$ (p. 586) - 1 is copied to index $n = \text{limit}()$ (p. 586) - 1 - p . The buffer's position is then set to $n+1$ and its limit is set to its capacity. The mark, if defined, is discarded.

The buffer's position is set to the number of bytes copied, rather than to zero, so that an invocation of this method can be followed immediately by an invocation of another relative put method.

Returns

a reference to this **CharBuffer** (p. 785).

Exceptions

ReadOnlyBuffer-Exception (p. 2244)	- If this buffer is read-only
--	-------------------------------

- virtual **CharBuffer** * **duplicate** ()

Creates a new char buffer that shares this buffer's content.

The content of the new buffer will be that of this buffer. Changes to this buffer's content will be visible in the new buffer, and vice versa; the two buffers' position, limit, and mark values will be independent.

The new buffer's capacity, limit, position, and mark values will be identical to those of this buffer. The new buffer will be read-only if, and only if, this buffer is read-only.

Returns

*a new char **Buffer** (p. 582) which the caller owns.*

- virtual char **get** ()

Relative get method.

Reads the character at this buffer's current position, and then increments the position.

Returns

the char at the current position.

Exceptions

BufferUnderflow-Exception (p. 611)	if there no more data to return
--	---------------------------------

- virtual char **get** (int index) const

Absolute get method.

Reads the char at the given index.

Parameters

index	The index in the Buffer (p. 582) where the char is to be read.
-------	---

Returns

the char that is located at the given index.

Exceptions

IndexOutOfBoundsException	if index is not smaller than the buffer's limit or is negative.
---------------------------	---

- virtual bool **hasArray** () const

Tells whether or not this buffer is backed by an accessible char array.

If this method returns true then the array and arrayOffset methods may safely be invoked. Subclasses should override this method if they do not have a backing array as this class always returns true.

Returns

true if, and only if, this buffer is backed by an array and is not read-only

- virtual bool **isReadOnly** () const

Tells whether or not this buffer is read-only.

Returns

true if, and only if, this buffer is read-only.

- virtual **CharBuffer & put** (char value)

Writes the given char into this buffer at the current position, and then increments the position.

Parameters

value	<i>The char value to be written.</i>
-------	--------------------------------------

Returns

a reference to this buffer.

Exceptions

BufferOverflow-Exception (p. 609)	<i>if this buffer's current position is not smaller than its limit</i>
ReadOnlyBuffer-Exception (p. 2244)	<i>if this buffer is read-only.</i>

- virtual **CharBuffer & put** (int index, char value)

Writes the given char into this buffer at the given index.

Parameters

index	<i>The position in the Buffer (p. 582) to write the data.</i>
value	<i>The char to write.</i>

Returns

a reference to this buffer.

Exceptions

IndexOutOfBoundsException	<i>if index greater than the buffer's limit minus the size of the type being written, or index is negative.</i>
ReadOnlyBuffer-Exception (p. 2244)	<i>if this buffer is read-only.</i>

- virtual **CharBuffer * slice ()** const

*Creates a new **CharBuffer** (p. 785) whose content is a shared subsequence of this buffer's content.*

The content of the new buffer will start at this buffer's current position. Changes to this buffer's content will be visible in the new buffer, and vice versa; the two buffers' position, limit, and mark values will be independent.

The new buffer's position will be zero, its capacity and its limit will be the number of bytes remaining in this buffer, and its mark will be undefined. The new buffer will be read-only if, and only if, this buffer is read-only.

Returns

*the newly create **CharBuffer** (p. 785) which the caller owns.*

- virtual **lang::CharSequence * subSequence** (int start, int end) const

Creates a new character buffer that represents the specified subsequence of this buffer, relative to the current position.

The new buffer will share this buffer's content; that is, if the content of this buffer is mutable then modifications to one buffer will cause the other to be modified. The new

buffer's capacity will be that of this buffer, its position will be **position()** (p. 587) + start, and its limit will be **position()** (p. 587) + end. The new **Buffer** (p. 582) will be read-only if, and only if, this buffer is read-only.

Parameters

start	The index, relative to the current position, of the first character in the subsequence; must be non-negative and no larger than remaining() (p. 588).
end	The index, relative to the current position, of the character following the last character in the subsequence; must be no smaller than start and no larger than remaining() (p. 588).

Returns

The new character buffer, caller owns.

Exceptions

IndexOutOfBoundsException	if the preconditions on start and end fail.
---------------------------	---

Protected Member Functions

- virtual void **setReadOnly** (bool value)

Sets this **CharArrayBuffer** (p. 773) as Read-Only.

Protected Attributes

- decaf::lang::Pointer < ByteArrayAdapter > _array
- int **offset**
- int **length**
- bool **readOnly**

6.111.1 Constructor & Destructor Documentation

6.111.1.1 decaf::internal::nio::CharArrayBuffer::CharArrayBuffer (int size, bool readOnly = false)

Creates a **CharArrayBuffer** (p. 773) object that has its backing array allocated internally and is then owned and deleted when this object is deleted.

The array is initially created with all elements initialized to zero.

Parameters

size	The size of the array, this is the limit we read and write to.
readOnly	Boolean indicating if this buffer should be read-only, default as false.

Exceptions

IllegalArgumentException	if the capacity value is negative.
--------------------------	------------------------------------

6.111.1.2 **decaf::internal::nio::CharArrayBuffer::CharArrayBuffer** (*char * array*, *int size*, *int offset*, *int length*, *bool readOnly = false*)

Creates a **CharArrayBuffer** (p. 773) object that wraps the given array.

If the own flag is set then it will delete this array when this object is deleted.

Parameters

<i>array</i>	The actual array to wrap.
<i>size</i>	The size of the given array.
<i>offset</i>	The position that is this buffers start position.
<i>length</i>	The limit of how many bytes into the array this Buffer can write.
<i>readOnly</i>	Boolean indicating if this buffer should be read-only, default as false.

Exceptions

<i>NullPointerException</i>	if buffer is NULL
<i>IndexOutOfBoundsException</i>	if offset is greater than array capacity.

6.111.1.3 **decaf::internal::nio::CharArrayBuffer::CharArrayBuffer** (*const decaf::lang::Pointer< ByteArrayAdapter > & array*, *int offset*, *int length*, *bool readOnly = false*)

Creates a byte buffer that wraps the passed ByteArrayAdapter and start at the given offset.

The capacity and limit of the new **CharArrayBuffer** (p. 773) will be that of the remaining capacity of the passed buffer.

Parameters

<i>array</i>	The ByteArrayAdapter to wrap.
<i>offset</i>	The position that is this buffers start position.
<i>length</i>	The limit of how many bytes into the array this Buffer can write.
<i>readOnly</i>	Boolean indicating if this buffer should be read-only, default as false.

Exceptions

<i>NullPointerException</i>	if array is NULL
<i>IndexOutOfBoundsException</i>	if offset + length is greater than array size.

6.111.1.4 decaf::internal::nio::CharArrayBuffer::CharArrayBuffer (const CharArrayBuffer & other)

Create a **CharArrayBuffer** (p. 773) that mirrors this one, meaning it shares a reference to this buffers ByteArrayAdapter and when changes are made to that data it is reflected in both.

Parameters

<i>other</i>	The CharArrayBuffer (p. 773) this one is to mirror.
--------------	--

6.111.1.5 virtual decaf::internal::nio::CharArrayBuffer::~CharArrayBuffer () [virtual]

6.111.2 Member Function Documentation

6.111.2.1 virtual char* decaf::internal::nio::CharArrayBuffer::array () [virtual]

Returns the character array that backs this buffer (optional operation).

Modifications to this buffer's content will cause the returned array's content to be modified, and vice versa.

Invoke the hasArray method before invoking this method in order to ensure that this buffer has an accessible backing array.

Returns

the array that backs this **Buffer** (p. 582).

Exceptions

ReadOnlyBuffer-Exception (p. 2244)	if this Buffer (p. 582) is read only.
<i>Unsupported-OperationException</i>	if the underlying store has no array.

Implements **decaf::nio::CharBuffer** (p. 790).

6.111.2.2 virtual int decaf::internal::nio::CharArrayBuffer::arrayOffset () [virtual]

Returns the offset within this buffer's backing array of the first element of the buffer (optional operation).

Invoke the hasArray method before invoking this method in order to ensure that this buffer has an accessible backing array.

Returns

The offset into the backing array where index zero starts.

Exceptions

<i>ReadOnlyBufferException</i> (p. 2244)	if this Buffer (p. 582) is read only.
<i>UnsupportedOperationException</i>	if the underlying store has no array.

Implements **decaf::nio::CharBuffer** (p. 791).

6.111.2.3 **virtual CharBuffer* decaf::internal::nio::CharArrayBuffer::asReadOnlyBuffer() const** [virtual]

Creates a new, read-only char buffer that shares this buffer's content.

The content of the new buffer will be that of this buffer. Changes to this buffer's content will be visible in the new buffer; the new buffer itself, however, will be read-only and will not allow the shared content to be modified. The two buffers' position, limit, and mark values will be independent.

If this buffer is itself read-only then this method behaves in exactly the same way as the duplicate method.

The new buffer's capacity, limit, position, and mark values will be identical to those of this buffer.

Returns

The new, read-only char buffer which the caller then owns.

Implements **decaf::nio::CharBuffer** (p. 791).

6.111.2.4 **virtual CharBuffer& decaf::internal::nio::CharArrayBuffer::compact()** [virtual]

Compacts this buffer.

The bytes between the buffer's current position and its limit, if any, are copied to the beginning of the buffer. That is, the byte at index $p = \text{position}()$ (p. 587) is copied to index zero, the byte at index $p + 1$ is copied to index one, and so forth until the byte at index $\text{limit}()$ (p. 586) - 1 is copied to index $n = \text{limit}()$ (p. 586) - 1 - p . The buffer's position is then set to $n + 1$ and its limit is set to its capacity. The mark, if defined, is discarded.

The buffer's position is set to the number of bytes copied, rather than to zero, so that an invocation of this method can be followed immediately by an invocation of another relative put method.

Returns

a reference to this **CharBuffer** (p. 785).

Exceptions

<i>ReadOnlyBuffer-Exception</i> (p. 2244)	- If this buffer is read-only
---	-------------------------------

Implements **decaf::nio::CharBuffer** (p. 792).

6.111.2.5 **virtual CharBuffer* decaf::internal::nio::CharArrayBuffer::duplicate ()**
[virtual]

Creates a new char buffer that shares this buffer's content.

The content of the new buffer will be that of this buffer. Changes to this buffer's content will be visible in the new buffer, and vice versa; the two buffers' position, limit, and mark values will be independent.

The new buffer's capacity, limit, position, and mark values will be identical to those of this buffer. The new buffer will be read-only if, and only if, this buffer is read-only.

Returns

a new char **Buffer** (p. 582) which the caller owns.

Implements **decaf::nio::CharBuffer** (p. 793).

6.111.2.6 **virtual char decaf::internal::nio::CharArrayBuffer::get ()** [virtual]

Relative get method.

Reads the character at this buffer's current position, and then increments the position.

Returns

the char at the current position.

Exceptions

<i>BufferUnderflow-Exception</i> (p. 611)	if there no more data to return
---	---------------------------------

Implements **decaf::nio::CharBuffer** (p. 793).

6.111.2.7 `virtual char decaf::internal::nio::CharArrayBuffer::get (int index) const`
`[virtual]`

Absolute get method.

Reads the char at the given index.

Parameters

<i>index</i>	The index in the Buffer (p. 582) where the char is to be read.
--------------	---

Returns

the char that is located at the given index.

Exceptions

<i>IndexOutOfBoundsException</i>	if index is not smaller than the buffer's limit or is negative.
----------------------------------	---

Implements **decaf::nio::CharBuffer** (p. 794).

6.111.2.8 `virtual bool decaf::internal::nio::CharArrayBuffer::hasArray () const`
`[inline, virtual]`

Tells whether or not this buffer is backed by an accessible char array.

If this method returns true then the array and arrayOffset methods may safely be invoked. Subclasses should override this method if they do not have a backing array as this class always returns true.

Returns

true if, and only if, this buffer is backed by an array and is not read-only

Implements **decaf::nio::CharBuffer** (p. 795).

6.111.2.9 `virtual bool decaf::internal::nio::CharArrayBuffer::isReadOnly () const`
`[inline, virtual]`

Tells whether or not this buffer is read-only.

Returns

true if, and only if, this buffer is read-only.

Implements **decaf::nio::Buffer** (p. 586).

6.111.2.10 virtual CharBuffer& decaf::internal::nio::CharArrayBuffer::put (char *value*) [virtual]

Writes the given char into this buffer at the current position, and then increments the position.

Parameters

<i>value</i>	The char value to be written.
--------------	-------------------------------

Returns

a reference to this buffer.

Exceptions

BufferOverflow-Exception (p. 609)	if this buffer's current position is not smaller than its limit
ReadOnlyBuffer-Exception (p. 2244)	if this buffer is read-only.

Implements **decaf::nio::CharBuffer** (p. 798).

6.111.2.11 virtual CharBuffer& decaf::internal::nio::CharArrayBuffer::put (int *index*, char *value*) [virtual]

Writes the given char into this buffer at the given index.

Parameters

<i>index</i>	The position in the Buffer (p. 582) to write the data.
<i>value</i>	The char to write.

Returns

a reference to this buffer.

Exceptions

IndexOutOfBoundsException	if index greater than the buffer's limit minus the size of the type being written, or index is negative.
ReadOnlyBuffer-Exception (p. 2244)	if this buffer is read-only.

Implements **decaf::nio::CharBuffer** (p. 798).

6.111.2.12 `virtual void decaf::internal::nio::CharArrayBuffer::setReadOnly (bool value) [inline, protected, virtual]`

Sets this **CharArrayBuffer** (p. 773) as Read-Only.

Parameters

<i>value</i>	Boolean value, true if this buffer is to be read-only, false otherwise.
--------------	---

6.111.2.13 `virtual CharBuffer* decaf::internal::nio::CharArrayBuffer::slice () const [virtual]`

Creates a new **CharBuffer** (p. 785) whose content is a shared subsequence of this buffer's content.

The content of the new buffer will start at this buffer's current position. Changes to this buffer's content will be visible in the new buffer, and vice versa; the two buffers' position, limit, and mark values will be independent.

The new buffer's position will be zero, its capacity and its limit will be the number of bytes remaining in this buffer, and its mark will be undefined. The new buffer will be read-only if, and only if, this buffer is read-only.

Returns

the newly create **CharBuffer** (p. 785) which the caller owns.

Implements **decaf::nio::CharBuffer** (p. 801).

6.111.2.14 `virtual lang::CharSequence* decaf::internal::nio::CharArrayBuffer::subSequence (int start, int end) const [virtual]`

Creates a new character buffer that represents the specified subsequence of this buffer, relative to the current position.

The new buffer will share this buffer's content; that is, if the content of this buffer is mutable then modifications to one buffer will cause the other to be modified. The new buffer's capacity will be that of this buffer, its position will be **position()** (p. 587) + *start*, and its limit will be **position()** (p. 587) + *end*. The new **Buffer** (p. 582) will be read-only if, and only if, this buffer is read-only.

Parameters

<i>start</i>	The index, relative to the current position, of the first character in the subsequence; must be non-negative and no larger than remaining() (p. 588).
<i>end</i>	The index, relative to the current position, of the character following the last character in the subsequence; must be no smaller than <i>start</i> and no larger than remaining() (p. 588).

Returns

The new character buffer, caller owns.

Exceptions

<i>IndexOutOfBoundsException</i>	if the preconditions on start and end fail.
----------------------------------	---

Implements **decaf::nio::CharBuffer** (p. 801).

6.111.3 Field Documentation

- 6.111.3.1 **decaf::lang::Pointer<ByteArrayAdapter>**
decaf::internal::nio::CharArrayBuffer::_array [protected]
- 6.111.3.2 **int decaf::internal::nio::CharArrayBuffer::length** [protected]
- 6.111.3.3 **int decaf::internal::nio::CharArrayBuffer::offset** [protected]
- 6.111.3.4 **bool decaf::internal::nio::CharArrayBuffer::readOnly** [protected]

The documentation for this class was generated from the following file:

- src/main/decaf/internal/nio/**CharArrayBuffer.h**

6.112 decaf::nio::CharBuffer Class Reference

This class defines four categories of operations upon character buffers:

```
#include <src/main/decaf/nio/CharBuffer.h>
```

Inheritance diagram for decaf::nio::CharBuffer:

Public Member Functions

- virtual **~CharBuffer** ()
- virtual std::string **toString** () const
- **CharBuffer & append** (char value)
Appends the specified character to this buffer.
- **CharBuffer & append** (const lang::CharSequence *value)
Appends the specified character sequence to this buffer.
- **CharBuffer & append** (const lang::CharSequence *value, int start, int end)

Appends a subsequence of the specified character sequence to this buffer. If value is Null the the string "null" is appended to the buffer.

- virtual char * **array** ()=0
Returns the character array that backs this buffer (optional operation).
- virtual int **arrayOffset** ()=0
Returns the offset within this buffer's backing array of the first element of the buffer (optional operation).
- virtual **CharBuffer** * **asReadOnlyBuffer** () const =0
Creates a new, read-only char buffer that shares this buffer's content.
- char **charAt** (int index) const
Reads the character at the given index relative to the current position.
- virtual **CharBuffer** & **compact** ()=0
Compacts this buffer.
- virtual **CharBuffer** * **duplicate** ()=0
Creates a new char buffer that shares this buffer's content.
- virtual char **get** ()=0
Relative get method.
- virtual char **get** (int index) const =0
Absolute get method.
- **CharBuffer** & **get** (std::vector< char > buffer)
Relative bulk get method.
- **CharBuffer** & **get** (char *buffer, int size, int offset, int **length**)
Relative bulk get method.
- virtual bool **hasArray** () const =0
Tells whether or not this buffer is backed by an accessible char array.
- int **length** () const
Returns the length of this character buffer.
- **CharBuffer** & **put** (**CharBuffer** &src)
This method transfers the chars remaining in the given source buffer into this buffer.
- **CharBuffer** & **put** (const char *buffer, int size, int offset, int **length**)
This method transfers chars into this buffer from the given source array.
- **CharBuffer** & **put** (std::vector< char > &buffer)
This method transfers the entire content of the given source char array into this buffer.
- virtual **CharBuffer** & **put** (char value)=0
Writes the given char into this buffer at the current position, and then increments the position.
- virtual **CharBuffer** & **put** (int index, char value)=0
Writes the given char into this buffer at the given index.
- **CharBuffer** & **put** (std::string &src, int start, int end)
Relative bulk put method (optional operation).
- **CharBuffer** & **put** (const std::string &src)
Relative bulk put method (optional operation).
- virtual int **read** (**CharBuffer** *target)
Attempts to read characters into the specified character buffer.

- virtual **lang::CharSequence** * **subSequence** (int start, int end) const =0
Creates a new character buffer that represents the specified subsequence of this buffer, relative to the current position.
- virtual **CharBuffer** * **slice** () const =0
*Creates a new **CharBuffer** (p. 785) whose content is a shared subsequence of this buffer's content.*
- virtual int **compareTo** (const **CharBuffer** &value) const
- virtual bool **equals** (const **CharBuffer** &value) const
- virtual bool **operator==** (const **CharBuffer** &value) const
- virtual bool **operator<** (const **CharBuffer** &value) const

Static Public Member Functions

- static **CharBuffer** * **allocate** (int capacity)
Allocates a new character buffer.
- static **CharBuffer** * **wrap** (char *array, int size, int offset, int length)
*Wraps the passed buffer with a new **CharBuffer** (p. 785).*
- static **CharBuffer** * **wrap** (std::vector< char > &buffer)
*Wraps the passed STL char Vector in a **CharBuffer** (p. 785).*

Protected Member Functions

- **CharBuffer** (int capacity)
*Creates a **CharBuffer** (p. 785) object that has its backing array allocated internally and is then owned and deleted when this object is deleted.*

6.112.1 Detailed Description

This class defines four categories of operations upon character buffers:

o Absolute and relative get and put methods that read and write single characters; o Relative bulk get methods that transfer contiguous sequences of characters from this buffer into an array; and o Relative bulk put methods that transfer contiguous sequences of characters from a character array, a string, or some other character buffer into this buffer. o Methods for compacting, duplicating, and slicing a character buffer.

Character buffers can be created either by allocation, which allocates space for the buffer's content, by wrapping an existing character array or string into a buffer, or by creating a view of an existing byte buffer

This class implements the `CharSequence` interface so that character buffers may be used wherever character sequences are accepted, for example in the regular-expression package `decaf.util.regex`.

Methods in this class that do not otherwise have a value to return are specified to return the buffer upon which they are invoked. This allows method invocations to be chained. The sequence of statements

```
cb.put("text/"); cb.put(subtype); cb.put("; charset="); cb.put(enc);
```

can, for example, be replaced by the single statement

```
cb.put("text/").put(subtype).put("; charset=").put(enc);
```

6.112.2 Constructor & Destructor Documentation

6.112.2.1 `decaf::nio::CharBuffer::CharBuffer (int capacity)` [protected]

Creates a **CharBuffer** (p. 785) object that has its backing array allocated internally and is then owned and deleted when this object is deleted.

The array is initially created with all elements initialized to zero.

Parameters

<i>capacity</i>	The size of the array, this is the limit we read and write to.
-----------------	--

Exceptions

<i>IllegalArgument-Exception</i>	if capacity is negative.
----------------------------------	--------------------------

6.112.2.2 `virtual decaf::nio::CharBuffer::~CharBuffer ()` [inline, virtual]

6.112.3 Member Function Documentation

6.112.3.1 `static CharBuffer* decaf::nio::CharBuffer::allocate (int capacity)` [static]

Allocates a new character buffer.

The new buffer's position will be zero, its limit will be its capacity, and its mark will be undefined. It will have a backing array, and its array offset will be zero.

Parameters

<i>capacity</i>	The size of the Char buffer in chars (1 byte).
-----------------	--

Returns

the **CharBuffer** (p. 785) that was allocated, caller owns.

Exceptions

<i>IndexOutOfBoundsException</i>	if capacity is negative.
----------------------------------	--------------------------

6.112.3.2 CharBuffer& decaf::nio::CharBuffer::append (char value) [virtual]

Appends the specified character to this buffer.

Parameters

<i>value</i>	The char to append.
--------------	---------------------

Returns

a reference to this modified **CharBuffer** (p. 785).

Exceptions

BufferOverflow-Exception (p. 609)	if there is no more space
ReadOnlyBuffer-Exception (p. 2244)	if this Buffer (p. 582) is read only.

Implements **decaf::lang::Appendable** (p. 443).

6.112.3.3 CharBuffer& decaf::nio::CharBuffer::append (const lang::CharSequence * value) [virtual]

Appends the specified character sequence to this buffer.

If value is Null the the string "null" is appended to the buffer.

Parameters

<i>value</i>	The CharSequence to append.
--------------	-----------------------------

Returns

a reference to this modified **CharBuffer** (p. 785)

Exceptions

BufferOverflow-Exception (p. 609)	if there is no more space
---	---------------------------

<i>ReadOnlyBuffer-Exception</i> (p. 2244)	if this Buffer (p. 582) is read only.
---	--

Implements **decaf::lang::Appendable** (p. 443).

6.112.3.4 **CharBuffer& decaf::nio::CharBuffer::append** (**const lang::CharSequence** * *value*, **int** *start*, **int** *end*) [virtual]

Appends a subsequence of the specified character sequence to this buffer. If *value* is Null the the string "null" is appended to the buffer.

Parameters

<i>value</i>	The CharSequence to append.
<i>start</i>	The index to start appending from.
<i>end</i>	The index to append to.

Returns

a reference to this modified **CharBuffer** (p. 785).

Exceptions

<i>BufferOverflow-Exception</i> (p. 609)	if there is no more space
<i>ReadOnlyBuffer-Exception</i> (p. 2244)	if this Buffer (p. 582) is read only.
<i>IndexOutOfBoundsException</i>	if <i>start</i> > <i>end</i> , or > length of sequence.

Implements **decaf::lang::Appendable** (p. 443).

6.112.3.5 **virtual char* decaf::nio::CharBuffer::array** () [pure virtual]

Returns the character array that backs this buffer (optional operation).

Modifications to this buffer's content will cause the returned array's content to be modified, and vice versa.

Invoke the `hasArray` method before invoking this method in order to ensure that this buffer has an accessible backing array.

Returns

the array that backs this **Buffer** (p. 582).

Exceptions

<i>ReadOnlyBuffer-Exception</i> (p. 2244)	if this Buffer (p. 582) is read only.
<i>Unsupported-Operation</i> Exception	if the underlying store has no array.

Implemented in **decaf::internal::nio::CharArrayBuffer** (p. 779).

6.112.3.6 virtual int decaf::nio::CharBuffer::arrayOffset () [pure virtual]

Returns the offset within this buffer's backing array of the first element of the buffer (optional operation).

Invoke the `hasArray` method before invoking this method in order to ensure that this buffer has an accessible backing array.

Returns

The offset into the backing array where index zero starts.

Exceptions

<i>ReadOnlyBuffer-Exception</i> (p. 2244)	if this Buffer (p. 582) is read only.
<i>Unsupported-Operation</i> Exception	if the underlying store has no array.

Implemented in **decaf::internal::nio::CharArrayBuffer** (p. 779).

6.112.3.7 virtual CharBuffer* decaf::nio::CharBuffer::asReadOnlyBuffer () const [pure virtual]

Creates a new, read-only char buffer that shares this buffer's content.

The content of the new buffer will be that of this buffer. Changes to this buffer's content will be visible in the new buffer; the new buffer itself, however, will be read-only and will not allow the shared content to be modified. The two buffers' position, limit, and mark values will be independent.

If this buffer is itself read-only then this method behaves in exactly the same way as the `duplicate` method.

The new buffer's capacity, limit, position, and mark values will be identical to those of

this buffer.

Returns

The new, read-only char buffer which the caller then owns.

Implemented in **decaf::internal::nio::CharArrayBuffer** (p. 780).

6.112.3.8 `char decaf::nio::CharBuffer::charAt (int index) const` [virtual]

Reads the character at the given index relative to the current position.

Parameters

<i>index</i>	- The index of the character to be read relative to position
--------------	--

Returns

The character at index **position()** (p. 587) + index.

Exceptions

<i>IndexOutOfBoundsException</i>	if the index + the current position exceeds the size of the buffer or the index is negative.
----------------------------------	--

Implements **decaf::lang::CharSequence** (p. 804).

6.112.3.9 `virtual CharBuffer& decaf::nio::CharBuffer::compact ()` [pure virtual]

Compacts this buffer.

The bytes between the buffer's current position and its limit, if any, are copied to the beginning of the buffer. That is, the byte at index $p = \mathbf{position}()$ (p. 587) is copied to index zero, the byte at index $p + 1$ is copied to index one, and so forth until the byte at index $\mathbf{limit}()$ (p. 586) - 1 is copied to index $n = \mathbf{limit}()$ (p. 586) - 1 - p . The buffer's position is then set to $n+1$ and its limit is set to its capacity. The mark, if defined, is discarded.

The buffer's position is set to the number of bytes copied, rather than to zero, so that an invocation of this method can be followed immediately by an invocation of another relative put method.

Returns

a reference to this **CharBuffer** (p. 785).

Exceptions

<i>ReadOnlyBuffer-Exception</i> (p. 2244)	- If this buffer is read-only
---	-------------------------------

Implemented in **decaf::internal::nio::CharArrayBuffer** (p. 780).

6.112.3.10 `virtual int decaf::nio::CharBuffer::compareTo (const CharBuffer & value)
const [virtual]`

6.112.3.11 `virtual CharBuffer* decaf::nio::CharBuffer::duplicate () [pure
virtual]`

Creates a new char buffer that shares this buffer's content.

The content of the new buffer will be that of this buffer. Changes to this buffer's content will be visible in the new buffer, and vice versa; the two buffers' position, limit, and mark values will be independent.

The new buffer's capacity, limit, position, and mark values will be identical to those of this buffer. The new buffer will be read-only if, and only if, this buffer is read-only.

Returns

a new char **Buffer** (p. 582) which the caller owns.

Implemented in **decaf::internal::nio::CharArrayBuffer** (p. 781).

6.112.3.12 `virtual bool decaf::nio::CharBuffer::equals (const CharBuffer & value)
const [virtual]`

6.112.3.13 `virtual char decaf::nio::CharBuffer::get () [pure virtual]`

Relative get method.

Reads the character at this buffer's current position, and then increments the position.

Returns

the char at the current position.

Exceptions

<i>BufferUnderflow-Exception</i> (p. 611)	if there no more data to return
---	---------------------------------

Implemented in **decaf::internal::nio::CharArrayBuffer** (p. 781).

6.112.3.14 `virtual char decaf::nio::CharBuffer::get (int index) const` [pure virtual]

Absolute get method.

Reads the char at the given index.

Parameters

<i>index</i>	The index in the Buffer (p. 582) where the char is to be read.
--------------	---

Returns

the char that is located at the given index.

Exceptions

<i>IndexOutOfBoundsException</i>	if index is not smaller than the buffer's limit or is negative.
----------------------------------	---

Implemented in **decaf::internal::nio::CharArrayBuffer** (p. 782).

6.112.3.15 `CharBuffer& decaf::nio::CharBuffer::get (std::vector< char > buffer)`

Relative bulk get method.

This method transfers chars from this buffer into the given destination vector. An invocation of this method of the form `src.get(a)` behaves in exactly the same way as the invocation. The vector must be sized to the amount of data that is to be read, that is to say, the caller should call `buffer.resize(N)` before calling this get method.

Returns

a reference to this **CharBuffer** (p. 785).

Exceptions

<i>BufferUnderflow-Exception</i> (p. 611)	if there are fewer than length chars remaining in this buffer.
--	--

6.112.3.16 `CharBuffer& decaf::nio::CharBuffer::get (char * buffer, int size, int offset, int length)`

Relative bulk get method.

This method transfers chars from this buffer into the given destination array. If there are fewer chars remaining in the buffer than are required to satisfy the request, that is, if

length > **remaining()** (p. 588), then no bytes are transferred and a **BufferUnderflow-Exception** (p. 611) is thrown.

Otherwise, this method copies length chars from this buffer into the given array, starting at the current position of this buffer and at the given offset in the array. The position of this buffer is then incremented by length.

Parameters

<i>buffer</i>	The pointer to an allocated buffer to fill.
<i>size</i>	The size of the buffer passed.
<i>offset</i>	The position in the buffer to start filling.
<i>length</i>	The amount of data to put in the passed buffer.

Returns

a reference to this **Buffer** (p. 582).

Exceptions

BufferUnderflow-Exception (p. 611)	if there are fewer than length chars remaining in this buffer
<i>NullPointerException</i>	if the passed buffer is null.
<i>IndexOutOfBoundsException</i>	if the preconditions of size, offset, or length are not met.

6.112.3.17 `virtual bool decaf::nio::CharBuffer::hasArray () const` `[pure virtual]`

Tells whether or not this buffer is backed by an accessible char array.

If this method returns true then the array and arrayOffset methods may safely be invoked. Subclasses should override this method if they do not have a backing array as this class always returns true.

Returns

true if, and only if, this buffer is backed by an array and is not read-only

Implemented in **decaf::internal::nio::CharArrayBuffer** (p. 782).

6.112.3.18 `int decaf::nio::CharBuffer::length () const` `[inline, virtual]`

Returns the length of this character buffer.

Returns

the length of this buffer from the position to the limit.

Implements **decaf::lang::CharSequence** (p. 804).

6.112.3.19 `virtual bool decaf::nio::CharBuffer::operator< (const CharBuffer & value) const`
[virtual]

6.112.3.20 `virtual bool decaf::nio::CharBuffer::operator== (const CharBuffer & value) const`
[virtual]

6.112.3.21 `CharBuffer& decaf::nio::CharBuffer::put (CharBuffer & src)`

This method transfers the chars remaining in the given source buffer into this buffer.

If there are more chars remaining in the source buffer than in this buffer, that is, if `src.remaining() > remaining()` (p. 588), then no chars are transferred and a **BufferOverflowException** (p. 609) is thrown.

Otherwise, this method copies `n = src.remaining()` chars from the given buffer into this buffer, starting at each buffer's current position. The positions of both buffers are then incremented by `n`.

Parameters

<code>src</code>	- the buffer to take chars from an place in this one.
------------------	---

Returns

a reference to this buffer.

Exceptions

BufferOverflowException (p. 609)	if there is insufficient space in this buffer for the remaining chars in the source buffer.
IllegalArgumentException	if the source buffer is this buffer.
ReadOnlyBufferException (p. 2244)	if this buffer is read-only.

6.112.3.22 `CharBuffer& decaf::nio::CharBuffer::put (const char * buffer, int size, int offset, int length)`

This method transfers chars into this buffer from the given source array.

If there are more chars to be copied from the array than remain in this buffer, that is,

if `length > remaining()` (p. 588), then no chars are transferred and a **BufferOverflow-Exception** (p. 609) is thrown.

Otherwise, this method copies `length` bytes from the given array into this buffer, starting at the given offset in the array and at the current position of this buffer. The position of this buffer is then incremented by `length`.

Parameters

<i>buffer</i>	The array from which chars are to be read.
<i>size</i>	The size of the buffer passed.
<i>offset</i>	The offset within the array of the first char to be read.
<i>length</i>	The number of chars to be read from the given array.

Returns

a reference to this buffer.

Exceptions

BufferOverflow-Exception (p. 609)	if there is insufficient space in this buffer
ReadOnlyBuffer-Exception (p. 2244)	if this buffer is read-only
<i>NullPointerException</i>	if the passed buffer is null.
<i>IndexOutOfBoundsException</i>	if the preconditions of size, offset, or length are not met.

6.112.3.23 CharBuffer& decaf::nio::CharBuffer::put (std::vector< char > &buffer)

This method transfers the entire content of the given source char array into this buffer.

This is the same as calling `put(&buffer[0], 0, buffer.size())`.

Parameters

<i>buffer</i>	The buffer whose contents are copied to this CharBuffer (p. 785).
---------------	--

Returns

a reference to this buffer.

Exceptions

BufferOverflow-Exception (p. 609)	if there is insufficient space in this buffer.
---	--

<i>ReadOnlyBuffer-Exception</i> (p. 2244)	if this buffer is read-only.
---	------------------------------

6.112.3.24 `virtual CharBuffer& decaf::nio::CharBuffer::put (char value)` [pure virtual]

Writes the given char into this buffer at the current position, and then increments the position.

Parameters

<i>value</i>	The char value to be written.
--------------	-------------------------------

Returns

a reference to this buffer.

Exceptions

<i>BufferOverflow-Exception</i> (p. 609)	if this buffer's current position is not smaller than its limit
<i>ReadOnlyBuffer-Exception</i> (p. 2244)	if this buffer is read-only.

Implemented in **decaf::internal::nio::CharArrayBuffer** (p. 783).

6.112.3.25 `virtual CharBuffer& decaf::nio::CharBuffer::put (int index, char value)` [pure virtual]

Writes the given char into this buffer at the given index.

Parameters

<i>index</i>	The position in the Buffer (p. 582) to write the data.
<i>value</i>	The char to write.

Returns

a reference to this buffer.

Exceptions

<i>IndexOutOfBoundsException</i>	if index greater than the buffer's limit minus the size of the type being written, or index is negative.
ReadOnlyBufferException (p. 2244)	if this buffer is read-only.

Implemented in `decaf::internal::nio::CharArrayBuffer` (p. 783).

6.112.3.26 `CharBuffer& decaf::nio::CharBuffer::put (std::string & src, int start, int end)`

Relative bulk put method (optional operation).

This method transfers characters from the given string into this buffer. If there are more characters to be copied from the string than remain in this buffer, that is, if `end - start > remaining()` (p. 588), then no characters are transferred and a **BufferOverflowException** (p. 609) is thrown.

Returns

a reference to this buffer

Otherwise, this method copies `n = end - start` characters from the given string into this buffer, starting at the given start index and at the current position of this buffer. The position of this buffer is then incremented by `n`.

Parameters

<i>src</i>	The string to copy from.
<i>start</i>	The position in <i>src</i> to start from.
<i>end</i>	The position in <i>src</i> to stop at.

Returns

a reference to this **CharBuffer** (p. 785).

Exceptions

BufferOverflowException (p. 609)	if this buffer's current position is not
<i>IndexOutOfBoundsException</i>	if index greater than the buffer's limit minus the size of the type being written.
ReadOnlyBufferException (p. 2244)	if this buffer is read-only

6.112.3.27 CharBuffer& decaf::nio::CharBuffer::put (const std::string & *src*)

Relative bulk put method (optional operation).

This method transfers the entire content of the given source string into this buffer. An invocation of this method of the form `dst.put(s)` behaves in exactly the same way as the invocation.

Parameters

<i>src</i>	The string to copy from.
------------	--------------------------

Returns

a reference to this **CharBuffer** (p. 785).

Exceptions

BufferOverflow-Exception (p. 609)	if this buffer's current position is not.
ReadOnlyBuffer-Exception (p. 2244)	if this buffer is read-only.

6.112.3.28 virtual int decaf::nio::CharBuffer::read (CharBuffer * *target*) [virtual]

Attempts to read characters into the specified character buffer.

The buffer is used as a repository of characters as-is: the only changes made are the results of a put operation. No flipping or rewinding of the buffer is performed.

Parameters

<i>target</i>	The buffer to read characters into
---------------	------------------------------------

Returns

The number of characters added to the buffer, or `string::npos` if this source of characters is at its end

Exceptions

<i>NullPointerException</i>	if <i>target</i> is Null.
<i>IllegalArgumentException</i>	if <i>target</i> is this CharBuffer (p. 785).
ReadOnlyBuffer-Exception (p. 2244)	if this buffer is in read-only mode.

6.112.3.29 `virtual CharBuffer* decaf::nio::CharBuffer::slice () const` [pure virtual]

Creates a new **CharBuffer** (p. 785) whose content is a shared subsequence of this buffer's content.

The content of the new buffer will start at this buffer's current position. Changes to this buffer's content will be visible in the new buffer, and vice versa; the two buffers' position, limit, and mark values will be independent.

The new buffer's position will be zero, its capacity and its limit will be the number of bytes remaining in this buffer, and its mark will be undefined. The new buffer will be read-only if, and only if, this buffer is read-only.

Returns

the newly create **CharBuffer** (p. 785) which the caller owns.

Implemented in **decaf::internal::nio::CharArrayBuffer** (p. 784).

6.112.3.30 `virtual lang::CharSequence* decaf::nio::CharBuffer::subSequence (int start, int end) const` [pure virtual]

Creates a new character buffer that represents the specified subsequence of this buffer, relative to the current position.

The new buffer will share this buffer's content; that is, if the content of this buffer is mutable then modifications to one buffer will cause the other to be modified. The new buffer's capacity will be that of this buffer, its position will be **position()** (p. 587) + start, and its limit will be **position()** (p. 587) + end. The new **Buffer** (p. 582) will be read-only if, and only if, this buffer is read-only.

Parameters

<i>start</i>	The index, relative to the current position, of the first character in the subsequence; must be non-negative and no larger than remaining() (p. 588).
<i>end</i>	The index, relative to the current position, of the character following the last character in the subsequence; must be no smaller than start and no larger than remaining() (p. 588).

Returns

The new character buffer, caller owns.

Exceptions

<i>IndexOutOfBoundsException</i>	if the preconditions on start and end fail.
----------------------------------	---

Implements **decaf::lang::CharSequence** (p. 804).

Implemented in **decaf::internal::nio::CharArrayBuffer** (p. 784).

6.112.3.31 **virtual std::string decaf::nio::CharBuffer::toString () const** [virtual]

Returns

a std::string describing this object

Implements **decaf::lang::CharSequence** (p. 805).

6.112.3.32 **static CharBuffer* decaf::nio::CharBuffer::wrap (char * array, int size, int offset, int length)** [static]

Wraps the passed buffer with a new **CharBuffer** (p. 785).

The new buffer will be backed by the given char array; that is, modifications to the buffer will cause the array to be modified and vice versa. The new buffer's capacity will be array.length, its position will be offset, its limit will be offset + length, and its mark will be undefined. Its backing array will be the given array, and its array offset will be zero.

Parameters

<i>array</i>	The array that will back the new buffer.
<i>size</i>	The size of the array passed in.
<i>offset</i>	The offset of the subarray to be used.
<i>length</i>	The length of the subarray to be used.

Returns

a new **CharBuffer** (p. 785) that is backed by buffer, caller owns.

Exceptions

<i>NullPointerException</i>	if the array pointer is Null.
<i>IndexOutOfBoundsException</i>	if capacity is negative.

6.112.3.33 **static CharBuffer* decaf::nio::CharBuffer::wrap (std::vector< char > & buffer)** [static]

Wraps the passed STL char Vector in a **CharBuffer** (p. 785).

The new buffer will be backed by the given char array; modifications to the buffer will cause the array to be modified and vice versa. The new buffer's capacity and limit will be buffer.size(), its position will be zero, and its mark will be undefined. Its backing array will be the given array, and its array offset will be zero.

Parameters

<i>buffer</i>	The vector that will back the new buffer, the vector must have been sized to the desired size already by calling <code>vector.resize(N)</code> .
---------------	--

Returns

a new **CharBuffer** (p. 785) that is backed by `buffer`, caller owns.

The documentation for this class was generated from the following file:

- `src/main/decaf/nio/CharBuffer.h`

6.113 decaf::lang::CharSequence Class Reference

A **CharSequence** (p. 803) is a readable sequence of char values.

```
#include <src/main/decaf/lang/CharSequence.h>
```

Inheritance diagram for `decaf::lang::CharSequence`:

Public Member Functions

- virtual `~CharSequence()`
- virtual `int length()` `const =0`
- virtual `char charAt(int index)` `const =0`
Returns the Char at the specified index so long as the index is not greater than the length of the sequence.
- virtual `CharSequence * subSequence(int start, int end)` `const =0`
*Returns a new **CharSequence** (p. 803) that is a subsequence of this sequence.*
- virtual `std::string toString()` `const =0`

6.113.1 Detailed Description

A **CharSequence** (p. 803) is a readable sequence of char values.

This interface provides uniform, read-only access to many different kinds of char sequences.

This interface does not define that a **CharSequence** (p. 803) should implement comparable, it is therefore up to the derived classes that implement this interface to define equality, which implies that comparison of two `CharSequences` does not have a contract on equality.

6.113.2 Constructor & Destructor Documentation

6.113.2.1 `virtual decaf::lang::CharSequence::~CharSequence () [inline, virtual]`

6.113.3 Member Function Documentation

6.113.3.1 `virtual char decaf::lang::CharSequence::charAt (int index) const [pure virtual]`

Returns the Char at the specified index so long as the index is not greater than the length of the sequence.

Parameters

<i>index</i>	The position to return the char at.
--------------	-------------------------------------

Returns

the char at the given position.

Exceptions

<i>IndexOutOfBoundsException</i>	if index is > than length() (p. 804) or negative
----------------------------------	---

Implemented in **decaf::nio::CharBuffer** (p. 792), and **decaf::lang::String** (p. 2624).

6.113.3.2 `virtual int decaf::lang::CharSequence::length () const [pure virtual]`

Returns

the length of the underlying character sequence.

Implemented in **decaf::nio::CharBuffer** (p. 795), and **decaf::lang::String** (p. 2624).

6.113.3.3 `virtual CharSequence* decaf::lang::CharSequence::subSequence (int start, int end) const [pure virtual]`

Returns a new **CharSequence** (p. 803) that is a subsequence of this sequence.

The subsequence starts with the char value at the specified index and ends with the char value at index end - 1. The length (in chars) of the returned sequence is end - start, so if start == end then an empty sequence is returned.

Parameters

<i>start</i>	The start index, inclusive.
<i>end</i>	The end index, exclusive.

Returns

a new **CharSequence** (p. 803)

Exceptions

<i>IndexOutOfBoundsException</i>	if start or end > length() (p. 804) or start or end are negative.
----------------------------------	--

Implemented in **decaf::nio::CharBuffer** (p. 801), **decaf::internal::nio::CharArrayBuffer** (p. 784), and **decaf::lang::String** (p. 2625).

6.113.3.4 `virtual std::string decaf::lang::CharSequence::toString () const [pure virtual]`

Returns

the string representation of this **CharSequence** (p. 803)

Implemented in **decaf::lang::String** (p. 2625), and **decaf::nio::CharBuffer** (p. 802).

The documentation for this class was generated from the following file:

- src/main/decaf/lang/**CharSequence.h**

6.114 decaf::util::zip::CheckedInputStream Class Reference

An implementation of a FilterInputStream that will maintain a **Checksum** (p. 810) of the bytes read, the **Checksum** (p. 810) can then be used to verify the integrity of the input stream.

```
#include <src/main/decaf/util/zip/CheckedInputStream.h>
```

Inheritance diagram for decaf::util::zip::CheckedInputStream:

Public Member Functions

- **CheckedInputStream** (InputStream *inputStream, Checksum *sum, bool own=false)
Create a new instance of a **CheckedInputStream** (p. 805).
- virtual ~**CheckedInputStream** ()

- **Checksum** * **getChecksum** () const

Returns a Pointer to the **Checksum** (p. 810) that is in use by this **CheckedInputStream** (p. 805).

- virtual long long **skip** (long long num)

Skips over and discards *n* bytes of data from this input stream.

The skip method may, for a variety of reasons, end up skipping over some smaller number of bytes, possibly 0. This may result from any of a number of conditions; reaching end of file before *n* bytes have been skipped is only one possibility. The actual number of bytes skipped is returned.

The skip method of **InputStream** (p. 1464) creates a byte array and then repeatedly reads into it until *num* bytes have been read or the end of the stream has been reached. Subclasses are encouraged to provide a more efficient implementation of this method.

Parameters

num	The number of bytes to skip.
-----	------------------------------

Returns

total bytes skipped

Exceptions

IOException (p. 1545)	if an I/O error occurs.
UnsupportedOperationException	if the concrete stream class does not support skipping bytes.

Protected Member Functions

- virtual int **doReadByte** ()
- virtual int **doReadArrayBounded** (unsigned char *buffer, int size, int offset, int length)

6.114.1 Detailed Description

An implementation of a **FilterInputStream** that will maintain a **Checksum** (p. 810) of the bytes read, the **Checksum** (p. 810) can then be used to verify the integrity of the input stream.

Since

1.0

6.114.2 Constructor & Destructor Documentation

- 6.114.2.1 **decaf::util::zip::CheckedInputStream::CheckedInputStream** (**InputStream** * *inputStream*, **Checksum** * *sum*, bool *own* = false)

Create a new instance of a **CheckedInputStream** (p. 805).

Parameters

<i>inputStream</i>	The <code>InputStream</code> instance to Wrap.
<i>sum</i>	The Checksum (p. 810) instance to use (does not take ownership of the Pointer).
<i>own</i>	Indicates if this filer should take ownership of the <code>InputStream</code> .

Exceptions

<i>NullPointerException</i>	if the Checksum (p. 810) pointer is NULL.
-----------------------------	--

6.114.2.2 virtual `decaf::util::zip::CheckedInputStream::~~CheckedInputStream ()`
[virtual]

6.114.3 Member Function Documentation

6.114.3.1 virtual `int decaf::util::zip::CheckedInputStream::doReadArrayBounded (unsigned char * buffer, int size, int offset, int length)` [protected, virtual]

Reimplemented from **decaf::io::FilterInputStream** (p. 1338).

6.114.3.2 virtual `int decaf::util::zip::CheckedInputStream::doReadByte ()`
[protected, virtual]

Reimplemented from **decaf::io::FilterInputStream** (p. 1338).

6.114.3.3 **Checksum*** `decaf::util::zip::CheckedInputStream::getChecksum ()`
`const` [inline]

Returns a Pointer to the **Checksum** (p. 810) that is in use by this **CheckedInputStream** (p. 805).

Returns

the pointer to the **Checksum** (p. 810) instance that is in use by this object.

6.114.3.4 virtual `long long decaf::util::zip::CheckedInputStream::skip (long long num)` [virtual]

Skips over and discards *n* bytes of data from this input stream.

The skip method may, for a variety of reasons, end up skipping over some smaller number of bytes, possibly 0. This may result from any of a number of conditions; reaching end of file before *n* bytes have been skipped is only one possibility. The actual number of bytes skipped is returned.

The skip method of **InputStream** (p. 1464) creates a byte array and then repeatedly reads into it until num bytes have been read or the end of the stream has been reached. Subclasses are encouraged to provide a more efficient implementation of this method.

Parameters

<i>num</i>	The number of bytes to skip.
------------	------------------------------

Returns

total bytes skipped

Exceptions

<i>IOException</i> (p. 1545)	if an I/O error occurs.
<i>UnsupportedOperationException</i>	if the concrete stream class does not support skipping bytes.

Adds the skipped bytes into the **Checksum** (p. 810).

Reimplemented from **decaf::io::FilterInputStream** (p. 1340).

The documentation for this class was generated from the following file:

- src/main/decaf/util/zip/**CheckedInputStream.h**

6.115 decaf::util::zip::CheckedOutputStream Class Reference

An implementation of a **FilterOutputStream** that will maintain a **Checksum** (p. 810) of the bytes written, the **Checksum** (p. 810) can then be used to verify the integrity of the output stream.

```
#include <src/main/decaf/util/zip/CheckedOutputStream.h>
```

Inheritance diagram for decaf::util::zip::CheckedOutputStream:

Public Member Functions

- **CheckedOutputStream** (decaf::io::OutputStream *outputStream, Checksum *sum, bool own=false)
Create a new instance of a **CheckedOutputStream** (p. 808).
- virtual ~**CheckedOutputStream** ()
- **Checksum** * **getChecksum** () const

Protected Member Functions

- virtual void **doWriteByte** (unsigned char value)
- virtual void **doWriteArrayBounded** (const unsigned char *buffer, int size, int offset, int length)

6.115.1 Detailed Description

An implementation of a `FilterOutputStream` that will maintain a **Checksum** (p. 810) of the bytes written, the **Checksum** (p. 810) can then be used to verify the integrity of the output stream.

Since

1.0

6.115.2 Constructor & Destructor Documentation

6.115.2.1 `decaf::util::zip::CheckedOutputStream::CheckedOutputStream (decaf::io::OutputStream * outputStream, Checksum * sum, bool own = false)`

Create a new instance of a **CheckedOutputStream** (p. 808).

Parameters

<i>output-Stream</i>	The <code>OutputStream</code> instance to Wrap.
<i>sum</i>	The Checksum (p. 810) instance to use (does not take ownership of the Pointer).
<i>own</i>	Indicates if this filer should take ownership of the <code>InputStream</code> .

Exceptions

<i>NullPointerException</i>	if the Checksum (p. 810) pointer is NULL.
-----------------------------	--

6.115.2.2 `virtual decaf::util::zip::CheckedOutputStream::~~CheckedOutputStream () [virtual]`

6.115.3 Member Function Documentation

6.115.3.1 `virtual void decaf::util::zip::CheckedOutputStream::doWriteArrayBounded (const unsigned char * buffer, int size, int offset, int length) [protected, virtual]`

Reimplemented from **decaf::io::FilterOutputStream** (p. 1343).

6.115.3.2 virtual void **decaf::util::zip::CheckedOutputStream::doWriteByte** (unsigned char *value*) [protected, virtual]

Reimplemented from **decaf::io::FilterOutputStream** (p. 1343).

6.115.3.3 **Checksum*** **decaf::util::zip::CheckedOutputStream::getChecksum** () const [inline]

Returns

a pointer to the **Checksum** (p. 810) instance in use by this object.

The documentation for this class was generated from the following file:

- src/main/decaf/util/zip/**CheckedOutputStream.h**

6.116 decaf::util::zip::Checksum Class Reference

An interface used to represent **Checksum** (p. 810) values in the Zip package.

```
#include <src/main/decaf/util/zip/Checksum.h>
```

Inheritance diagram for **decaf::util::zip::Checksum**:

Public Member Functions

- virtual **~Checksum** ()
- virtual long long **getValue** () const =0
- virtual void **reset** ()=0
Reset the checksum to its initial value.
- virtual void **update** (const std::vector< unsigned char > &buffer)=0
Updates the current checksum with the specified vector of bytes.
- virtual void **update** (const std::vector< unsigned char > &buffer, int offset, int length)=0
Updates the current checksum with the specified array of bytes.
- virtual void **update** (const unsigned char *buffer, int size, int offset, int length)=0
Updates the current checksum with the specified array of bytes.
- virtual void **update** (int byte)=0
Updates the current checksum with the specified byte value.

6.116.1 Detailed Description

An interface used to represent **Checksum** (p. 810) values in the Zip package.

Since

1.0

6.116.2 Constructor & Destructor Documentation

6.116.2.1 `virtual decaf::util::zip::Checksum::~Checksum () [inline, virtual]`

6.116.3 Member Function Documentation

6.116.3.1 `virtual long long decaf::util::zip::Checksum::getValue () const [pure virtual]`

Returns

the current checksum value.

Implemented in **decaf::util::zip::Adler32** (p. 440), and **decaf::util::zip::CRC32** (p. 1066).

6.116.3.2 `virtual void decaf::util::zip::Checksum::reset () [pure virtual]`

Reset the checksum to its initial value.

Implemented in **decaf::util::zip::Adler32** (p. 440), and **decaf::util::zip::CRC32** (p. 1066).

6.116.3.3 `virtual void decaf::util::zip::Checksum::update (const std::vector< unsigned char > & buffer) [pure virtual]`

Updates the current checksum with the specified vector of bytes.

Parameters

<i>buffer</i>	The buffer to read the updated bytes from.
---------------	--

Implemented in **decaf::util::zip::Adler32** (p. 440), and **decaf::util::zip::CRC32** (p. 1067).

6.116.3.4 `virtual void decaf::util::zip::Checksum::update (const std::vector< unsigned char > & buffer, int offset, int length) [pure virtual]`

Updates the current checksum with the specified array of bytes.

Parameters

<i>buffer</i>	The buffer to read the updated bytes from.
<i>offset</i>	The position in the buffer to start reading.
<i>length</i>	The amount of data to read from the byte buffer.

Exceptions

<i>IndexOutOfBoundsException</i>	if $\text{offset} + \text{length} > \text{size of the buffer}$.
----------------------------------	--

Implemented in **decaf::util::zip::Adler32** (p. 440), and **decaf::util::zip::CRC32** (p. 1067).

6.116.3.5 `virtual void decaf::util::zip::Checksum::update (const unsigned char * buffer, int size, int offset, int length) [pure virtual]`

Updates the current checksum with the specified array of bytes.

Parameters

<i>buffer</i>	The buffer to read the updated bytes from.
<i>size</i>	The size of the passed buffer.
<i>offset</i>	The position in the buffer to start reading.
<i>length</i>	The amount of data to read from the byte buffer.

Exceptions

<i>NullPointerException</i>	if the passed buffer is NULL.
<i>IndexOutOfBoundsException</i>	if $\text{offset} + \text{length} > \text{size of the buffer}$.

Implemented in **decaf::util::zip::Adler32** (p. 441), and **decaf::util::zip::CRC32** (p. 1067).

6.116.3.6 `virtual void decaf::util::zip::Checksum::update (int byte) [pure virtual]`

Updates the current checksum with the specified byte value.

Parameters

<i>byte</i>	The byte value to update the current Checksum (p. 810) with (0..255).
-------------	--

Implemented in **decaf::util::zip::Adler32** (p. 441), and **decaf::util::zip::CRC32** (p. 1068).

The documentation for this class was generated from the following file:

- src/main/decaf/util/zip/Checksum.h

6.117 decaf::lang::exceptions::ClassCastException Class Reference

```
#include <src/main/decaf/lang/exceptions/ClassCastException.h>
```

Inheritance diagram for decaf::lang::exceptions::ClassCastException:

Public Member Functions

- **ClassCastException** () throw ()
Default Constructor.
- **ClassCastException** (const **Exception** &ex) throw ()
*Conversion Constructor from some other **Exception** (p. 1279).*
- **ClassCastException** (const **ClassCastException** &ex) throw ()
Copy Constructor.
- **ClassCastException** (const std::exception *cause) throw ()
Constructor.
- **ClassCastException** (const char *file, const int lineNumber, const char *msg,...) throw ()
Constructor - Initializes the file name and line number where this message occurred.
- **ClassCastException** (const char *file, const int lineNumber, const std::exception *cause, const char *msg,...) throw ()
Constructor - Initializes the file name and line number where this message occurred.
- virtual **ClassCastException** * clone () const
Clones this exception.
- virtual ~**ClassCastException** () throw ()

6.117.1 Constructor & Destructor Documentation

6.117.1.1 decaf::lang::exceptions::ClassCastException::ClassCastException ()
throw () [inline]

Default Constructor.

6.117.1.2 decaf::lang::exceptions::ClassCastException::ClassCastException (const **Exception** & ex) throw () [inline]

Conversion Constructor from some other **Exception** (p. 1279).

Parameters

<i>ex</i>	The Exception (p. 1279) whose data is to be copied into this one.
-----------	--

6.117.1.3 **decaf::lang::exceptions::ClassCastException::ClassCastException (**
const ClassCastException & *ex*) throw () `[inline]`

Copy Constructor.

Parameters

<i>ex</i>	The Exception (p. 1279) whose data is to be copied into this one.
-----------	--

6.117.1.4 **decaf::lang::exceptions::ClassCastException::ClassCastException (**
const std::exception * *cause*) throw () `[inline]`

Constructor.

Parameters

<i>cause</i>	Pointer (p.2083) to the exception that caused this one to be thrown, the object is cloned caller retains ownership.
--------------	--

6.117.1.5 **decaf::lang::exceptions::ClassCastException::ClassCastException**
(const char * *file*, const int *lineNumber*, const char * *msg*, ...) throw ()
`[inline]`

Constructor - Initializes the file name and line number where this message occurred.

Sets the message to report, using an optional list of arguments to parse into the message

Parameters

<i>file</i>	The file name where exception occurs
<i>lineNumber</i>	The line number where the exception occurred.
<i>msg</i>	The message to report
<i>...</i>	list of primitives that are formatted into the message

6.117.1.6 **decaf::lang::exceptions::ClassCastException::ClassCastException (**
const char * *file*, const int *lineNumber*, const std::exception * *cause*, const char * *msg*, ...) throw () `[inline]`

Constructor - Initializes the file name and line number where this message occurred.

Sets the message to report, using an optional list of arguments to parse into the message

Parameters

<i>file</i>	The file name where exception occurs
<i>lineNumber</i>	The line number where the exception occurred.
<i>cause</i>	The exception that was the cause for this one to be thrown.
<i>msg</i>	The message to report
...	list of primitives that are formatted into the message

6.117.1.7 `virtual decaf::lang::exceptions::ClassCastException-
::~ClassCastException () throw () [inline,
virtual]`

6.117.2 Member Function Documentation

6.117.2.1 `virtual ClassCastException* decaf::lang::exceptions-
::ClassCastException::clone () const [inline,
virtual]`

Clones this exception.

This is useful for cases where you need to preserve the type of the original exception as well as the message. All subclasses should override.

Returns

an new **Exception** (p. 1279) instance that is a copy of this one.

Reimplemented from **decaf::lang::Exception** (p. 1282).

The documentation for this class was generated from the following file:

- src/main/decaf/lang/exceptions/**ClassCastException.h**

6.118 cms::Closeable Class Reference

Interface for a class that implements the close method.

```
#include <src/main/cms/Closeable.h>
```

Inheritance diagram for cms::Closeable:

Public Member Functions

- virtual **~Closeable** () throw ()
- virtual void **close** ()=0

Closes this object and deallocates the appropriate resources.

6.118.1 Detailed Description

Interface for a class that implements the close method.

A class that implements this interface should release all resources upon the close call and should throw an exception from any methods that require those resources after they have been closed.

Since

1.0

6.118.2 Constructor & Destructor Documentation

6.118.2.1 `virtual cms::Closeable::~~Closeable () throw ()` [virtual]

6.118.3 Member Function Documentation

6.118.3.1 `virtual void cms::Closeable::close ()` [pure virtual]

Closes this object and deallocates the appropriate resources.

The object is generally no longer usable after calling close.

Exceptions

<i>CMSException</i> (p. 826)	- If an error occurs while the resource is being closed.
--	--

Implemented in `activemq::core::ActiveMQConnection` (p. 195), `activemq::core::ActiveMQSession` (p. 338), `cms::Session` (p. 2365), `activemq::core::ActiveMQProducer` (p. 310), `activemq::core::ActiveMQConsumer` (p. 237), `activemq::cmsutil::PooledSession` (p. 2095), `activemq::core::ActiveMQQueueBrowser` (p. 327), `cms::Connection` (p. 935), `activemq::commands::ActiveMQTempDestination` (p. 380), `activemq::cmsutil::CachedConsumer` (p. 736), and `activemq::cmsutil::CachedProducer` (p. 740).

The documentation for this class was generated from the following file:

- `src/main/cms/Closeable.h`

6.119 decaf::io::Closeable Class Reference

Interface for a class that implements the close method.

```
#include <src/main/decaf/io/Closeable.h>
```

Inheritance diagram for `decaf::io::Closeable`:

Public Member Functions

- virtual `~Closeable()`
- virtual void `close()`=0

Closes this object and deallocates the appropriate resources.

6.119.1 Detailed Description

Interface for a class that implements the close method.

6.119.2 Constructor & Destructor Documentation

6.119.2.1 virtual `decaf::io::Closeable::~~Closeable()` [inline, virtual]

6.119.3 Member Function Documentation

6.119.3.1 virtual void `decaf::io::Closeable::close()` [pure virtual]

Closes this object and deallocates the appropriate resources.

The object is generally no longer usable after calling close.

Exceptions

<i>IOException</i> (p. 1545)	if an error occurs while closing.
--	-----------------------------------

Implemented in `decaf::net::Socket` (p. 2458), `activemq::transport::IOTransport` (p. 1551), `activemq::transport::mock::MockTransport` (p. 1951), `activemq::wireformat::openwire::OpenWireFormatNegotiator` (p. 2062), `activemq::transport::TransportFilter` (p. 2802), `activemq::transport::failover::FailoverTransport` (p. 1308), `decaf::util::zip::InflaterInputStream` (p. 1461), `decaf::util::zip::DeflaterOutputStream` (p. 1192), `activemq::transport::tcp::TcpTransport` (p. 2694), `decaf::util::logging::StreamHandler` (p. 2604), `decaf::io::FilterOutputStream` (p. 1342), `activemq::transport::correlator::ResponseCorrelator` (p. 2304), `decaf::internal::net::ssl::openssl::OpenSSLSocket` (p. 2020), `decaf::io::BufferedInputStream` (p. 592), `decaf::io::BlockingByteArrayInputStream` (p. 537), `decaf::io::FilterInputStream` (p. 1337), `decaf::internal::net::tcp::TcpSocketInputStream` (p. 2690), `activemq::transport::inactivity::InactivityMonitor` (p. 1427), `decaf::io::InputStreamReader` (p. 1476), `decaf::io::OutputStreamWriter` (p. 2075), `decaf::util::logging::ConsoleHandler` (p. 991), `decaf::internal::net::ssl::openssl::OpenSSLSocketInputStream` (p. 2042), `decaf::io::InputStream` (p. 1466), `decaf::io::OutputStream` (p. 2068), `decaf::internal::net::tcp::TcpSocketOutputStream` (p. 2692), `decaf::internal::net::ssl::openssl::OpenSSLSocketOutputStream` (p. 2044), `decaf::internal::io::StandardErrorOutputStream` (p. 2529), and `decaf::internal::io::StandardOutputStream` (p. 2533).

The documentation for this class was generated from the following file:

- src/main/decaf/io/Closeable.h

6.120 activemq::transport::failover::CloseTransportsTask Class Reference

```
#include <src/main/activemq/transport/failover/Close-
TransportsTask.h>
```

Inheritance diagram for activemq::transport::failover::CloseTransportsTask:

Public Member Functions

- **CloseTransportsTask** ()
- virtual **~CloseTransportsTask** ()
- void **add** (const **Pointer**< **Transport** > &transport)
*Add a new **Transport** (p. 2790) to close.*
- virtual bool **isPending** () const
This Task is pending if there are transports in the Queue that need to be closed.
- virtual bool **iterate** ()
Return true until all transports have been closed and removed from the queue.

6.120.1 Constructor & Destructor Documentation

6.120.1.1 **activemq::transport::failover::CloseTransportsTask::CloseTransportsTask** ()

6.120.1.2 virtual **activemq::transport::failover::CloseTransportsTask::~~CloseTransportsTask** () [virtual]

6.120.2 Member Function Documentation

6.120.2.1 void **activemq::transport::failover::CloseTransportsTask::add** (const **Pointer**< **Transport** > & *transport*)

Add a new **Transport** (p. 2790) to close.

6.120.2.2 virtual bool **activemq::transport::failover::CloseTransportsTask::isPending** () const [virtual]

This Task is pending if there are transports in the Queue that need to be closed.

Returns

true if there is a transport in the queue that needs closed.

Implements **activemq::threads::CompositeTask** (p. 893).

6.120.2.3 `virtual bool activemq::transport::failover::CloseTransportsTask::iterate () [virtual]`

Return true until all transports have been closed and removed from the queue.

Implements **activemq::threads::Task** (p. 2676).

The documentation for this class was generated from the following file:

- `src/main/activemq/transport/failover/CloseTransportsTask.h`

6.121 activemq::cmsutil::CmsAccessor Class Reference

Base class for **activemq.cmsutil.CmsTemplate** (p. 836) and other CMS-accessing gateway helpers, defining common properties such as the CMS **cms.ConnectionFactory** (p. 955) to operate on.

```
#include <src/main/activemq/cmsutil/CmsAccessor.h>
```

Inheritance diagram for **activemq::cmsutil::CmsAccessor**:

Public Member Functions

- **CmsAccessor** ()
- virtual **~CmsAccessor** ()
- virtual **ResourceLifecycleManager** * **getResourceLifecycleManager** ()
- virtual const **ResourceLifecycleManager** * **getResourceLifecycleManager** () const
- virtual void **setConnectionFactory** (**cms::ConnectionFactory** *connectionFactory)
- *Set the ConnectionFactory to use for obtaining CMS Connections.*
- virtual const **cms::ConnectionFactory** * **getConnectionFactory** () const
- *Return the ConnectionFactory that this accessor uses for obtaining CMS Connections.*
- virtual **cms::ConnectionFactory** * **getConnectionFactory** ()
- *Return the ConnectionFactory that this accessor uses for obtaining CMS Connections.*
- virtual void **setSessionAcknowledgeMode** (**cms::Session::AcknowledgeMode** sessionAcknowledgeMode)
- *Set the CMS acknowledgment mode that is used when creating a CMS Session to send a message.*

- virtual **cms::Session::AcknowledgeMode** getSessionAcknowledgeMode () const

Return the acknowledgment mode for CMS sessions.

Protected Member Functions

- **CmsAccessor** (const **CmsAccessor** &)
- **CmsAccessor** & **operator=** (const **CmsAccessor** &)
- virtual void **init** ()
Initializes this object and prepares it for use.
- virtual void **destroy** ()
Shuts down this object and destroys any allocated resources.
- virtual **cms::Connection** * **createConnection** ()
Create a CMS Connection via this template's ConnectionFactory.
- virtual **cms::Session** * **createSession** (**cms::Connection** *con)
Create a CMS Session for the given Connection.
- virtual void **checkConnectionFactory** ()
Verifies that the connection factory is valid.

6.121.1 Detailed Description

Base class for **activemq.cmsutil.CmsTemplate** (p. 836) and other CMS-accessing gateway helpers, defining common properties such as the CMS **cms.ConnectionFactory** (p. 955) to operate on.

The subclass **activemq.cmsutil.CmsDestinationAccessor** (p. 823) adds further, destination-related properties.

Not intended to be used directly.

See also

activemq.cmsutil.CmsDestinationAccessor (p. 823)
activemq.cmsutil.CmsTemplate (p. 836)

6.121.2 Constructor & Destructor Documentation

- 6.121.2.1 **activemq::cmsutil::CmsAccessor::CmsAccessor** (const **CmsAccessor** &) [protected]
- 6.121.2.2 **activemq::cmsutil::CmsAccessor::CmsAccessor** ()
- 6.121.2.3 virtual **activemq::cmsutil::CmsAccessor::~~CmsAccessor** () [virtual]

6.121.3 Member Function Documentation

6.121.3.1 `virtual void activemq::cmsutil::CmsAccessor::checkConnectionFactory ()` [protected, virtual]

Verifies that the connection factory is valid.

Exceptions

<i>IllegalStateException</i>	if this object has not been initialized.
------------------------------	--

6.121.3.2 `virtual cms::Connection* activemq::cmsutil::CmsAccessor::createConnection ()` [protected, virtual]

Create a CMS Connection via this template's ConnectionFactory.

Returns

the new CMS Connection

Exceptions

<i>CMSEException</i>	if thrown by CMS API methods
<i>IllegalStateException</i>	if this object has not been initialized.

6.121.3.3 `virtual cms::Session* activemq::cmsutil::CmsAccessor::createSession (cms::Connection * con)` [protected, virtual]

Create a CMS Session for the given Connection.

Parameters

<i>con</i>	The CMS Connection to create a Session for
------------	--

Returns

the new CMS Session

Exceptions

<i>CMSEException</i>	if thrown by CMS API methods
<i>IllegalStateException</i>	if this object has not been initialized.

6.121.3.4 `virtual void activemq::cmsutil::CmsAccessor::destroy () [inline, protected, virtual]`

Shuts down this object and destroys any allocated resources.

Exceptions

<i>CMSException</i>	if an error occurs during destruction.
<i>IllegalStateException</i>	if this object has already been destroyed.

Reimplemented in **activemq::cmsutil::CmsTemplate** (p. 840), and **activemq::cmsutil::CmsDestinationAccessor** (p. 824).

6.121.3.5 `virtual const cms::ConnectionFactory* activemq::cmsutil::CmsAccessor::getConnectionFactory () const [inline, virtual]`

Return the ConnectionFactory that this accessor uses for obtaining CMS Connections.

6.121.3.6 `virtual cms::ConnectionFactory* activemq::cmsutil::CmsAccessor::getConnectionFactory () [inline, virtual]`

Return the ConnectionFactory that this accessor uses for obtaining CMS Connections.

6.121.3.7 `virtual ResourceLifecycleManager* activemq::cmsutil::CmsAccessor::getResourceLifecycleManager () [inline, virtual]`

6.121.3.8 `virtual const ResourceLifecycleManager* activemq::cmsutil::CmsAccessor::getResourceLifecycleManager () const [inline, virtual]`

6.121.3.9 `virtual cms::Session::AcknowledgeMode activemq::cmsutil::CmsAccessor::getSessionAcknowledgeMode () const [inline, virtual]`

Return the acknowledgment mode for CMS sessions.

Returns

the acknowledgment mode applied by this accessor

6.121.3.10 `virtual void activemq::cmsutil::CmsAccessor::init () [protected, virtual]`

Initializes this object and prepares it for use.

This should be called before any other methods are called. This version does nothing.

Exceptions

<i>CMSException</i>	if an error occurs during initialization.
<i>IllegalStateException</i>	if this object has already been initialized.

Reimplemented in **activemq::cmsutil::CmsTemplate** (p.843), and **activemq::cmsutil::CmsDestinationAccessor** (p.825).

6.121.3.11 **CmsAccessor& activemq::cmsutil::CmsAccessor::operator= (const CmsAccessor &)** [protected]

6.121.3.12 **virtual void activemq::cmsutil::CmsAccessor::setConnectionFactory (cms::ConnectionFactory * *connectionFactory*)** [inline, virtual]

Set the ConnectionFactory to use for obtaining CMS Connections.

6.121.3.13 **virtual void activemq::cmsutil::CmsAccessor::setSessionAcknowledgeMode (cms::Session::AcknowledgeMode *sessionAcknowledgeMode*)** [inline, virtual]

Set the CMS acknowledgment mode that is used when creating a CMS Session to send a message.

Default is AUTO_ACKNOWLEDGE.

Parameters

<i>session-AcknowledgeMode</i>	The acknowledgment mode to assign to the Session.
--------------------------------	---

The documentation for this class was generated from the following file:

- src/main/activemq/cmsutil/**CmsAccessor.h**

6.122 activemq::cmsutil::CmsDestinationAccessor Class Reference

Extends the **CmsAccessor** (p.819) to add support for resolving destination names.

```
#include <src/main/activemq/cmsutil/CmsDestinationAccessor.h>
```

Inheritance diagram for activemq::cmsutil::CmsDestinationAccessor:

Public Member Functions

- **CmsDestinationAccessor** ()
- virtual **~CmsDestinationAccessor** ()
- virtual bool **isPubSubDomain** () const
- virtual void **setPubSubDomain** (bool pubSubDomain)
- virtual **DestinationResolver** * **getDestinationResolver** ()
- virtual const **DestinationResolver** * **getDestinationResolver** () const
- virtual void **setDestinationResolver** (**DestinationResolver** *destRes)

Protected Member Functions

- virtual void **init** ()
Initializes this object and prepares it for use.
- virtual void **destroy** ()
Shuts down this object and destroys any allocated resources.
- virtual **cms::Destination** * **resolveDestinationName** (**cms::Session** *session, const std::string &destName)
- virtual void **checkDestinationResolver** ()

6.122.1 Detailed Description

Extends the **CmsAccessor** (p. 819) to add support for resolving destination names.

Not intended to be used directly.

See also

CmsTemplate (p. 836)
CmsAccessor (p. 819)

6.122.2 Constructor & Destructor Documentation

6.122.2.1 **activemq::cmsutil::CmsDestinationAccessor::CmsDestinationAccessor** ()

6.122.2.2 **virtual activemq::cmsutil::CmsDestinationAccessor::~~CmsDestinationAccessor** () [virtual]

6.122.3 Member Function Documentation

6.122.3.1 **virtual void activemq::cmsutil::CmsDestinationAccessor::checkDestinationResolver** () [protected, virtual]

6.122.3.2 `virtual void activemq::cmsutil::CmsDestinationAccessor::destroy ()`
`[protected, virtual]`

Shuts down this object and destroys any allocated resources.

Exceptions

<i>CMSException</i>	if an error occurs during destruction.
<i>IllegalStateException</i>	if this object has already been destroyed.

Reimplemented from **activemq::cmsutil::CmsAccessor** (p. 821).

Reimplemented in **activemq::cmsutil::CmsTemplate** (p. 840).

6.122.3.3 `virtual DestinationResolver* activemq::cmsutil::CmsDestinationAccessor::getDestinationResolver ()` `[inline, virtual]`

6.122.3.4 `virtual const DestinationResolver* activemq::cmsutil::CmsDestinationAccessor::getDestinationResolver () const` `[inline, virtual]`

6.122.3.5 `virtual void activemq::cmsutil::CmsDestinationAccessor::init ()`
`[protected, virtual]`

Initializes this object and prepares it for use.

This should be called before any other methods are called. This version does nothing.

Exceptions

<i>CMSException</i>	if an error occurs during initialization.
<i>IllegalStateException</i>	if this object has already been initialized.

Reimplemented from **activemq::cmsutil::CmsAccessor** (p. 822).

Reimplemented in **activemq::cmsutil::CmsTemplate** (p. 843).

6.122.3.6 `virtual bool activemq::cmsutil::CmsDestinationAccessor::isPubSubDomain () const` `[inline, virtual]`

6.122.3.7 `virtual cms::Destination* activemq::cmsutil::CmsDestinationAccessor::resolveDestinationName (cms::Session * session, const std::string & destName)` `[protected, virtual]`

6.122.3.8 `virtual void activemq::cmsutil::CmsDestinationAccessor::setDestinationResolver (DestinationResolver * destRes)` `[inline, virtual]`

6.122.3.9 `virtual void activemq::cmsutil::CmsDestinationAccessor-
::setPubSubDomain (bool pubSubDomain) [inline,
virtual]`

Reimplemented in `activemq::cmsutil::CmsTemplate` (p. 849).

Referenced by `activemq::cmsutil::CmsTemplate::setPubSubDomain()`.

The documentation for this class was generated from the following file:

- `src/main/activemq/cmsutil/CmsDestinationAccessor.h`

6.123 cms::CMSException Class Reference

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

```
#include <src/main/cms/CMSException.h>
```

Inheritance diagram for `cms::CMSException`:

Public Member Functions

- `CMSException ()`
- `CMSException (const CMSException &ex)`
- `CMSException (const std::string &message)`
- `CMSException (const std::string &message, const std::exception *cause)`
- `CMSException (const std::string &message, const std::exception *cause, const std::vector< std::pair< std::string, int > > &stackTrace)`
- `virtual ~CMSException () throw ()`
- `virtual std::string getMessage () const`
Gets the cause of the error.
- `virtual const std::exception * getCause () const`
Gets the exception that caused this one to be thrown, this allows for chaining of exceptions in the case of a method that throws only a particular exception but wishes to allow for the real causal exception to be passed only in case the caller knows about that type of exception and wishes to respond to it.
- `virtual std::vector< std::pair < std::string, int > > getStackTrace () const`
Provides the stack trace for every point where this exception was caught, marked, and rethrown.
- `virtual void setMark (const char *file, const int lineNumber)`
Adds a file/line number to the stack trace.
- `virtual void printStackTrace () const`
Prints the stack trace to std::err.
- `virtual void printStackTrace (std::ostream &stream) const`
Prints the stack trace to the given output stream.

- virtual std::string **getStackTraceString** () const
Gets the stack trace as one contiguous string.
- virtual const char * **what** () const throw ()
*Overloads the std::exception **what()** (p. 829) function to return the cause of the exception.*

6.123.1 Detailed Description

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

This class represents an error that has occurred in CMS, providers can wrap provider specific exceptions in this class by setting the cause to an instance of a provider specific exception provided it can be cast to an std::exception.

Since the contained cause exception is of type std::exception and the C++ exception class has no clone or copy method defined the contained exception can only be owned by one instance of an **CMSException** (p. 826). To that end the class hands off the exception to each successive copy so care must be taken when handling **CMSException** (p. 826) instances.

Since

1.0

6.123.2 Constructor & Destructor Documentation

- 6.123.2.1 **cms::CMSException::CMSException** ()
- 6.123.2.2 **cms::CMSException::CMSException** (const **CMSException** & *ex*)
- 6.123.2.3 **cms::CMSException::CMSException** (const std::string & *message*)
- 6.123.2.4 **cms::CMSException::CMSException** (const std::string & *message*, const std::exception * *cause*)
- 6.123.2.5 **cms::CMSException::CMSException** (const std::string & *message*, const std::exception * *cause*, const std::vector< std::pair< std::string, int > > & *stackTrace*)
- 6.123.2.6 virtual **cms::CMSException::~~CMSException** () throw () [virtual]

6.123.3 Member Function Documentation

- 6.123.3.1 virtual const std::exception* **cms::CMSException::getCause** () const [virtual]

Gets the exception that caused this one to be thrown, this allows for chaining of exceptions in the case of a method that throws only a particular exception but wishes to allow

for the real causal exception to be passed only in case the caller knows about that type of exception and wishes to respond to it.

Returns

a const pointer reference to the causal exception, if there was no cause associated with this exception then NULL is returned.

6.123.3.2 `virtual std::string cms::CMSException::getMessage () const` [virtual]

Gets the cause of the error.

Returns

string errors message

6.123.3.3 `virtual std::vector< std::pair< std::string, int> > cms::CMSException::getStackTrace () const` [virtual]

Provides the stack trace for every point where this exception was caught, marked, and rethrown.

Returns

vector containing stack trace strings

6.123.3.4 `virtual std::string cms::CMSException::getStackTraceString () const` [virtual]

Gets the stack trace as one contiguous string.

Returns

string with formatted stack trace data

6.123.3.5 `virtual void cms::CMSException::printStackTrace () const` [virtual]

Prints the stack trace to std::err.

6.123.3.6 `virtual void cms::CMSException::printStackTrace (std::ostream & stream) const` [virtual]

Prints the stack trace to the given output stream.

Parameters

<i>stream</i>	the target output stream.
---------------	---------------------------

6.123.3.7 virtual void **cms::CMSException::setMark** (const char * *file*, const int *lineNumber*) [virtual]

Adds a file/line number to the stack trace.

Parameters

<i>file</i>	The name of the file calling this method (use <code>__FILE__</code>).
<i>lineNumber</i>	The line number in the calling file (use <code>__LINE__</code>).

6.123.3.8 virtual const char* **cms::CMSException::what** () const throw () [virtual]

Overloads the std::exception **what()** (p. 829) function to return the cause of the exception.

Returns

const char pointer to error message

The documentation for this class was generated from the following file:

- src/main/cms/**CMSException.h**

6.124 activemq::util::CMSExceptionSupport Class Reference

```
#include <src/main/activemq/util/CMSExceptionSupport.h>
```

Public Member Functions

- virtual **~CMSExceptionSupport** ()

Static Public Member Functions

- static **cms::CMSException create** (const std::string &msg, const **decaf::lang::Exception** &cause)
- static **cms::CMSException create** (const **decaf::lang::Exception** &cause)
- static **cms::MessageEOFException createMessageEOFException** (const **decaf::lang::Exception** &cause)
- static **cms::MessageFormatException createMessageFormatException** (const **decaf::lang::Exception** &cause)

6.124.1 Constructor & Destructor Documentation

6.124.1.1 `virtual activemq::util::CMSExceptionSupport::~~CMSExceptionSupport () [virtual]`

6.124.2 Member Function Documentation

6.124.2.1 `static cms::CMSException activemq::util::CMSExceptionSupport::create (const std::string & msg, const decaf::lang::Exception & cause) [static]`

6.124.2.2 `static cms::CMSException activemq::util::CMSExceptionSupport::create (const decaf::lang::Exception & cause) [static]`

6.124.2.3 `static cms::MessageEOFException activemq::util::CMSExceptionSupport::createMessageEOFException (const decaf::lang::Exception & cause) [static]`

6.124.2.4 `static cms::MessageFormatException activemq::util::CMSExceptionSupport::createMessageFormatException (const decaf::lang::Exception & cause) [static]`

Referenced by `activemq::commands::ActiveMQMessageTemplate< cms::ObjectMessage >::getBooleanProperty()`, `activemq::commands::ActiveMQMessageTemplate< cms::ObjectMessage >::getByteProperty()`, `activemq::commands::ActiveMQMessageTemplate< cms::ObjectMessage >::getDoubleProperty()`, `activemq::commands::ActiveMQMessageTemplate< cms::ObjectMessage >::getFloatProperty()`, `activemq::commands::ActiveMQMessageTemplate< cms::ObjectMessage >::getIntProperty()`, `activemq::commands::ActiveMQMessageTemplate< cms::ObjectMessage >::getLongProperty()`, `activemq::commands::ActiveMQMessageTemplate< cms::ObjectMessage >::getShortProperty()`, and `activemq::commands::ActiveMQMessageTemplate< cms::ObjectMessage >::getStringProperty()`.

The documentation for this class was generated from the following file:

- `src/main/activemq/util/CMSExceptionSupport.h`

6.125 cms::CMSProperties Class Reference

Interface for a Java-like properties object.

```
#include <src/main/cms/CMSProperties.h>
```

Inheritance diagram for `cms::CMSProperties`:

Public Member Functions

- virtual `~CMSProperties ()` throw `()`
- virtual `int size ()` const =0
Returns the current count of all the Properties that are currently stored in the - Properties object.
- virtual `bool isEmpty ()` const =0
Returns true if the properties object is empty.
- virtual `const char * getProperty (const std::string &name)` const =0
Looks up the value for the given property.
- virtual `std::string getProperty (const std::string &name, const std::string &defaultValue)` const =0
Looks up the value for the given property.
- virtual `void setProperty (const std::string &name, const std::string &value)`=0
Sets the value for a given property.
- virtual `bool hasProperty (const std::string &name)` const =0
Check to see if the Property exists in the set.
- virtual `std::string remove (const std::string &name)`=0
Removes the property with the given name.
- virtual `std::vector< std::string > propertyNames ()` const =0
Returns a vector containing all the names of the properties currently stored in the Properties object.
- virtual `std::vector< std::pair < std::string, std::string > > toArray ()` const =0
Method that serializes the contents of the property map to an array.
- virtual `void copy (const CMSProperties *source)`=0
Copies the contents of the given properties object to this one.
- virtual `CMSProperties * clone ()` const =0
Clones this object.
- virtual `void clear ()`=0
Clears all properties from the map.
- virtual `std::string toString ()` const =0
Formats the contents of the Properties Object into a string that can be logged, etc.

6.125.1 Detailed Description

Interface for a Java-like properties object.

This is essentially a map of key-value string pairs.

Since

1.1

6.125.2 Constructor & Destructor Documentation

6.125.2.1 `virtual cms::CMSProperties::~~CMSProperties () throw () [virtual]`

6.125.3 Member Function Documentation

6.125.3.1 `virtual void cms::CMSProperties::clear () [pure virtual]`

Clears all properties from the map.

Implemented in **activemq::util::ActiveMQProperties** (p. 317).

6.125.3.2 `virtual CMSProperties* cms::CMSProperties::clone () const [pure virtual]`

Clones this object.

Returns

a replica of this object.

Implemented in **activemq::util::ActiveMQProperties** (p. 317).

6.125.3.3 `virtual void cms::CMSProperties::copy (const CMSProperties * source) [pure virtual]`

Copies the contents of the given properties object to this one.

Parameters

<i>source</i>	The source properties object.
---------------	-------------------------------

6.125.3.4 `virtual const char* cms::CMSProperties::getProperty (const std::string & name) const [pure virtual]`

Looks up the value for the given property.

Parameters

<i>name</i>	The name of the property to be looked up.
-------------	---

Returns

the value of the property with the given name, if it exists. If it does not exist, returns NULL.

Implemented in **activemq::util::ActiveMQProperties** (p. 318).

6.125.3.5 `virtual std::string cms::CMSProperties::getProperty (const std::string & name, const std::string & defaultValue) const` [pure virtual]

Looks up the value for the given property.

Parameters

<i>name</i>	the name of the property to be looked up.
<i>defaultValue</i>	The value to be returned if the given property does not exist.

Returns

The value of the property specified by *name*, if it exists, otherwise the *defaultValue*.

Implemented in **activemq::util::ActiveMQProperties** (p.318).

6.125.3.6 `virtual bool cms::CMSProperties::hasProperty (const std::string & name) const` [pure virtual]

Check to see if the Property exists in the set.

Parameters

<i>name</i>	the name of the property to check
-------------	-----------------------------------

Returns

true if property exists, false otherwise.

Implemented in **activemq::util::ActiveMQProperties** (p.318).

6.125.3.7 `virtual bool cms::CMSProperties::isEmpty () const` [pure virtual]

Returns true if the properties object is empty.

Returns

true if empty

Implemented in **activemq::util::ActiveMQProperties** (p.319).

6.125.3.8 `virtual std::vector<std::string> cms::CMSProperties::propertyNames () const` [pure virtual]

Returns a vector containing all the names of the properties currently stored in the - Properties object.

Returns

an STL `std::vector<std::string>` with all the currently stored property names.

Implemented in **activemq::util::ActiveMQProperties** (p. 319).

6.125.3.9 `virtual std::string cms::CMSProperties::remove (const std::string & name)`
`[pure virtual]`

Removes the property with the given name.

If the property existed in the collection then it is removed and returned, otherwise an empty string is returned.

Parameters

<i>name</i>	the name of the property to be removed.
-------------	---

Returns

the value that was removed from the Properties, or empty string.

Implemented in **activemq::util::ActiveMQProperties** (p. 319).

6.125.3.10 `virtual void cms::CMSProperties::setProperty (const std::string & name,`
`const std::string & value) [pure virtual]`

Sets the value for a given property.

If the property already exists, overwrites the value.

Parameters

<i>name</i>	The name of the value to be written.
<i>value</i>	The value to be written.

Implemented in **activemq::util::ActiveMQProperties** (p. 320).

6.125.3.11 `virtual int cms::CMSProperties::size () const` `[pure virtual]`

Returns the current count of all the Properties that are currently stored in the Properties object.

Returns

the number of properties currently stored.

Implemented in **activemq::util::ActiveMQProperties** (p. 320).

6.125.3.12 `virtual std::vector< std::pair< std::string, std::string > >
 cms::CMSProperties::toArray() const [pure virtual]`

Method that serializes the contents of the property map to an array.

Returns

list of pairs where the first is the name and the second is the value.

Implemented in **activemq::util::ActiveMQProperties** (p. 320).

6.125.3.13 `virtual std::string cms::CMSProperties::toString() const [pure
 virtual]`

Formats the contents of the Properties Object into a string that can be logged, etc.

Returns

string value of this object.

Implemented in **activemq::util::ActiveMQProperties** (p. 321).

The documentation for this class was generated from the following file:

- src/main/cms/**CMSProperties.h**

6.126 cms::CMSSecurityException Class Reference

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

```
#include <src/main/cms/CMSSecurityException.h>
```

Inheritance diagram for cms::CMSSecurityException:

Public Member Functions

- **CMSSecurityException** ()
- **CMSSecurityException** (const **CMSSecurityException** &ex)
- **CMSSecurityException** (const std::string &message)
- **CMSSecurityException** (const std::string &message, const std::exception *cause)
- **CMSSecurityException** (const std::string &message, const std::exception *cause, const std::vector< std::pair< std::string, int > > &stackTrace)
- virtual ~**CMSSecurityException** () throw ()

6.126.1 Detailed Description

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

It may also be thrown for any case where a security restriction prevents a method from completing.

Since

1.3

6.126.2 Constructor & Destructor Documentation

6.126.2.1 `cms::CMSSecurityException::CMSSecurityException ()`

6.126.2.2 `cms::CMSSecurityException::CMSSecurityException (const CMSSecurityException & ex)`

6.126.2.3 `cms::CMSSecurityException::CMSSecurityException (const std::string & message)`

6.126.2.4 `cms::CMSSecurityException::CMSSecurityException (const std::string & message, const std::exception * cause)`

6.126.2.5 `cms::CMSSecurityException::CMSSecurityException (const std::string & message, const std::exception * cause, const std::vector< std::pair< std::string, int > > & stackTrace)`

6.126.2.6 `virtual cms::CMSSecurityException::~~CMSSecurityException () throw () [virtual]`

The documentation for this class was generated from the following file:

- `src/main/cms/CMSSecurityException.h`

6.127 activemq::cmsutil::CmsTemplate Class Reference

CmsTemplate (p. 836) simplifies performing synchronous CMS operations.

```
#include <src/main/activemq/cmsutil/CmsTemplate.h>
```

Inheritance diagram for `activemq::cmsutil::CmsTemplate`:

Data Structures

- class **ProducerExecutor**
- class **ReceiveExecutor**
- class **ResolveProducerExecutor**
- class **ResolveReceiveExecutor**
- class **SendExecutor**

Public Member Functions

- **CmsTemplate** ()
- **CmsTemplate** (cms::ConnectionFactory *connectionFactory)
- virtual ~**CmsTemplate** () throw ()
- virtual void **setDefaultDestination** (cms::Destination *defaultDestination)
Sets the destination object to be used by default for send/receive operations.
- virtual const cms::Destination * **getDefaultDestination** () const
Retrieves the default destination to be used for send/receive operations.
- virtual cms::Destination * **getDefaultDestination** ()
Retrieves the default destination to be used for send/receive operations.
- virtual void **setDefaultDestinationName** (const std::string &defaultDestinationName)
Sets the name of the default destination to be used from send/receive operations.
- virtual const std::string **getDefaultDestinationName** () const
Gets the name of the default destination to be used for send/receive operations.
- virtual void **setPubSubDomain** (bool pubSubDomain)
Indicates whether the default destination is a topic (true) or a queue (false).
- virtual void **setMessageIdEnabled** (bool messageIdEnabled)
- virtual bool **isMessageIdEnabled** () const
- virtual void **setMessageTimestampEnabled** (bool messageTimestampEnabled)
- virtual bool **isMessageTimestampEnabled** () const
- virtual void **setNoLocal** (bool noLocal)
- virtual bool **isNoLocal** () const
- virtual void **setReceiveTimeout** (long long receiveTimeout)
- virtual long long **getReceiveTimeout** () const
- virtual void **setExplicitQosEnabled** (bool explicitQosEnabled)
Set if the QOS values (deliveryMode, priority, timeToLive) should be used for sending a message.
- virtual bool **isExplicitQosEnabled** () const
If "true", then the values of deliveryMode, priority, and timeToLive will be used when sending a message.
- virtual void **setDeliveryPersistent** (bool deliveryPersistent)
Set whether message delivery should be persistent or non-persistent, specified as boolean value ("true" or "false").
- virtual void **setDeliveryMode** (int deliveryMode)
Set the delivery mode to use when sending a message.

- virtual int **getDeliveryMode** () const
Return the delivery mode to use when sending a message.
- virtual void **setPriority** (int priority)
Set the priority of a message when sending.
- virtual int **getPriority** () const
Return the priority of a message when sending.
- virtual void **setTimeToLive** (long long timeToLive)
Set the time-to-live of the message when sending.
- virtual long long **getTimeToLive** () const
Return the time-to-live of the message when sending.
- virtual void **execute** (**SessionCallback** *action)
Executes the given action within a CMS Session.
- virtual void **execute** (**ProducerCallback** *action)
Executes the given action and provides it with a CMS Session and producer.
- virtual void **execute** (**cms::Destination** *dest, **ProducerCallback** *action)
Executes the given action and provides it with a CMS Session and producer.
- virtual void **execute** (const std::string &destinationName, **ProducerCallback** *action)
Executes the given action and provides it with a CMS Session and producer.
- virtual void **send** (**MessageCreator** *messageCreator)
Convenience method for sending a message to the default destination.
- virtual void **send** (**cms::Destination** *dest, **MessageCreator** *messageCreator)
Convenience method for sending a message to the specified destination.
- virtual void **send** (const std::string &destinationName, **MessageCreator** *messageCreator)
Convenience method for sending a message to the specified destination.
- virtual **cms::Message** * **receive** ()
Performs a synchronous read from the default destination.
- virtual **cms::Message** * **receive** (**cms::Destination** *destination)
Performs a synchronous read from the specified destination.
- virtual **cms::Message** * **receive** (const std::string &destinationName)
Performs a synchronous read from the specified destination.
- virtual **cms::Message** * **receiveSelected** (const std::string &selector)
Performs a synchronous read consuming only messages identified by the given selector.
- virtual **cms::Message** * **receiveSelected** (**cms::Destination** *destination, const std::string &selector)
Performs a synchronous read from the specified destination, consuming only messages identified by the given selector.
- virtual **cms::Message** * **receiveSelected** (const std::string &destinationName, const std::string &selector)
Performs a synchronous read from the specified destination, consuming only messages identified by the given selector.

Static Public Attributes

- static const long long **RECEIVE_TIMEOUT_NO_WAIT**
Timeout value indicating that a receive operation should check if a message is immediately available without blocking.
- static const long long **RECEIVE_TIMEOUT_INDEFINITE_WAIT**
Timeout value indicating a blocking receive without timeout.
- static const int **DEFAULT_PRIORITY**
Default message priority.
- static const long long **DEFAULT_TIME_TO_LIVE**
My default, messages should live forever.

Protected Member Functions

- void **init** ()
Initializes this object and prepares it for use.
- void **destroy** ()
Shuts down this object and destroys any allocated resources.

Friends

- class **ProducerExecutor**
- class **ResolveProducerExecutor**
- class **SendExecutor**
- class **ReceiveExecutor**
- class **ResolveReceiveExecutor**

6.127.1 Detailed Description

CmsTemplate (p. 836) simplifies performing synchronous CMS operations.

This class is intended to be for CMS what Spring's `JmsTemplate` is for JMS. Provided with a `CMS ConnectionFactory`, creates and manages all other CMS resources internally.

Before using **CmsTemplate** (p. 836) the user must first set the destination (either by name or by setting the destination object directly) and then call `init` to initialize the object for use.

CmsTemplate (p. 836) allows the user to get access to a `CMS Session` through a user-defined **SessionCallback** (p. 2377). Similarly, if the user wants direct access to a `CMS MessageProducer`, it can provide a **ProducerCallback** (p. 2179). As a convenience, the user can bypass having to provide callbacks altogether for sending messages, by calling one of the `send` methods.

See also

SessionCallback (p. 2377)
ProducerCallback (p. 2179)
MessageCreator (p. 1880)

6.127.2 Constructor & Destructor Documentation

6.127.2.1 `activemq::cmsutil::CmsTemplate::CmsTemplate ()`

6.127.2.2 `activemq::cmsutil::CmsTemplate::CmsTemplate (cms::ConnectionFactory * connectionFactory)`

6.127.2.3 `virtual activemq::cmsutil::CmsTemplate::~~CmsTemplate () throw ()`
`[virtual]`

6.127.3 Member Function Documentation

6.127.3.1 `void activemq::cmsutil::CmsTemplate::destroy ()` `[protected, virtual]`

Shuts down this object and destroys any allocated resources.

Exceptions

<i>CMSException</i>	if an error occurs during destruction.
<i>IllegalStateException</i>	if this object has already been destroyed.

Reimplemented from **activemq::cmsutil::CmsDestinationAccessor** (p. 824).

6.127.3.2 `virtual void activemq::cmsutil::CmsTemplate::execute (SessionCallback * action)` `[virtual]`

Executes the given action within a CMS Session.

Parameters

<i>action</i>	the action to perform within a CMS Session
---------------	--

Exceptions

<i>cms::CMS-Exception</i> (p. 826)	thrown if an error occurs.
--	----------------------------

6.127.3.3 virtual void **activemq::cmsutil::CmsTemplate::execute** (**ProducerCallback** * *action*) [virtual]

Executes the given action and provides it with a CMS Session and producer.

Parameters

<i>action</i>	the action to perform
---------------	-----------------------

Exceptions

<i>cms::CMS-Exception</i> (p. 826)	thrown if an error occurs.
--	----------------------------

6.127.3.4 virtual void **activemq::cmsutil::CmsTemplate::execute** (**cms::Destination** * *dest*, **ProducerCallback** * *action*) [virtual]

Executes the given action and provides it with a CMS Session and producer.

Parameters

<i>dest</i>	the destination to send messages to
<i>action</i>	the action to perform

Exceptions

<i>cms::CMS-Exception</i> (p. 826)	thrown if an error occurs.
--	----------------------------

6.127.3.5 virtual void **activemq::cmsutil::CmsTemplate::execute** (const std::string & *destinationName*, **ProducerCallback** * *action*) [virtual]

Executes the given action and provides it with a CMS Session and producer.

Parameters

<i>destination-Name</i>	the name of the destination to send messages to (to internally be resolved to an actual destination)
<i>action</i>	the action to perform

Exceptions

<i>cms::CMS-Exception</i> (p. 826)	thrown if an error occurs.
--	----------------------------

6.127.3.6 `virtual const cms::Destination* activemq::cmsutil::CmsTemplate::getDefaultDestination () const [inline, virtual]`

Retrieves the default destination to be used for send/receive operations.

Returns

the default destination. Const version of this method.

6.127.3.7 `virtual cms::Destination* activemq::cmsutil::CmsTemplate::getDefaultDestination () [inline, virtual]`

Retrieves the default destination to be used for send/receive operations.

Returns

the default destination. Non-const version of this method.

6.127.3.8 `virtual const std::string activemq::cmsutil::CmsTemplate::getDefaultDestinationName () const [inline, virtual]`

Gets the name of the default destination to be used for send/receive operations.

The destination type (topic/queue) is determined by the `pubSubDomain` property.

Returns

the default name of the destination for send/receive operations.

6.127.3.9 `virtual int activemq::cmsutil::CmsTemplate::getDeliveryMode () const [inline, virtual]`

Return the delivery mode to use when sending a message.

6.127.3.10 `virtual int activemq::cmsutil::CmsTemplate::getPriority () const [inline, virtual]`

Return the priority of a message when sending.

6.127.3.11 `virtual long long activemq::cmsutil::CmsTemplate::getReceiveTimeout () const [inline, virtual]`

6.127.3.12 `virtual long long activemq::cmsutil::CmsTemplate::getTimeToLive () const [inline, virtual]`

Return the time-to-live of the message when sending.

6.127.3.13 `void activemq::cmsutil::CmsTemplate::init () [protected, virtual]`

Initializes this object and prepares it for use.

This should be called before any other methods are called. This version does nothing.

Exceptions

<i>CMSException</i>	if an error occurs during initialization.
<i>IllegalStateException</i>	if this object has already been initialized.

Reimplemented from **activemq::cmsutil::CmsDestinationAccessor** (p. 825).

6.127.3.14 `virtual bool activemq::cmsutil::CmsTemplate::isExplicitQosEnabled () const [inline, virtual]`

If "true", then the values of deliveryMode, priority, and timeToLive will be used when sending a message.

Otherwise, the default values, that may be set administratively, will be used.

Returns

true if overriding default values of QOS parameters (deliveryMode, priority, and timeToLive)

See also

setDeliveryMode (p. 848)

setPriority (p. 849)

setTimeToLive (p. 850)

6.127.3.15 `virtual bool activemq::cmsutil::CmsTemplate::isMessageIdEnabled () const [inline, virtual]`

6.127.3.16 `virtual bool activemq::cmsutil::CmsTemplate::is-MessageTimestampEnabled () const [inline, virtual]`

6.127.3.17 `virtual bool activemq::cmsutil::CmsTemplate::isNoLocal () const`
`[inline, virtual]`

6.127.3.18 `virtual cms::Message* activemq::cmsutil::CmsTemplate::receive ()`
`[virtual]`

Performs a synchronous read from the default destination.

Returns

the message

Exceptions

<i>cms::CMS-Exception</i> (p. 826)	thrown if an error occurs
--	---------------------------

6.127.3.19 `virtual cms::Message* activemq::cmsutil::CmsTemplate::receive (cms::Destination * destination)` `[virtual]`

Performs a synchronous read from the specified destination.

Parameters

<i>destination</i>	the destination to receive on
--------------------	-------------------------------

Returns

the message

Exceptions

<i>cms::CMS-Exception</i> (p. 826)	thrown if an error occurs
--	---------------------------

6.127.3.20 `virtual cms::Message* activemq::cmsutil::CmsTemplate::receive (const std::string & destinationName)` `[virtual]`

Performs a synchronous read from the specified destination.

Parameters

<i>destination-Name</i>	the name of the destination to receive on (will be resolved to destination internally).
-------------------------	---

Returns

the message

Exceptions

<i>cms::CMS-Exception</i> (p. 826)	thrown if an error occurs
--	---------------------------

6.127.3.21 `virtual cms::Message* activemq::cmsutil::CmsTemplate::receiveSelected (const std::string & selector)`
[virtual]

Performs a synchronous read consuming only messages identified by the given selector.

Parameters

<i>selector</i>	the selector expression.
-----------------	--------------------------

Returns

the message

Exceptions

<i>cms::CMS-Exception</i> (p. 826)	thrown if an error occurs
--	---------------------------

6.127.3.22 `virtual cms::Message* activemq::cmsutil::CmsTemplate::receiveSelected (cms::Destination * destination, const std::string & selector)`
[virtual]

Performs a synchronous read from the specified destination, consuming only messages identified by the given selector.

Parameters

<i>destination</i>	the destination to receive on.
<i>selector</i>	the selector expression.

Returns

the message

Exceptions

<i>cms::CMS-Exception</i> (p. 826)	thrown if an error occurs
--	---------------------------

6.127.3.23 **virtual cms::Message* activemq::cmsutil::CmsTemplate::receive-Selected (const std::string & *destinationName*, const std::string & *selector*)**
[virtual]

Performs a synchronous read from the specified destination, consuming only messages identified by the given selector.

Parameters

<i>destination-Name</i>	the name of the destination to receive on (will be resolved to destination internally).
<i>selector</i>	the selector expression.

Returns

the message

Exceptions

<i>cms::CMS-Exception</i> (p. 826)	thrown if an error occurs
--	---------------------------

6.127.3.24 **virtual void activemq::cmsutil::CmsTemplate::send (MessageCreator * *messageCreator*)** [virtual]

Convenience method for sending a message to the default destination.

Parameters

<i>message-Creator</i>	Responsible for creating the message to be sent
------------------------	---

Exceptions

<i>cms::CMS-Exception</i> (p. 826)	thrown if an error occurs.
--	----------------------------

6.127.3.25 `virtual void activemq::cmsutil::CmsTemplate::send (cms::Destination *
dest, MessageCreator * messageCreator) [virtual]`

Convenience method for sending a message to the specified destination.

Parameters

<i>dest</i>	The destination to send to
<i>message-Creator</i>	Responsible for creating the message to be sent

Exceptions

<i>cms::CMS-Exception</i> (p. 826)	thrown if an error occurs.
--	----------------------------

6.127.3.26 `virtual void activemq::cmsutil::CmsTemplate::send (const std::string &
destinationName, MessageCreator * messageCreator) [virtual]`

Convenience method for sending a message to the specified destination.

Parameters

<i>destination-Name</i>	The name of the destination to send to.
<i>message-Creator</i>	Responsible for creating the message to be sent

Exceptions

<i>cms::CMS-Exception</i> (p. 826)	thrown if an error occurs.
--	----------------------------

6.127.3.27 `virtual void activemq::cmsutil::CmsTemplate::setDefaultDestination (cms::Destination * defaultDestination) [inline, virtual]`

Sets the destination object to be used by default for send/receive operations.

If no default destination is provided, the `defaultDestinationName` property is used to resolve this default destination for send/receive operations.

Parameters

<i>default-Destination</i>	the default destination
----------------------------	-------------------------

6.127.3.28 `virtual void activemq::cmsutil::CmsTemplate::setDefaultDestinationName (const std::string & defaultDestinationName) [inline, virtual]`

Sets the name of the default destination to be used from send/receive operations.

Calling this method will set the `defaultDestination` property to NULL. The destination type (topic/queue) is determined by the `pubSubDomain` property.

Parameters

<i>default-Destination-Name</i>	the name of the destination for send/receive to by default.
---------------------------------	---

6.127.3.29 `virtual void activemq::cmsutil::CmsTemplate::setDeliveryMode (int deliveryMode) [inline, virtual]`

Set the delivery mode to use when sending a message.

Default is the Message default: "PERSISTENT".

Since a default value may be defined administratively, this is only used when "isExplicitQosEnabled" equals "true".

Parameters

<i>delivery-Mode</i>	the delivery mode to use
----------------------	--------------------------

See also

isExplicitQosEnabled (p. 843)

6.127.3.30 `virtual void activemq::cmsutil::CmsTemplate::setDeliveryPersistent (bool deliveryPersistent) [inline, virtual]`

Set whether message delivery should be persistent or non-persistent, specified as boolean value ("true" or "false").

This will set the delivery mode accordingly, to either "PERSISTENT" or "NON_PERSISTENT".

Default it "true" aka delivery mode "PERSISTENT".

See also

setDeliveryMode(int) (p. 848)

6.127.3.31 `virtual void activemq::cmsutil::CmsTemplate::setExplicitQosEnabled (bool explicitQosEnabled) [inline, virtual]`

Set if the QOS values (deliveryMode, priority, timeToLive) should be used for sending a message.

See also

setDeliveryMode (p. 848)

setPriority (p. 849)

setTimeToLive (p. 850)

6.127.3.32 `virtual void activemq::cmsutil::CmsTemplate::setMessageIdEnabled (bool messageIdEnabled) [inline, virtual]`

6.127.3.33 `virtual void activemq::cmsutil::CmsTemplate::setMessageTimestampEnabled (bool messageTimestampEnabled) [inline, virtual]`

6.127.3.34 `virtual void activemq::cmsutil::CmsTemplate::setNoLocal (bool noLocal) [inline, virtual]`

6.127.3.35 `virtual void activemq::cmsutil::CmsTemplate::setPriority (int priority) [inline, virtual]`

Set the priority of a message when sending.

Since a default value may be defined administratively, this is only used when "isExplicitQosEnabled" equals "true".

See also

isExplicitQosEnabled (p. 843)

6.127.3.36 `virtual void activemq::cmsutil::CmsTemplate::setPubSubDomain (bool pubSubDomain) [inline, virtual]`

Indicates whether the default destination is a topic (true) or a queue (false).

Calling this method will set the `defaultDestination` property to NULL.

Parameters

<i>pubSubDomain</i>	indicates whether to use pub-sub messaging (topics).
---------------------	--

Reimplemented from **activemq::cmsutil::CmsDestinationAccessor** (p. 825).

References `activemq::cmsutil::CmsDestinationAccessor::setPubSubDomain()`.

6.127.3.37 `virtual void activemq::cmsutil::CmsTemplate::setReceiveTimeout (long long receiveTimeout)` `[inline, virtual]`

6.127.3.38 `virtual void activemq::cmsutil::CmsTemplate::setTimeToLive (long long timeToLive)` `[inline, virtual]`

Set the time-to-live of the message when sending.

Since a default value may be defined administratively, this is only used when "isExplicit-QosEnabled" equals "true".

Parameters

<i>timeToLive</i>	the message's lifetime (in milliseconds)
-------------------	--

See also

isExplicitQosEnabled (p. 843)

6.127.4 Friends And Related Function Documentation

6.127.4.1 `friend class ProducerExecutor` `[friend]`

6.127.4.2 `friend class ReceiveExecutor` `[friend]`

6.127.4.3 `friend class ResolveProducerExecutor` `[friend]`

6.127.4.4 `friend class ResolveReceiveExecutor` `[friend]`

6.127.4.5 `friend class SendExecutor` `[friend]`

6.127.5 Field Documentation

6.127.5.1 `const int activemq::cmsutil::CmsTemplate::DEFAULT_PRIORITY` `[static]`

Default message priority.

6.127.5.2 `const long long activemq::cmsutil::CmsTemplate::DEFAULT_TIME_TO_LIVE` `[static]`

My default, messages should live forever.

6.127.5.3 `const long long activemq::cmsutil::CmsTemplate::RECEIVE_TIMEOUT_INDEFINITE_WAIT` `[static]`

Timeout value indicating a blocking receive without timeout.

6.127.5.4 `const long long activemq::cmsutil::CmsTemplate::RECEIVE_TIMEOUT_N-O_WAIT` `[static]`

Timeout value indicating that a receive operation should check if a message is immediately available without blocking.

The documentation for this class was generated from the following file:

- `src/main/activemq/cmsutil/CmsTemplate.h`

6.128 code Struct Reference

```
#include <src/main/decaf/internal/util/zip/inftrees.h>
```

Data Fields

- unsigned char **op**
- unsigned char **bits**
- unsigned short **val**

6.128.1 Field Documentation

6.128.1.1 unsigned char `code::bits`

6.128.1.2 unsigned char `code::op`

6.128.1.3 unsigned short `code::val`

The documentation for this struct was generated from the following file:

- `src/main/decaf/internal/util/zip/inftrees.h`

6.129 decaf::util::Collection< E > Class Template Reference

The root interface in the collection hierarchy.

```
#include <src/main/decaf/util/Collection.h>
```

Inheritance diagram for `decaf::util::Collection< E >`:

Public Member Functions

- virtual `~Collection()`

- virtual void **copy** (const **Collection**< E > &collection)=0
*Renders this **Collection** (p. 851) as a Copy of the given **Collection** (p. 851).*
- virtual bool **add** (const E &value)=0
Returns true if this collection changed as a result of the call.
- virtual bool **addAll** (const **Collection**< E > &collection)=0
Adds all of the elements in the specified collection to this collection.
- virtual void **clear** ()=0
Removes all of the elements from this collection (optional operation).
- virtual bool **contains** (const E &value) const =0
Returns true if this collection contains the specified element.
- virtual bool **containsAll** (const **Collection**< E > &collection) const =0
Returns true if this collection contains all of the elements in the specified collection.
- virtual bool **equals** (const **Collection**< E > &value) const =0
Compares the passed collection to this one, if they contain the same elements, i.e.
- virtual bool **isEmpty** () const =0
- virtual bool **remove** (const E &value)=0
Removes a single instance of the specified element from the collection.
- virtual bool **removeAll** (const **Collection**< E > &collection)=0
Removes all this collection's elements that are also contained in the specified collection (optional operation).
- virtual bool **retainAll** (const **Collection**< E > &collection)=0
Retains only the elements in this collection that are contained in the specified collection (optional operation).
- virtual int **size** () const =0
Returns the number of elements in this collection.
- virtual std::vector< E > **toArray** () const =0
Returns an array containing all of the elements in this collection.

6.129.1 Detailed Description

template<typename E>class decaf::util::Collection< E >

The root interface in the collection hierarchy.

A collection represents a group of objects, known as its elements. Some collections allow duplicate elements and others do not. Some are ordered and others unordered. This interface is typically used to pass collections around and manipulate them where maximum generality is desired.

All general-purpose **Collection** (p. 851) implementation classes (which typically implement **Collection** (p. 851) indirectly through one of its subinterfaces) should provide two "standard" constructors: a void (no arguments) constructor, which creates an empty collection, and a constructor with a single argument of type **Collection** (p. 851), which creates a new collection with the same elements as its argument. In effect, the latter constructor allows the user to copy any collection, producing an equivalent collection

of the desired implementation type. There is no way to enforce this convention (as interfaces cannot contain constructors) but all of the general-purpose **Collection** (p. 851) implementations in the Decaf platform libraries comply.

The "destructive" methods contained in this interface, that is, the methods that modify the collection on which they operate, are specified to throw `UnsupportedOperationException` if this collection does not support the operation. If this is the case, these methods may, but are not required to, throw an `UnsupportedOperationException` if the invocation would have no effect on the collection. For example, invoking the `addAll(-Collection)` method on an unmodifiable collection may, but is not required to, throw the exception if the collection to be added is empty.

Many methods in Collections Framework interfaces are defined in terms of the `equals` method. For example, the specification for the `contains(Object o)` method says: "returns true if and only if this collection contains at least one element `e` such that (`o==null ? e==null : o.equals(e)`)."

Since

1.0

6.129.2 Constructor & Destructor Documentation

6.129.2.1 `template<typename E> virtual decaf::util::Collection< E >::~Collection ()`
`[inline, virtual]`

6.129.3 Member Function Documentation

6.129.3.1 `template<typename E> virtual bool decaf::util::Collection< E >::add (const E & value)` `[pure virtual]`

Returns true if this collection changed as a result of the call.

(Returns false if this collection does not permit duplicates and already contains the specified element.)

Collections that support this operation may place limitations on what elements may be added to this collection. In particular, some collections will refuse to add null elements, and others will impose restrictions on the type of elements that may be added. **Collection** (p. 851) classes should clearly specify in their documentation any restrictions on what elements may be added.

If a collection refuses to add a particular element for any reason other than that it already contains the element, it must throw an exception (rather than returning false). This preserves the invariant that a collection always contains the specified element after this call returns.

For non-pointer values, i.e. class instances or string's the object will be copied into the collection, thus the object must support being copied, must not hide the copy constructor and assignment operator.

Parameters

<i>value</i>	The reference to the element to add to this Collection (p. 851).
--------------	---

Returns

true if the element was added to this **Collection** (p. 851).

Exceptions

<i>UnsupportedOperationException</i>	if this is an unmodifiable collection.
<i>NullPointerException</i>	if the Collection (p. 851) is a container of pointers and does not allow NULL values.
<i>IllegalArgumentException</i>	if some property of the element prevents it from being added to this collection
<i>IllegalStateException</i>	if the element cannot be added at this time due to insertion restrictions.

Implemented in **decaf::util::StlList< E >** (p. 2542), **decaf::util::AbstractList< E >** (p. 125), **decaf::util::AbstractList< Pointer< Transport > >** (p. 125), **decaf::util::AbstractList< cms::MessageConsumer * >** (p. 125), **decaf::util::AbstractList< CompositeTask * >** (p. 125), **decaf::util::AbstractList< URI >** (p. 125), **decaf::util::AbstractList< Pointer< MessageDispatch > >** (p. 125), **decaf::util::AbstractList< Pointer< DestinationInfo > >** (p. 125), **decaf::util::AbstractList< PrimitiveValueNode >** (p. 125), **decaf::util::AbstractList< Pointer< Command > >** (p. 125), **decaf::util::AbstractList< Pointer< BackupTransport > >** (p. 125), **decaf::util::AbstractList< cms::MessageProducer * >** (p. 125), **decaf::util::AbstractList< cms::Destination * >** (p. 125), **decaf::util::AbstractList< cms::Session * >** (p. 125), **decaf::util::AbstractList< cms::Connection * >** (p. 125), **decaf::util::PriorityQueue< E >** (p. 2165), **decaf::util::StlSet< E >** (p. 2577), **decaf::util::StlSet< Pointer< Synchronization > >** (p. 2577), **decaf::util::StlSet< Resource * >** (p. 2577), **decaf::util::ArrayList< E >** (p. 448), **decaf::util::LinkedList< E >** (p. 1639), **decaf::util::LinkedList< Pointer< Transport > >** (p. 1639), **decaf::util::LinkedList< cms::MessageConsumer * >** (p. 1639), **decaf::util::LinkedList< CompositeTask * >** (p. 1639), **decaf::util::LinkedList< URI >** (p. 1639), **decaf::util::LinkedList< Pointer< MessageDispatch > >** (p. 1639), **decaf::util::LinkedList< Pointer< DestinationInfo > >** (p. 1639), **decaf::util::LinkedList< PrimitiveValueNode >** (p. 1639), **decaf::util::LinkedList< Pointer< Command > >** (p. 1639), **decaf::util::LinkedList< Pointer< BackupTransport > >** (p. 1639), **decaf::util::LinkedList< cms::MessageProducer * >** (p. 1639), **decaf::util::LinkedList< cms::Destination * >** (p. 1639), **decaf::util::LinkedList< cms::Session * >** (p. 1639), **decaf::util::LinkedList< cms::Connection * >** (p. 1639), **decaf::util::concurrent::CopyOnWriteArrayList< E >** (p. 1033), **decaf::util::concurrent::CopyOnWriteArrayList< ServiceListener * >** (p. 1033), **decaf::util::concurrent::CopyOnWriteArraySet< E >** (p. 1054), and **decaf::util::AbstractQueue< E >** (p. 139).

Referenced by **decaf::util::concurrent::SynchronousQueue< E >::drainTo()**, and **decaf::util::concurrent::LinkedBlockingQueue< E >::drainTo()**.

6.129.3.2 `template<typename E> virtual bool decaf::util::Collection< E >::addAll (const Collection< E > & collection) [pure virtual]`

Adds all of the elements in the specified collection to this collection.

The behavior of this operation is undefined if the specified collection is modified while the operation is in progress. (This implies that the behavior of this call is undefined if the specified collection is this collection, and this collection is nonempty.)

Parameters

<i>collection</i>	The Collection (p. 851) whose elements are added to this one.
-------------------	--

Returns

true if this collection changed as a result of the call

Exceptions

<i>UnsupportedOperationException</i>	if this is an unmodifiable collection.
<i>NullPointerException</i>	if the Collection (p. 851) is a container of pointers and does not allow NULL values.
<i>IllegalArgumentException</i>	if some property of an element prevents it from being added to this collection
<i>IllegalStateException</i>	if an element cannot be added at this time due to insertion restrictions.

Implemented in **decaf::util::StlList< E >** (p. 2543), **decaf::util::ArrayList< E >** (p. 450), **decaf::util::AbstractCollection< E >** (p. 110), **decaf::util::AbstractCollection< Pointer< Transport > >** (p. 110), **decaf::util::AbstractCollection< Pointer< Synchronization > >** (p. 110), **decaf::util::AbstractCollection< Resource * >** (p. 110), **decaf::util::AbstractCollection< cms::MessageConsumer * >** (p. 110), **decaf::util::AbstractCollection< CompositeTask * >** (p. 110), **decaf::util::AbstractCollection< URI >** (p. 110), **decaf::util::AbstractCollection< Pointer< MessageDispatch > >** (p. 110), **decaf::util::AbstractCollection< Pointer< DestinationInfo > >** (p. 110), **decaf::util::AbstractCollection< PrimitiveValueNode >** (p. 110), **decaf::util::AbstractCollection< Pointer< Command > >** (p. 110), **decaf::util::AbstractCollection< Pointer< BackupTransport > >** (p. 110), **decaf::util::AbstractCollection< cms::MessageProducer * >** (p. 110), **decaf::util::AbstractCollection< cms::Destination * >** (p. 110), **decaf::util::AbstractCollection< cms::Session * >** (p. 110), **decaf::util::AbstractCollection< cms::Connection * >** (p. 110), **decaf::util::LinkedList< E >** (p. 1641), **decaf::util::LinkedList< Pointer< Transport > >** (p. 1641), **decaf::util::LinkedList< cms::MessageConsumer * >** (p. 1641), **decaf::util::LinkedList< CompositeTask * >** (p. 1641), **decaf::util::LinkedList< URI >** (p. 1641), **decaf::util::LinkedList< Pointer< MessageDispatch > >** (p. 1641), **decaf::util::LinkedList< Pointer< DestinationInfo > >** (p. 1641), **decaf::util::LinkedList< PrimitiveValueNode >** (p. 1641), **decaf::util::LinkedList< Pointer< Command > >** (p. 1641), **decaf::util::LinkedList< Pointer< BackupTransport > >** (p. 1641), **decaf::util::LinkedList<**

cms::MessageProducer * > (p. 1641), **decaf::util::LinkedList**< **cms::Destination** * > (p. 1641), **decaf::util::LinkedList**< **cms::Session** * > (p. 1641), **decaf::util::LinkedList**< **cms::Connection** * > (p. 1641), **decaf::util::concurrent::CopyOnWriteArrayList**< **E** > (p. 1034), **decaf::util::concurrent::CopyOnWriteArrayList**< **ServiceListener** * > (p. 1034), **decaf::util::concurrent::CopyOnWriteArraySet**< **E** > (p. 1055), and **decaf::util::AbstractQueue**< **E** > (p. 140).

6.129.3.3 `template<typename E> virtual void decaf::util::Collection< E >::clear ()`
`[pure virtual]`

Removes all of the elements from this collection (optional operation).

This collection will be empty after this method returns unless it throws an exception.

Exceptions

<i>Unsupported- OperationException</i>	if this is an unmodifiable collection.
--	--

Implemented in **decaf::util::AbstractList**< **E** > (p. 127), **decaf::util::AbstractList**< **Pointer**< **Transport** > > (p. 127), **decaf::util::AbstractList**< **cms::MessageConsumer** * > (p. 127), **decaf::util::AbstractList**< **CompositeTask** * > (p. 127), **decaf::util::AbstractList**< **URI** > (p. 127), **decaf::util::AbstractList**< **Pointer**< **MessageDispatch** > > (p. 127), **decaf::util::AbstractList**< **Pointer**< **DestinationInfo** > > (p. 127), **decaf::util::AbstractList**< **PrimitiveValueNode** > (p. 127), **decaf::util::AbstractList**< **Pointer**< **Command** > > (p. 127), **decaf::util::AbstractList**< **Pointer**< **BackupTransport** > > (p. 127), **decaf::util::AbstractList**< **cms::MessageProducer** * > (p. 127), **decaf::util::AbstractList**< **cms::Destination** * > (p. 127), **decaf::util::AbstractList**< **cms::Session** * > (p. 127), **decaf::util::AbstractList**< **cms::Connection** * > (p. 127), **decaf::util::StlList**< **E** > (p. 2545), **decaf::util::PriorityQueue**< **E** > (p. 2166), **decaf::util::concurrent::SynchronousQueue**< **E** > (p. 2658), **decaf::util::concurrent::LinkedBlockingQueue**< **E** > (p. 1624), **decaf::util::LinkedList**< **E** > (p. 1644), **decaf::util::LinkedList**< **Pointer**< **Transport** > > (p. 1644), **decaf::util::LinkedList**< **cms::MessageConsumer** * > (p. 1644), **decaf::util::LinkedList**< **CompositeTask** * > (p. 1644), **decaf::util::LinkedList**< **URI** > (p. 1644), **decaf::util::LinkedList**< **Pointer**< **MessageDispatch** > > (p. 1644), **decaf::util::LinkedList**< **Pointer**< **DestinationInfo** > > (p. 1644), **decaf::util::LinkedList**< **PrimitiveValueNode** > (p. 1644), **decaf::util::LinkedList**< **Pointer**< **Command** > > (p. 1644), **decaf::util::LinkedList**< **Pointer**< **BackupTransport** > > (p. 1644), **decaf::util::LinkedList**< **cms::MessageProducer** * > (p. 1644), **decaf::util::LinkedList**< **cms::Destination** * > (p. 1644), **decaf::util::LinkedList**< **cms::Session** * > (p. 1644), **decaf::util::LinkedList**< **cms::Connection** * > (p. 1644), **decaf::util::concurrent::CopyOnWriteArrayList**< **E** > (p. 1036), **decaf::util::concurrent::CopyOnWriteArrayList**< **ServiceListener** * > (p. 1036), **decaf::util::StlSet**< **E** > (p. 2578), **decaf::util::StlSet**< **Pointer**< **Synchronization** > > (p. 2578), **decaf::util::StlSet**< **Resource** * > (p. 2578), **decaf::util::ArrayList**< **E** > (p. 452), **decaf::util::AbstractQueue**< **E** > (p. 141), **decaf::util::concurrent::CopyOnWriteArraySet**< **E** > (p. 1056), **decaf::util::AbstractCollection**< **E** > (p. 111), **decaf::util::AbstractCollection**< **Pointer**< **Transport** > > (p. 111), **decaf::util::AbstractCollection**< **Pointer**< **Synchronization** > >

(p. 111), **decaf::util::AbstractCollection< Resource * >** (p. 111), **decaf::util::AbstractCollection< cms::MessageConsumer * >** (p. 111), **decaf::util::AbstractCollection< CompositeTask * >** (p. 111), **decaf::util::AbstractCollection< URI >** (p. 111), **decaf::util::AbstractCollection< Pointer< MessageDispatch > >** (p. 111), **decaf::util::AbstractCollection< Pointer< DestinationInfo > >** (p. 111), **decaf::util::AbstractCollection< PrimitiveValueNode >** (p. 111), **decaf::util::AbstractCollection< Pointer< Command > >** (p. 111), **decaf::util::AbstractCollection< Pointer< BackupTransport > >** (p. 111), **decaf::util::AbstractCollection< cms::MessageProducer * >** (p. 111), **decaf::util::AbstractCollection< cms::Destination * >** (p. 111), **decaf::util::AbstractCollection< cms::Session * >** (p. 111), and **decaf::util::AbstractCollection< cms::Connection * >** (p. 111).

6.129.3.4 `template<typename E> virtual bool decaf::util::Collection< E >::contains (const E & value) const [pure virtual]`

Returns true if this collection contains the specified element.

More formally, returns true if and only if this collection contains at least one element *e* such that `(value == NULL ? e == NULL : value == e)`.

Parameters

<i>value</i>	The value to check for presence in the collection.
--------------	--

Returns

true if there is at least one of the elements in the collection

Exceptions

<i>NullPointerException</i>	if the Collection (p.851) contains pointers and the Collection (p. 851) does not allow for NULL elements (optional check).
-----------------------------	--

Implemented in **decaf::util::StlList< E >** (p. 2545), **decaf::util::ArrayList< E >** (p. 452), **decaf::util::LinkedList< E >** (p. 1644), **decaf::util::LinkedList< Pointer< Transport > >** (p. 1644), **decaf::util::LinkedList< cms::MessageConsumer * >** (p. 1644), **decaf::util::LinkedList< CompositeTask * >** (p. 1644), **decaf::util::LinkedList< URI >** (p. 1644), **decaf::util::LinkedList< Pointer< MessageDispatch > >** (p. 1644), **decaf::util::LinkedList< Pointer< DestinationInfo > >** (p. 1644), **decaf::util::LinkedList< PrimitiveValueNode >** (p. 1644), **decaf::util::LinkedList< Pointer< Command > >** (p. 1644), **decaf::util::LinkedList< Pointer< BackupTransport > >** (p. 1644), **decaf::util::LinkedList< cms::MessageProducer * >** (p. 1644), **decaf::util::LinkedList< cms::Destination * >** (p. 1644), **decaf::util::LinkedList< cms::Session * >** (p. 1644), **decaf::util::LinkedList< cms::Connection * >** (p. 1644), **decaf::util::concurrent::CopyOnWriteArrayList< E >** (p. 1037), **decaf::util::concurrent::CopyOnWriteArrayList< ServiceListener * >** (p. 1037), **decaf::util::StlSet< E >** (p. 2578), **decaf::util::StlSet< Pointer< Synchronization > >** (p. 2578), **decaf::util::StlSet< Resource * >** (p. 2578), **decaf::util::AbstractCollection< E >** (p. 112), **decaf::util::AbstractCollection<**

Pointer< **Transport** > > (p. 112), **decaf::util::AbstractCollection**< **Pointer**< **Synchronization** > > (p. 112), **decaf::util::AbstractCollection**< **Resource** * > (p. 112), **decaf::util::AbstractCollection**< **cms::MessageConsumer** * > (p. 112), **decaf::util::AbstractCollection**< **CompositeTask** * > (p. 112), **decaf::util::AbstractCollection**< **URI** > (p. 112), **decaf::util::AbstractCollection**< **Pointer**< **MessageDispatch** > > (p. 112), **decaf::util::AbstractCollection**< **Pointer**< **DestinationInfo** > > (p. 112), **decaf::util::AbstractCollection**< **PrimitiveValueNode** > (p. 112), **decaf::util::AbstractCollection**< **Pointer**< **Command** > > (p. 112), **decaf::util::AbstractCollection**< **Pointer**< **BackupTransport** > > (p. 112), **decaf::util::AbstractCollection**< **cms::MessageProducer** * > (p. 112), **decaf::util::AbstractCollection**< **cms::Destination** * > (p. 112), **decaf::util::AbstractCollection**< **cms::Session** * > (p. 112), **decaf::util::AbstractCollection**< **cms::Connection** * > (p. 112), and **decaf::util::concurrent::CopyOnWriteArraySet**< **E** > (p. 1056).

Referenced by **decaf::util::AbstractSet**< **Resource** * >::**removeAll**(), **decaf::util::AbstractCollection**< **cms::Connection** * >::**removeAll**(), **decaf::util::concurrent::CopyOnWriteArrayList**< **ServiceListener** * >::**removeAll**(), **decaf::util::AbstractCollection**< **cms::Connection** * >::**retainAll**(), and **decaf::util::concurrent::CopyOnWriteArrayList**< **ServiceListener** * >::**retainAll**().

6.129.3.5 **template**<typename **E**> **virtual bool decaf::util::Collection**< **E** >::**containsAll** (**const Collection**< **E** > & *collection*) **const** [pure virtual]

Returns true if this collection contains all of the elements in the specified collection.

Parameters

<i>collection</i>	The Collection (p. 851) to compare to this one.
-------------------	--

Exceptions

<i>NullPointerException</i>	if the Collection (p. 851) contains pointers and the Collection (p. 851) does not allow for NULL elements (optional check).
-----------------------------	---

Implemented in **decaf::util::concurrent::SynchronousQueue**< **E** > (p. 2658), **decaf::util::concurrent::CopyOnWriteArrayList**< **E** > (p. 1037), **decaf::util::concurrent::CopyOnWriteArrayList**< **ServiceListener** * > (p. 1037), **decaf::util::AbstractCollection**< **E** > (p. 113), **decaf::util::AbstractCollection**< **Pointer**< **Transport** > > (p. 113), **decaf::util::AbstractCollection**< **Pointer**< **Synchronization** > > (p. 113), **decaf::util::AbstractCollection**< **Resource** * > (p. 113), **decaf::util::AbstractCollection**< **cms::MessageConsumer** * > (p. 113), **decaf::util::AbstractCollection**< **CompositeTask** * > (p. 113), **decaf::util::AbstractCollection**< **URI** > (p. 113), **decaf::util::AbstractCollection**< **Pointer**< **MessageDispatch** > > (p. 113), **decaf::util::AbstractCollection**< **Pointer**< **DestinationInfo** > > (p. 113), **decaf::util::AbstractCollection**< **PrimitiveValueNode** > (p. 113), **decaf::util::AbstractCollection**< **Pointer**< **Command** > > (p. 113), **decaf::util::AbstractCollection**< **Pointer**< **BackupTransport** > > (p. 113), **decaf::util::AbstractCollection**< **cms::MessageProducer** * > (p. 113), **decaf::util::AbstractCollection**< **cms::Destination** * > (p. 113), **decaf::util::AbstractCollection**< **cms::Session** * > (p. 113),

decaf::util::AbstractCollection< **cms::Connection** * > (p. 113), and **decaf::util::concurrent::CopyOnWriteArraySet**< **E** > (p. 1057).

6.129.3.6 `template<typename E> virtual void decaf::util::Collection< E >::copy (const Collection< E > & collection) [pure virtual]`

Renders this **Collection** (p. 851) as a Copy of the given **Collection** (p. 851).

Parameters

<i>collection</i>	The collection to mirror.
-------------------	---------------------------

Exceptions

<i>UnsupportedOperationException</i>	if this is an unmodifiable collection.
<i>IllegalStateException</i>	if the elements cannot be added at this time due to insertion restrictions.

Implemented in **decaf::util::StlList**< **E** > (p. 2546), **decaf::util::LinkedList**< **E** > (p. 1645), **decaf::util::LinkedList**< **Pointer**< **Transport** > > (p. 1645), **decaf::util::LinkedList**< **cms::MessageConsumer** * > (p. 1645), **decaf::util::LinkedList**< **CompositeTask** * > (p. 1645), **decaf::util::LinkedList**< **URI** > (p. 1645), **decaf::util::LinkedList**< **Pointer**< **MessageDispatch** > > (p. 1645), **decaf::util::LinkedList**< **Pointer**< **DestinationInfo** > > (p. 1645), **decaf::util::LinkedList**< **PrimitiveValueNode** > (p. 1645), **decaf::util::LinkedList**< **Pointer**< **Command** > > (p. 1645), **decaf::util::LinkedList**< **Pointer**< **BackupTransport** > > (p. 1645), **decaf::util::LinkedList**< **cms::MessageProducer** * > (p. 1645), **decaf::util::LinkedList**< **cms::Destination** * > (p. 1645), **decaf::util::LinkedList**< **cms::Session** * > (p. 1645), **decaf::util::LinkedList**< **cms::Connection** * > (p. 1645), **decaf::util::StlSet**< **E** > (p. 2579), **decaf::util::StlSet**< **Pointer**< **Synchronization** > > (p. 2579), **decaf::util::StlSet**< **Resource** * > (p. 2579), **decaf::util::AbstractCollection**< **E** > (p. 113), **decaf::util::AbstractCollection**< **Pointer**< **Transport** > > (p. 113), **decaf::util::AbstractCollection**< **Pointer**< **Synchronization** > > (p. 113), **decaf::util::AbstractCollection**< **Resource** * > (p. 113), **decaf::util::AbstractCollection**< **cms::MessageConsumer** * > (p. 113), **decaf::util::AbstractCollection**< **CompositeTask** * > (p. 113), **decaf::util::AbstractCollection**< **URI** > (p. 113), **decaf::util::AbstractCollection**< **Pointer**< **MessageDispatch** > > (p. 113), **decaf::util::AbstractCollection**< **Pointer**< **DestinationInfo** > > (p. 113), **decaf::util::AbstractCollection**< **PrimitiveValueNode** > (p. 113), **decaf::util::AbstractCollection**< **Pointer**< **Command** > > (p. 113), **decaf::util::AbstractCollection**< **Pointer**< **BackupTransport** > > (p. 113), **decaf::util::AbstractCollection**< **cms::MessageProducer** * > (p. 113), **decaf::util::AbstractCollection**< **cms::Destination** * > (p. 113), **decaf::util::AbstractCollection**< **cms::Session** * > (p. 113), **decaf::util::AbstractCollection**< **cms::Connection** * > (p. 113), **decaf::util::concurrent::CopyOnWriteArrayList**< **E** > (p. 1038), **decaf::util::concurrent::CopyOnWriteArrayList**< **ServiceListener** * > (p. 1038), and **decaf::util::concurrent::CopyOnWriteArraySet**< **E** > (p. 1057).

```
6.129.3.7  template<typename E> virtual bool decaf::util::Collection< E >::equals (
            const Collection< E > & value ) const    [pure virtual]
```

Compares the passed collection to this one, if they contain the same elements, i.e. all their elements are equivalent, then it returns true.

Returns

true if the Collections contain the same elements.

Implemented in **decaf::util::StlList< E >** (p.2546), **decaf::util::concurrent::CopyOnWriteArrayList< E >** (p.1038), **decaf::util::concurrent::CopyOnWriteArrayList< ServiceListener * >** (p.1038), **decaf::util::concurrent::SynchronousQueue< E >** (p.2660), **decaf::util::StlSet< E >** (p.2580), **decaf::util::StlSet< Pointer< Synchronization > >** (p.2580), **decaf::util::StlSet< Resource * >** (p.2580), **decaf::util::AbstractCollection< E >** (p.114), **decaf::util::AbstractCollection< Pointer< Transport > >** (p.114), **decaf::util::AbstractCollection< Pointer< Synchronization > >** (p.114), **decaf::util::AbstractCollection< Resource * >** (p.114), **decaf::util::AbstractCollection< cms::MessageConsumer * >** (p.114), **decaf::util::AbstractCollection< CompositeTask * >** (p.114), **decaf::util::AbstractCollection< URI >** (p.114), **decaf::util::AbstractCollection< Pointer< MessageDispatch > >** (p.114), **decaf::util::AbstractCollection< Pointer< DestinationInfo > >** (p.114), **decaf::util::AbstractCollection< PrimitiveValueNode >** (p.114), **decaf::util::AbstractCollection< Pointer< Command > >** (p.114), **decaf::util::AbstractCollection< Pointer< BackupTransport > >** (p.114), **decaf::util::AbstractCollection< cms::MessageProducer * >** (p.114), **decaf::util::AbstractCollection< cms::Destination * >** (p.114), **decaf::util::AbstractCollection< cms::Session * >** (p.114), **decaf::util::AbstractCollection< cms::Connection * >** (p.114), and **decaf::util::concurrent::CopyOnWriteArraySet< E >** (p.1058).

```
6.129.3.8  template<typename E> virtual bool decaf::util::Collection< E >::isEmpty ( )
            const    [pure virtual]
```

Returns

true if this collection contains no elements.

Implemented in **decaf::util::StlList< E >** (p.2548), **decaf::util::concurrent::CopyOnWriteArrayList< E >** (p.1040), **decaf::util::concurrent::CopyOnWriteArrayList< ServiceListener * >** (p.1040), **decaf::util::concurrent::SynchronousQueue< E >** (p.2660), **decaf::util::LinkedList< E >** (p.1648), **decaf::util::LinkedList< Pointer< Transport > >** (p.1648), **decaf::util::LinkedList< cms::MessageConsumer * >** (p.1648), **decaf::util::LinkedList< CompositeTask * >** (p.1648), **decaf::util::LinkedList< URI >** (p.1648), **decaf::util::LinkedList< Pointer< MessageDispatch > >** (p.1648), **decaf::util::LinkedList< Pointer< DestinationInfo > >** (p.1648), **decaf::util::LinkedList< PrimitiveValueNode >** (p.1648), **decaf::util::LinkedList< Pointer< Command > >** (p.1648), **decaf::util::LinkedList< Pointer< BackupTransport > >** (p.1648), **decaf::util::LinkedList<**

cms::MessageProducer * > (p. 1648), decaf::util::LinkedList< cms::Destination * > (p. 1648), decaf::util::LinkedList< cms::Session * > (p. 1648), decaf::util::LinkedList< cms::Connection * > (p. 1648), decaf::util::StlSet< E > (p. 2580), decaf::util::StlSet< Pointer< Synchronization > > (p. 2580), decaf::util::StlSet< Resource * > (p. 2580), decaf::util::AbstractCollection< E > (p. 114), decaf::util::AbstractCollection< Pointer< Transport > > (p. 114), decaf::util::AbstractCollection< Pointer< Synchronization > > (p. 114), decaf::util::AbstractCollection< Resource * > (p. 114), decaf::util::AbstractCollection< cms::MessageConsumer * > (p. 114), decaf::util::AbstractCollection< CompositeTask * > (p. 114), decaf::util::AbstractCollection< URI > (p. 114), decaf::util::AbstractCollection< Pointer< MessageDispatch > > (p. 114), decaf::util::AbstractCollection< Pointer< DestinationInfo > > (p. 114), decaf::util::AbstractCollection< PrimitiveValueNode > > (p. 114), decaf::util::AbstractCollection< Pointer< Command > > (p. 114), decaf::util::AbstractCollection< Pointer< BackupTransport > > (p. 114), decaf::util::AbstractCollection< cms::MessageProducer * > (p. 114), decaf::util::AbstractCollection< cms::Destination * > (p. 114), decaf::util::AbstractCollection< cms::Session * > (p. 114), decaf::util::AbstractCollection< cms::Connection * > (p. 114), decaf::util::ArrayList< E > (p. 454), and decaf::util::concurrent::CopyOnWriteArraySet< E > (p. 1058).

Referenced by decaf::util::AbstractList< cms::Connection * >::addAll(), decaf::util::StlList< E >::addAll(), decaf::util::concurrent::SynchronousQueue< E >::containsAll(), decaf::util::concurrent::CopyOnWriteArrayList< ServiceListener * >::removeAll(), and decaf::util::concurrent::CopyOnWriteArrayList< ServiceListener * >::retainAll().

6.129.3.9 `template<typename E> virtual bool decaf::util::Collection< E >::remove (const E & value) [pure virtual]`

Removes a single instance of the specified element from the collection.

More formally, removes an element *e* such that (*value* == NULL ? *e* == NULL : *value* == *e*), if this collection contains one or more such elements. Returns true if this collection contained the specified element (or equivalently, if this collection changed as a result of the call).

Parameters

<i>value</i>	The reference to the element to remove from this Collection (p. 851).
--------------	--

Returns

true if the collection was changed, false otherwise.

Exceptions

<i>UnsupportedOperationException</i>	if this is an unmodifiable collection.
<i>NullPointerException</i>	if the Collection (p. 851) is a container of pointers and does not allow NULL values.

Implemented in `decaf::util::StlList< E >` (p. 2550), `decaf::util::concurrent::LinkedBlockingQueue< E >` (p. 1631), `decaf::util::PriorityQueue< E >` (p. 2170), `decaf::util::ArrayList< E >` (p. 455), `decaf::util::concurrent::CopyOnWriteArrayList< E >` (p. 1043), `decaf::util::concurrent::CopyOnWriteArrayList< ServiceListener * >` (p. 1043), `decaf::util::AbstractCollection< E >` (p. 116), `decaf::util::AbstractCollection< Pointer< Transport > >` (p. 116), `decaf::util::AbstractCollection< Pointer< Synchronization > >` (p. 116), `decaf::util::AbstractCollection< Resource * >` (p. 116), `decaf::util::AbstractCollection< cms::MessageConsumer * >` (p. 116), `decaf::util::AbstractCollection< CompositeTask * >` (p. 116), `decaf::util::AbstractCollection< URI >` (p. 116), `decaf::util::AbstractCollection< Pointer< MessageDispatch > >` (p. 116), `decaf::util::AbstractCollection< Pointer< DestinationInfo > >` (p. 116), `decaf::util::AbstractCollection< PrimitiveValueNode >` (p. 116), `decaf::util::AbstractCollection< Pointer< Command > >` (p. 116), `decaf::util::AbstractCollection< Pointer< BackupTransport > >` (p. 116), `decaf::util::AbstractCollection< cms::MessageProducer * >` (p. 116), `decaf::util::AbstractCollection< cms::Destination * >` (p. 116), `decaf::util::AbstractCollection< cms::Session * >` (p. 116), `decaf::util::AbstractCollection< cms::Connection * >` (p. 116), `decaf::util::StlSet< E >` (p. 2580), `decaf::util::StlSet< Pointer< Synchronization > >` (p. 2580), `decaf::util::StlSet< Resource * >` (p. 2580), `decaf::util::LinkedList< E >` (p. 1654), `decaf::util::LinkedList< Pointer< Transport > >` (p. 1654), `decaf::util::LinkedList< cms::MessageConsumer * >` (p. 1654), `decaf::util::LinkedList< CompositeTask * >` (p. 1654), `decaf::util::LinkedList< URI >` (p. 1654), `decaf::util::LinkedList< Pointer< MessageDispatch > >` (p. 1654), `decaf::util::LinkedList< Pointer< DestinationInfo > >` (p. 1654), `decaf::util::LinkedList< PrimitiveValueNode >` (p. 1654), `decaf::util::LinkedList< Pointer< Command > >` (p. 1654), `decaf::util::LinkedList< Pointer< BackupTransport > >` (p. 1654), `decaf::util::LinkedList< cms::MessageProducer * >` (p. 1654), `decaf::util::LinkedList< cms::Destination * >` (p. 1654), `decaf::util::LinkedList< cms::Session * >` (p. 1654), `decaf::util::LinkedList< cms::Connection * >` (p. 1654), and `decaf::util::concurrent::CopyOnWriteArraySet< E >` (p. 1059).

6.129.3.10 `template<typename E> virtual bool decaf::util::Collection< E >::removeAll (const Collection< E > & collection) [pure virtual]`

Removes all this collection's elements that are also contained in the specified collection (optional operation).

After this call returns, this collection will contain no elements in common with the specified collection.

Parameters

<i>collection</i>	The Collection (p. 851) whose elements are to be removed from this one.
-------------------	--

Returns

true if the collection changed as a result of this call.

Exceptions

<i>UnsupportedOperationException</i>	if this is an unmodifiable collection.
<i>NullPointerException</i>	if the Collection (p. 851) is a container of pointers and does not allow NULL values.

Implemented in **decaf::util::concurrent::CopyOnWriteArrayList< E >** (p. 1044), **decaf::util::concurrent::CopyOnWriteArrayList< ServiceListener * >** (p. 1044), **decaf::util::AbstractCollection< E >** (p. 117), **decaf::util::AbstractCollection< Pointer< Transport > >** (p. 117), **decaf::util::AbstractCollection< Pointer< Synchronization > >** (p. 117), **decaf::util::AbstractCollection< Resource * >** (p. 117), **decaf::util::AbstractCollection< cms::MessageConsumer * >** (p. 117), **decaf::util::AbstractCollection< CompositeTask * >** (p. 117), **decaf::util::AbstractCollection< URI >** (p. 117), **decaf::util::AbstractCollection< Pointer< MessageDispatch > >** (p. 117), **decaf::util::AbstractCollection< Pointer< DestinationInfo > >** (p. 117), **decaf::util::AbstractCollection< PrimitiveValueNode >** (p. 117), **decaf::util::AbstractCollection< Pointer< Command > >** (p. 117), **decaf::util::AbstractCollection< Pointer< BackupTransport > >** (p. 117), **decaf::util::AbstractCollection< cms::MessageProducer * >** (p. 117), **decaf::util::AbstractCollection< cms::Destination * >** (p. 117), **decaf::util::AbstractCollection< cms::Session * >** (p. 117), **decaf::util::AbstractCollection< cms::Connection * >** (p. 117), **decaf::util::concurrent::CopyOnWriteArraySet< E >** (p. 1059), **decaf::util::AbstractSet< E >** (p. 153), **decaf::util::AbstractSet< Pointer< Synchronization > >** (p. 153), and **decaf::util::AbstractSet< Resource * >** (p. 153).

6.129.3.11 `template<typename E> virtual bool decaf::util::Collection< E >::retainAll (const Collection< E > & collection) [pure virtual]`

Retains only the elements in this collection that are contained in the specified collection (optional operation).

In other words, removes from this collection all of its elements that are not contained in the specified collection.

Parameters

<i>collection</i>	The Collection (p. 851) whose elements are to be retained.
-------------------	---

Returns

true if the collection changed as a result of this call.

Exceptions

<i>UnsupportedOperationException</i>	if this is an unmodifiable collection.
<i>NullPointerException</i>	if the Collection (p. 851) is a container of pointers and does not allow NULL values.

Implemented in `decaf::util::concurrent::CopyOnWriteArrayList< E >` (p. 1045), `decaf::util::concurrent::CopyOnWriteArrayList< ServiceListener * >` (p. 1045), `decaf::util::AbstractCollection< E >` (p. 118), `decaf::util::AbstractCollection< Pointer< Transport > >` (p. 118), `decaf::util::AbstractCollection< Pointer< Synchronization > >` (p. 118), `decaf::util::AbstractCollection< Resource * >` (p. 118), `decaf::util::AbstractCollection< cms::MessageConsumer * >` (p. 118), `decaf::util::AbstractCollection< CompositeTask * >` (p. 118), `decaf::util::AbstractCollection< URI >` (p. 118), `decaf::util::AbstractCollection< Pointer< MessageDispatch > >` (p. 118), `decaf::util::AbstractCollection< Pointer< DestinationInfo > >` (p. 118), `decaf::util::AbstractCollection< PrimitiveValueNode >` (p. 118), `decaf::util::AbstractCollection< Pointer< Command > >` (p. 118), `decaf::util::AbstractCollection< Pointer< BackupTransport > >` (p. 118), `decaf::util::AbstractCollection< cms::MessageProducer * >` (p. 118), `decaf::util::AbstractCollection< cms::Destination * >` (p. 118), `decaf::util::AbstractCollection< cms::Session * >` (p. 118), `decaf::util::AbstractCollection< cms::Connection * >` (p. 118), and `decaf::util::concurrent::CopyOnWriteArraySet< E >` (p. 1061).

6.129.3.12 `template<typename E> virtual int decaf::util::Collection< E >::size () const`
`[pure virtual]`

Returns the number of elements in this collection.

If this collection contains more than `Integer::MAX_VALUE` elements, returns `Integer::MAX_VALUE`.

Returns

the number of elements in this collection

Implemented in `decaf::util::concurrent::CopyOnWriteArrayList< E >` (p. 1046), `decaf::util::concurrent::CopyOnWriteArrayList< ServiceListener * >` (p. 1046), `decaf::util::StlList< E >` (p. 2552), `decaf::util::PriorityQueue< E >` (p. 2171), `decaf::util::concurrent::SynchronousQueue< E >` (p. 2664), `decaf::util::concurrent::LinkedBlockingQueue< E >` (p. 1632), `decaf::util::LinkedList< E >` (p. 1658), `decaf::util::LinkedList< Pointer< Transport > >` (p. 1658), `decaf::util::LinkedList< cms::MessageConsumer * >` (p. 1658), `decaf::util::LinkedList< CompositeTask * >` (p. 1658), `decaf::util::LinkedList< URI >` (p. 1658), `decaf::util::LinkedList< Pointer< MessageDispatch > >` (p. 1658), `decaf::util::LinkedList< Pointer< DestinationInfo > >` (p. 1658), `decaf::util::LinkedList< PrimitiveValueNode >` (p. 1658), `decaf::util::LinkedList< Pointer< Command > >` (p. 1658), `decaf::util::LinkedList< Pointer< BackupTransport > >` (p. 1658), `decaf::util::LinkedList< cms::MessageProducer * >` (p. 1658), `decaf::util::LinkedList< cms::Destination * >` (p. 1658), `decaf::util::LinkedList< cms::Session * >` (p. 1658), `decaf::util::LinkedList< cms::Connection * >` (p. 1658), `decaf::util::StlSet< E >` (p. 2581), `decaf::util::StlSet< Pointer< Synchronization > >` (p. 2581), `decaf::util::StlSet< Resource * >` (p. 2581), `decaf::util::ArrayList< E >` (p. 457), and `decaf::util::concurrent::CopyOnWriteArraySet< E >` (p. 1061).

Referenced by `decaf::util::AbstractList< cms::Connection * >::add()`, `decaf::util::concurrent::CopyOnWriteArrayList< ServiceListener * >::addAll()`, `decaf::util::ArrayList< E >::addAll()`, `decaf::util::concurrent::CopyOnWriteArrayList< ServiceListener`

* >::addAllAbsent(), decaf::util::ArrayList< E >::ArrayList(), decaf::util::AbstractList< cms::Connection * >::clear(), decaf::util::concurrent::CopyOnWriteArraySet< E >::equals(), decaf::util::AbstractCollection< cms::Connection * >::equals(), decaf::util::concurrent::CopyOnWriteArrayList< ServiceListener * >::equals(), decaf::util::AbstractCollection< cms::Connection * >::isEmpty(), decaf::util::AbstractList< cms::Connection * >::lastIndexOf(), decaf::util::AbstractSet< Resource * >::removeAll(), and decaf::util::AbstractCollection< cms::Connection * >::toArray().

6.129.3.13 `template<typename E> virtual std::vector<E> decaf::util::Collection< E >::toArray () const [pure virtual]`

Returns an array containing all of the elements in this collection.

If the collection makes any guarantees as to what order its elements are returned by its iterator, this method must return the elements in the same order.

This method acts as bridge between array-based and collection-based APIs.

Returns

an array of the elements in this collection in the form of an STL vector.

Implemented in `decaf::util::concurrent::LinkedBlockingQueue< E >` (p. 1632), `decaf::util::concurrent::CopyOnWriteArrayList< E >` (p. 1047), `decaf::util::concurrent::CopyOnWriteArrayList< ServiceListener * >` (p. 1047), `decaf::util::ArrayList< E >` (p. 457), `decaf::util::AbstractCollection< E >` (p. 119), `decaf::util::AbstractCollection< Pointer< Transport > >` (p. 119), `decaf::util::AbstractCollection< Pointer< Synchronization > >` (p. 119), `decaf::util::AbstractCollection< Resource * >` (p. 119), `decaf::util::AbstractCollection< cms::MessageConsumer * >` (p. 119), `decaf::util::AbstractCollection< CompositeTask * >` (p. 119), `decaf::util::AbstractCollection< URI >` (p. 119), `decaf::util::AbstractCollection< Pointer< MessageDispatch > >` (p. 119), `decaf::util::AbstractCollection< Pointer< DestinationInfo > >` (p. 119), `decaf::util::AbstractCollection< PrimitiveValueNode >` (p. 119), `decaf::util::AbstractCollection< Pointer< Command > >` (p. 119), `decaf::util::AbstractCollection< Pointer< BackupTransport > >` (p. 119), `decaf::util::AbstractCollection< cms::MessageProducer * >` (p. 119), `decaf::util::AbstractCollection< cms::Destination * >` (p. 119), `decaf::util::AbstractCollection< cms::Session * >` (p. 119), `decaf::util::AbstractCollection< cms::Connection * >` (p. 119), `decaf::util::LinkedList< E >` (p. 1658), `decaf::util::LinkedList< Pointer< Transport > >` (p. 1658), `decaf::util::LinkedList< cms::MessageConsumer * >` (p. 1658), `decaf::util::LinkedList< CompositeTask * >` (p. 1658), `decaf::util::LinkedList< URI >` (p. 1658), `decaf::util::LinkedList< Pointer< MessageDispatch > >` (p. 1658), `decaf::util::LinkedList< Pointer< DestinationInfo > >` (p. 1658), `decaf::util::LinkedList< PrimitiveValueNode >` (p. 1658), `decaf::util::LinkedList< Pointer< Command > >` (p. 1658), `decaf::util::LinkedList< Pointer< BackupTransport > >` (p. 1658), `decaf::util::LinkedList< cms::MessageProducer * >` (p. 1658), `decaf::util::LinkedList< cms::Destination * >` (p. 1658), `decaf::util::LinkedList< cms::Session * >` (p. 1658), `decaf::util::LinkedList< cms::Connection * >` (p. 1658), `decaf::util::concurrent::SynchronousQueue< E >` (p. 2665), and `decaf::util::concurrent::CopyOnWriteArraySet< E >` (p. 1062).

Referenced by `decaf::util::ArrayList< E >::addAll()`, and `decaf::util::StlList< E >::addAll()`.

The documentation for this class was generated from the following file:

- `src/main/decaf/util/Collection.h`

6.130 activemq::commands::Command Class Reference

```
#include <src/main/activemq/commands/Command.h>
```

Inheritance diagram for `activemq::commands::Command`:

Public Member Functions

- virtual `~Command ()`
- virtual void `setCommandId (int id)=0`
*Sets the **Command** (p. 866) Id of this **Message** (p. 1821).*
- virtual int `getCommandId () const =0`
*Gets the **Command** (p. 866) Id of this **Message** (p. 1821).*
- virtual void `setResponseRequired (const bool required)=0`
*Set if this **Message** (p. 1821) requires a **Response** (p. 2298).*
- virtual bool `isResponseRequired () const =0`
*Is a **Response** (p. 2298) required for this **Command** (p. 866).*
- virtual `std::string toString () const =0`
Returns a provider-specific string that provides information about the contents of the command.
- virtual `decaf::lang::Pointer < commands::Command > visit (activemq::state::CommandVisitor *visitor)=0`
Allows a Visitor to visit this command and return a response to the command based on the command type being visited.
- virtual bool `isConnectionControl () const =0`
- virtual bool `isConnectionInfo () const =0`
- virtual bool `isConsumerInfo () const =0`
- virtual bool `isBrokerInfo () const =0`
- virtual bool `isKeepAliveInfo () const =0`
- virtual bool `isMessage () const =0`
- virtual bool `isMessageAck () const =0`
- virtual bool `isMessageDispatch () const =0`
- virtual bool `isMessageDispatchNotification () const =0`
- virtual bool `isProducerAck () const =0`
- virtual bool `isProducerInfo () const =0`
- virtual bool `isResponse () const =0`

- virtual bool **isRemoveInfo** () const =0
- virtual bool **isRemoveSubscriptionInfo** () const =0
- virtual bool **isShutdownInfo** () const =0
- virtual bool **isTransactionInfo** () const =0
- virtual bool **isWireFormatInfo** () const =0

6.130.1 Constructor & Destructor Documentation

6.130.1.1 virtual **activemq::commands::Command::~~Command** () [inline, virtual]

6.130.2 Member Function Documentation

6.130.2.1 virtual int **activemq::commands::Command::getCommandId** () const [pure virtual]

Gets the **Command** (p. 866) Id of this **Message** (p. 1821).

Returns

Command (p. 866) Id

Implemented in **activemq::commands::BaseCommand** (p. 495).

6.130.2.2 virtual bool **activemq::commands::Command::isBrokerInfo** () const [pure virtual]

Implemented in **activemq::commands::BrokerInfo** (p. 575), and **activemq::commands::BaseCommand** (p. 495).

6.130.2.3 virtual bool **activemq::commands::Command::isConnectionControl** () const [pure virtual]

Implemented in **activemq::commands::ConnectionControl** (p. 941), and **activemq::commands::BaseCommand** (p. 495).

6.130.2.4 virtual bool **activemq::commands::Command::isConnectionInfo** () const [pure virtual]

Implemented in **activemq::commands::ConnectionInfo** (p. 971), and **activemq::commands::BaseCommand** (p. 495).

6.130.2.5 `virtual bool activemq::commands::Command::isConsumerInfo () const`
[pure virtual]

Implemented in `activemq::commands::ConsumerInfo` (p. 1014), and `activemq::commands::BaseCommand` (p. 495).

6.130.2.6 `virtual bool activemq::commands::Command::isKeepAliveInfo () const`
[pure virtual]

Implemented in `activemq::commands::BaseCommand` (p. 495), and `activemq::commands::KeepAliveInfo` (p. 1593).

6.130.2.7 `virtual bool activemq::commands::Command::isMessage () const`
[pure virtual]

Implemented in `activemq::commands::Message` (p. 1832), and `activemq::commands::BaseCommand` (p. 496).

6.130.2.8 `virtual bool activemq::commands::Command::isMessageAck () const`
[pure virtual]

Implemented in `activemq::commands::MessageAck` (p. 1871), and `activemq::commands::BaseCommand` (p. 496).

6.130.2.9 `virtual bool activemq::commands::Command::isMessageDispatch () const`
[pure virtual]

Implemented in `activemq::commands::BaseCommand` (p. 496), and `activemq::commands::MessageDispatch` (p. 1884).

6.130.2.10 `virtual bool activemq::commands::Command::isMessageDispatchNotification () const`
[pure virtual]

Implemented in `activemq::commands::BaseCommand` (p. 496), and `activemq::commands::MessageDispatchNotification` (p. 1898).

6.130.2.11 `virtual bool activemq::commands::Command::isProducerAck () const`
[pure virtual]

Implemented in `activemq::commands::BaseCommand` (p. 496), and `activemq::commands::ProducerAck` (p. 2173).

6.130.2.12 `virtual bool activemq::commands::Command::isProducerInfo () const`
[pure virtual]

Implemented in **activemq::commands::BaseCommand** (p.496), and **activemq::commands::ProducerInfo** (p.2193).

6.130.2.13 `virtual bool activemq::commands::Command::isRemoveInfo () const`
[pure virtual]

Implemented in **activemq::commands::BaseCommand** (p.496), and **activemq::commands::RemoveInfo** (p.2270).

6.130.2.14 `virtual bool activemq::commands::Command::isRemoveSubscriptionInfo () const` [pure virtual]

Implemented in **activemq::commands::BaseCommand** (p.497), and **activemq::commands::RemoveSubscriptionInfo** (p.2278).

6.130.2.15 `virtual bool activemq::commands::Command::isResponse () const`
[pure virtual]

Implemented in **activemq::commands::BaseCommand** (p.497), and **activemq::commands::Response** (p.2300).

6.130.2.16 `virtual bool activemq::commands::Command::isResponseRequired () const` [pure virtual]

Is a **Response** (p.2298) required for this **Command** (p.866).

Returns

true if a response is required.

Implemented in **activemq::commands::BaseCommand** (p.497).

6.130.2.17 `virtual bool activemq::commands::Command::isShutdownInfo () const`
[pure virtual]

Implemented in **activemq::commands::BaseCommand** (p.497), and **activemq::commands::ShutdownInfo** (p.2433).

6.130.2.18 `virtual bool activemq::commands::Command::isTransactionInfo () const`
[pure virtual]

Implemented in **activemq::commands::BaseCommand** (p.497), and **activemq::commands::TransactionInfo** (p.2777).

6.130.2.19 `virtual bool activemq::commands::Command::isWireFormatInfo () const`
`[pure virtual]`

Implemented in `activemq::commands::WireFormatInfo` (p. 2896), and `activemq::commands::BaseCommand` (p. 497).

6.130.2.20 `virtual void activemq::commands::Command::setCommandId (int id)`
`[pure virtual]`

Sets the **Command** (p. 866) Id of this **Message** (p. 1821).

Parameters

<i>id</i>	Command (p. 866) Id
-----------	----------------------------

Implemented in `activemq::commands::BaseCommand` (p. 498).

6.130.2.21 `virtual void activemq::commands::Command::setResponseRequired (`
`const bool required) [pure virtual]`

Set if this **Message** (p. 1821) requires a **Response** (p. 2298).

Parameters

<i>required</i>	true if response is required
-----------------	------------------------------

Implemented in `activemq::commands::BaseCommand` (p. 498).

6.130.2.22 `virtual std::string activemq::commands::Command::toString () const`
`[pure virtual]`

Returns a provider-specific string that provides information about the contents of the command.

Reimplemented from `activemq::commands::BaseDataStructure` (p. 531).

Implemented in `activemq::commands::Message` (p. 1836), `activemq::commands::ConsumerInfo` (p. 1016), `activemq::commands::ActiveMQBytesMessage` (p. 178), `activemq::commands::BrokerInfo` (p. 577), `activemq::commands::ConnectionInfo` (p. 972), `activemq::commands::MessageAck` (p. 1871), `activemq::commands::ConsumerControl` (p. 995), `activemq::commands::ProducerInfo` (p. 2194), `activemq::commands::ConnectionControl` (p. 942), `activemq::commands::DestinationInfo` (p. 1217), `activemq::commands::MessagePull` (p. 1943), `activemq::commands::SessionInfo` (p. 2389), `activemq::commands::MessageDispatch` (p. 1885), `activemq::commands::MessageDispatchNotification` (p. 1898), `activemq::commands::TransactionInfo` (p. 2777), `activemq::commands::ConnectionError` (p. 950), `activemq::commands::RemoveSubscriptionInfo` (p. 2279), `activemq::commands::ActiveMQStreamMessage` (p. 369), `activemq::commands::ProducerAck` (p. 2174), `activemq::commands::`

::RemoveInfo (p. 2270), **activemq::commands::ActiveMQMapMessage** (p. 283), **activemq::commands::DataArrayResponse** (p. 1071), **activemq::commands::DataResponse** (p. 1114), **activemq::commands::ExceptionResponse** (p. 1290), **activemq::commands::ReplayCommand** (p. 2286), **activemq::commands::ControlCommand** (p. 1025), **activemq::commands::IntegerResponse** (p. 1518), **activemq::commands::Response** (p. 2300), **activemq::commands::FlushCommand** (p. 1383), **activemq::commands::KeepAliveInfo** (p. 1593), **activemq::commands::ShutdownInfo** (p. 2433), **activemq::commands::BaseCommand** (p. 498), **activemq::commands::ActiveMQBlobMessage** (p. 161), **activemq::commands::WireFormatInfo** (p. 2899), **activemq::commands::ActiveMQTextMessage** (p. 409), **activemq::commands::ActiveMQObjectMessage** (p. 303), and **activemq::commands::ActiveMQMessage** (p. 290).

```
6.130.2.23 virtual decaf::lang::Pointer<commands::Command> activemq-
::commands::Command::visit ( activemq::state::CommandVisitor *
visitor ) [pure virtual]
```

Allows a Visitor to visit this command and return a response to the command based on the command type being visited.

The command will call the proper processXXX method in the visitor.

Returns

a **Response** (p. 2298) to the visitor being called or NULL if no response.

Implemented in **activemq::commands::Message** (p. 1837), **activemq::commands::ConsumerInfo** (p. 1016), **activemq::commands::BrokerInfo** (p. 577), **activemq::commands::ConnectionInfo** (p. 973), **activemq::commands::MessageAck** (p. 1872), **activemq::commands::ConnectionControl** (p. 942), **activemq::commands::ProducerInfo** (p. 2194), **activemq::commands::ConsumerControl** (p. 995), **activemq::commands::MessageDispatch** (p. 1885), **activemq::commands::MessageDispatchNotification** (p. 1898), **activemq::commands::MessagePull** (p. 1943), **activemq::commands::DestinationInfo** (p. 1217), **activemq::commands::RemoveSubscriptionInfo** (p. 2279), **activemq::commands::TransactionInfo** (p. 2777), **activemq::commands::BrokerError** (p. 562), **activemq::commands::SessionInfo** (p. 2389), **activemq::commands::ProducerAck** (p. 2174), **activemq::commands::RemoveInfo** (p. 2270), **activemq::commands::ConnectionError** (p. 950), **activemq::commands::Response** (p. 2301), **activemq::commands::ReplayCommand** (p. 2286), **activemq::commands::KeepAliveInfo** (p. 1594), **activemq::commands::ShutdownInfo** (p. 2434), **activemq::commands::ControlCommand** (p. 1025), **activemq::commands::FlushCommand** (p. 1383), and **activemq::commands::WireFormatInfo** (p. 2899).

The documentation for this class was generated from the following file:

- src/main/activemq/commands/**Command.h**

6.131 activemq::state::CommandVisitor Class Reference

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

```
#include <src/main/activemq/state/CommandVisitor.h>
```

Inheritance diagram for activemq::state::CommandVisitor:

Public Member Functions

- virtual **~CommandVisitor** ()
- virtual **decaf::lang::Pointer** < **commands::Command** > **processTransactionInfo** (**commands::TransactionInfo** *info)=0
- virtual **decaf::lang::Pointer** < **commands::Command** > **processRemoveInfo** (**commands::RemoveInfo** *info)=0
- virtual **decaf::lang::Pointer** < **commands::Command** > **processConnectionInfo** (**commands::ConnectionInfo** *info)=0
- virtual **decaf::lang::Pointer** < **commands::Command** > **processSessionInfo** (**commands::SessionInfo** *info)=0
- virtual **decaf::lang::Pointer** < **commands::Command** > **processProducerInfo** (**commands::ProducerInfo** *info)=0
- virtual **decaf::lang::Pointer** < **commands::Command** > **processConsumerInfo** (**commands::ConsumerInfo** *info)=0
- virtual **decaf::lang::Pointer** < **commands::Command** > **processRemoveConnection** (**commands::ConnectionId** *id)=0
- virtual **decaf::lang::Pointer** < **commands::Command** > **processRemoveSession** (**commands::SessionId** *id)=0
- virtual **decaf::lang::Pointer** < **commands::Command** > **processRemoveProducer** (**commands::ProducerId** *id)=0
- virtual **decaf::lang::Pointer** < **commands::Command** > **processRemoveConsumer** (**commands::ConsumerId** *id)=0
- virtual **decaf::lang::Pointer** < **commands::Command** > **processDestinationInfo** (**commands::DestinationInfo** *info)=0
- virtual **decaf::lang::Pointer** < **commands::Command** > **processRemoveDestination** (**commands::DestinationInfo** *info)=0
- virtual **decaf::lang::Pointer** < **commands::Command** > **processRemoveSubscriptionInfo** (**commands::RemoveSubscriptionInfo** *info)=0
- virtual **decaf::lang::Pointer** < **commands::Command** > **processMessage** (**commands::Message** *send)=0
- virtual **decaf::lang::Pointer** < **commands::Command** > **processMessageAck** (**commands::MessageAck** *ack)=0
- virtual **decaf::lang::Pointer** < **commands::Command** > **processMessagePull** (**commands::MessagePull** *pull)=0
- virtual **decaf::lang::Pointer** < **commands::Command** > **processBeginTransaction** (**commands::TransactionInfo** *info)=0

- virtual **decaf::lang::Pointer** < **commands::Command** > **processPrepareTransaction** (**commands::TransactionInfo** *info)=0
- virtual **decaf::lang::Pointer** < **commands::Command** > **processCommitTransactionOnePhase** (**commands::TransactionInfo** *info)=0
- virtual **decaf::lang::Pointer** < **commands::Command** > **processCommitTransactionTwoPhase** (**commands::TransactionInfo** *info)=0
- virtual **decaf::lang::Pointer** < **commands::Command** > **processRollbackTransaction** (**commands::TransactionInfo** *info)=0
- virtual **decaf::lang::Pointer** < **commands::Command** > **processWireFormat** (**commands::WireFormatInfo** *info)=0
- virtual **decaf::lang::Pointer** < **commands::Command** > **processKeepAliveInfo** (**commands::KeepAliveInfo** *info)=0
- virtual **decaf::lang::Pointer** < **commands::Command** > **processShutdownInfo** (**commands::ShutdownInfo** *info)=0
- virtual **decaf::lang::Pointer** < **commands::Command** > **processFlushCommand** (**commands::FlushCommand** *command)=0
- virtual **decaf::lang::Pointer** < **commands::Command** > **processBrokerInfo** (**commands::BrokerInfo** *info)=0
- virtual **decaf::lang::Pointer** < **commands::Command** > **processRecoverTransactions** (**commands::TransactionInfo** *info)=0
- virtual **decaf::lang::Pointer** < **commands::Command** > **processForgetTransaction** (**commands::TransactionInfo** *info)=0
- virtual **decaf::lang::Pointer** < **commands::Command** > **processEndTransaction** (**commands::TransactionInfo** *info)=0
- virtual **decaf::lang::Pointer** < **commands::Command** > **processMessageDispatchNotification** (**commands::MessageDispatchNotification** *notification)=0
- virtual **decaf::lang::Pointer** < **commands::Command** > **processProducerAck** (**commands::ProducerAck** *ack)=0
- virtual **decaf::lang::Pointer** < **commands::Command** > **processMessageDispatch** (**commands::MessageDispatch** *dispatch)=0
- virtual **decaf::lang::Pointer** < **commands::Command** > **processControlCommand** (**commands::ControlCommand** *command)=0
- virtual **decaf::lang::Pointer** < **commands::Command** > **processConnectionError** (**commands::ConnectionError** *error)=0
- virtual **decaf::lang::Pointer** < **commands::Command** > **processConnectionControl** (**commands::ConnectionControl** *control)=0
- virtual **decaf::lang::Pointer** < **commands::Command** > **processConsumerControl** (**commands::ConsumerControl** *control)=0
- virtual **decaf::lang::Pointer** < **commands::Command** > **processBrokerError** (**commands::BrokerError** *error)=0
- virtual **decaf::lang::Pointer** < **commands::Command** > **processReplayCommand** (**commands::ReplayCommand** *replay)=0
- virtual **decaf::lang::Pointer** < **commands::Command** > **processResponse** (**commands::Response** *response)=0

6.131.1 Detailed Description

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

The Commands themselves implement a `visit` method that is called with an instance of this interface and each one then call the appropriate `processXXX` method.

Since

3.0

6.131.2 Constructor & Destructor Documentation

6.131.2.1 `virtual activemq::state::CommandVisitor::~~CommandVisitor ()`
[inline, virtual]

6.131.3 Member Function Documentation

6.131.3.1 `virtual decaf::lang::Pointer<commands::Command>`
`activemq::state::CommandVisitor::processBeginTransaction (`
`commands::TransactionInfo * info)` [pure virtual]

Implemented in `activemq::state::ConnectionStateTracker` (p. 987).

6.131.3.2 `virtual decaf::lang::Pointer<commands::Command>`
`activemq::state::CommandVisitor::processBrokerError (`
`commands::BrokerError * error)` [pure virtual]

6.131.3.3 `virtual decaf::lang::Pointer<commands::Command>`
`activemq::state::CommandVisitor::processBrokerInfo (`
`commands::BrokerInfo * info)` [pure virtual]

6.131.3.4 `virtual decaf::lang::Pointer<commands::Command> activemq-`
`::state::CommandVisitor::processCommitTransactionOnePhase (`
`commands::TransactionInfo * info)` [pure virtual]

Implemented in `activemq::state::ConnectionStateTracker` (p. 987).

6.131.3.5 `virtual decaf::lang::Pointer<commands::Command> activemq-`
`::state::CommandVisitor::processCommitTransactionTwoPhase (`
`commands::TransactionInfo * info)` [pure virtual]

Implemented in `activemq::state::ConnectionStateTracker` (p. 987).

6.131.3.6 virtual decaf::lang::Pointer<commands::Command>
activemq::state::CommandVisitor::processConnectionControl (
commands::ConnectionControl * *control*) [pure virtual]

6.131.3.7 virtual decaf::lang::Pointer<commands::Command>
activemq::state::CommandVisitor::processConnectionError (
commands::ConnectionError * *error*) [pure virtual]

6.131.3.8 virtual decaf::lang::Pointer<commands::Command>
activemq::state::CommandVisitor::processConnectionInfo (
commands::ConnectionInfo * *info*) [pure virtual]

Implemented in **activemq::state::ConnectionStateTracker** (p. 987).

6.131.3.9 virtual decaf::lang::Pointer<commands::Command>
activemq::state::CommandVisitor::processConsumerControl (
commands::ConsumerControl * *control*) [pure virtual]

6.131.3.10 virtual decaf::lang::Pointer<commands::Command>
activemq::state::CommandVisitor::processConsumerInfo (
commands::ConsumerInfo * *info*) [pure virtual]

Implemented in **activemq::state::ConnectionStateTracker** (p. 987).

6.131.3.11 virtual decaf::lang::Pointer<commands::Command>
activemq::state::CommandVisitor::processControlCommand (
commands::ControlCommand * *command*) [pure virtual]

6.131.3.12 virtual decaf::lang::Pointer<commands::Command>
activemq::state::CommandVisitor::processDestinationInfo (
commands::DestinationInfo * *info*) [pure virtual]

Implemented in **activemq::state::ConnectionStateTracker** (p. 987).

6.131.3.13 virtual decaf::lang::Pointer<commands::Command>
activemq::state::CommandVisitor::processEndTransaction (
commands::TransactionInfo * *info*) [pure virtual]

Implemented in **activemq::state::ConnectionStateTracker** (p. 987).

6.131.3.14 virtual decaf::lang::Pointer<commands::Command>
activemq::state::CommandVisitor::processFlushCommand (
commands::FlushCommand * *command*) [pure virtual]

6.131.3.15 virtual decaf::lang::Pointer<commands::Command>
 activemq::state::CommandVisitor::processForgetTransaction (
 commands::TransactionInfo * *info*) [pure virtual]

6.131.3.16 virtual decaf::lang::Pointer<commands::Command>
 activemq::state::CommandVisitor::processKeepAliveInfo (
 commands::KeepAliveInfo * *info*) [pure virtual]

6.131.3.17 virtual decaf::lang::Pointer<commands::Command>
 activemq::state::CommandVisitor::processMessage (
 commands::Message * *send*) [pure virtual]

Implemented in **activemq::state::ConnectionStateTracker** (p. 988).

6.131.3.18 virtual decaf::lang::Pointer<commands::Command>
 activemq::state::CommandVisitor::processMessageAck (
 commands::MessageAck * *ack*) [pure virtual]

Implemented in **activemq::state::ConnectionStateTracker** (p. 988).

6.131.3.19 virtual decaf::lang::Pointer<commands::Command>
 activemq::state::CommandVisitor::processMessageDispatch (
 commands::MessageDispatch * *dispatch*) [pure virtual]

6.131.3.20 virtual decaf::lang::Pointer<commands::Command> activemq-
 ::state::CommandVisitor::processMessageDispatchNotification
 (commands::MessageDispatchNotification * *notification*) [pure
 virtual]

6.131.3.21 virtual decaf::lang::Pointer<commands::Command>
 activemq::state::CommandVisitor::processMessagePull (
 commands::MessagePull * *pull*) [pure virtual]

6.131.3.22 virtual decaf::lang::Pointer<commands::Command>
 activemq::state::CommandVisitor::processPrepareTransaction (
 commands::TransactionInfo * *info*) [pure virtual]

Implemented in **activemq::state::ConnectionStateTracker** (p. 988).

6.131.3.23 virtual decaf::lang::Pointer<commands::Command>
 activemq::state::CommandVisitor::processProducerAck (
 commands::ProducerAck * *ack*) [pure virtual]

6.131.3.24 `virtual decaf::lang::Pointer<commands::Command>
activemq::state::CommandVisitor::processProducerInfo (
commands::ProducerInfo * info) [pure virtual]`

Implemented in **activemq::state::ConnectionStateTracker** (p. 988).

6.131.3.25 `virtual decaf::lang::Pointer<commands::Command>
activemq::state::CommandVisitor::processRecoverTransactions (
commands::TransactionInfo * info) [pure virtual]`

6.131.3.26 `virtual decaf::lang::Pointer<commands::Command>
activemq::state::CommandVisitor::processRemoveConnection (
commands::ConnectionId * id) [pure virtual]`

Implemented in **activemq::state::ConnectionStateTracker** (p. 988).

6.131.3.27 `virtual decaf::lang::Pointer<commands::Command>
activemq::state::CommandVisitor::processRemoveConsumer (
commands::ConsumerId * id) [pure virtual]`

Implemented in **activemq::state::ConnectionStateTracker** (p. 988).

6.131.3.28 `virtual decaf::lang::Pointer<commands::Command>
activemq::state::CommandVisitor::processRemoveDestination (
commands::DestinationInfo * info) [pure virtual]`

Implemented in **activemq::state::ConnectionStateTracker** (p. 988).

6.131.3.29 `virtual decaf::lang::Pointer<commands::Command>
activemq::state::CommandVisitor::processRemoveInfo (
commands::RemoveInfo * info) [pure virtual]`

Implemented in **activemq::state::CommandVisitorAdapter** (p. 884).

6.131.3.30 `virtual decaf::lang::Pointer<commands::Command>
activemq::state::CommandVisitor::processRemoveProducer (
commands::ProducerId * id) [pure virtual]`

Implemented in **activemq::state::ConnectionStateTracker** (p. 989).

6.131.3.31 `virtual decaf::lang::Pointer<commands::Command>
activemq::state::CommandVisitor::processRemoveSession (
commands::SessionId * id) [pure virtual]`

Implemented in **activemq::state::ConnectionStateTracker** (p. 989).

6.131.3.32 `virtual decaf::lang::Pointer<commands::Command>
activemq::state::CommandVisitor::processRemoveSubscriptionInfo (
commands::RemoveSubscriptionInfo * info) [pure virtual]`

6.131.3.33 `virtual decaf::lang::Pointer<commands::Command>
activemq::state::CommandVisitor::processReplayCommand (
commands::ReplayCommand * replay) [pure virtual]`

6.131.3.34 `virtual decaf::lang::Pointer<commands::Command>
activemq::state::CommandVisitor::processResponse (
commands::Response * response) [pure virtual]`

6.131.3.35 `virtual decaf::lang::Pointer<commands::Command>
activemq::state::CommandVisitor::processRollbackTransaction (
commands::TransactionInfo * info) [pure virtual]`

Implemented in `activemq::state::ConnectionStateTracker` (p. 989).

6.131.3.36 `virtual decaf::lang::Pointer<commands::Command>
activemq::state::CommandVisitor::processSessionInfo (
commands::SessionInfo * info) [pure virtual]`

Implemented in `activemq::state::ConnectionStateTracker` (p. 989).

6.131.3.37 `virtual decaf::lang::Pointer<commands::Command>
activemq::state::CommandVisitor::processShutdownInfo (
commands::ShutdownInfo * info) [pure virtual]`

6.131.3.38 `virtual decaf::lang::Pointer<commands::Command>
activemq::state::CommandVisitor::processTransactionInfo (
commands::TransactionInfo * info) [pure virtual]`

Implemented in `activemq::state::CommandVisitorAdapter` (p. 885).

6.131.3.39 `virtual decaf::lang::Pointer<commands::Command>
activemq::state::CommandVisitor::processWireFormat (
commands::WireFormatInfo * info) [pure virtual]`

The documentation for this class was generated from the following file:

- `src/main/activemq/state/CommandVisitor.h`

6.132 `activemq::state::CommandVisitorAdapter` Class Reference

Default Implementation of a `CommandVisitor` (p. 872) that returns NULL for all calls.

```
#include <src/main/activemq/state/CommandVisitorAdapter.h>
```

Inheritance diagram for activemq::state::CommandVisitorAdapter:

Public Member Functions

- virtual **~CommandVisitorAdapter** ()
- virtual **decaf::lang::Pointer** < **commands::Command** > **processRemoveConnection** (**commands::ConnectionId** *id AMQCPP_UNUSED)
- virtual **decaf::lang::Pointer** < **commands::Command** > **processRemoveSession** (**commands::SessionId** *id AMQCPP_UNUSED)
- virtual **decaf::lang::Pointer** < **commands::Command** > **processRemoveProducer** (**commands::ProducerId** *id AMQCPP_UNUSED)
- virtual **decaf::lang::Pointer** < **commands::Command** > **processRemoveConsumer** (**commands::ConsumerId** *id AMQCPP_UNUSED)
- virtual **decaf::lang::Pointer** < **commands::Command** > **processDestinationInfo** (**commands::DestinationInfo** *info AMQCPP_UNUSED)
- virtual **decaf::lang::Pointer** < **commands::Command** > **processRemoveDestination** (**commands::DestinationInfo** *info AMQCPP_UNUSED)
- virtual **decaf::lang::Pointer** < **commands::Command** > **processRemoveSubscriptionInfo** (**commands::RemoveSubscriptionInfo** *info AMQCPP_UNUSED)
- virtual **decaf::lang::Pointer** < **commands::Command** > **processMessage** (**commands::Message** *send AMQCPP_UNUSED)
- virtual **decaf::lang::Pointer** < **commands::Command** > **processMessageAck** (**commands::MessageAck** *ack AMQCPP_UNUSED)
- virtual **decaf::lang::Pointer** < **commands::Command** > **processMessagePull** (**commands::MessagePull** *pull AMQCPP_UNUSED)
- virtual **decaf::lang::Pointer** < **commands::Command** > **processBeginTransaction** (**commands::TransactionInfo** *info AMQCPP_UNUSED)
- virtual **decaf::lang::Pointer** < **commands::Command** > **processPrepareTransaction** (**commands::TransactionInfo** *info AMQCPP_UNUSED)
- virtual **decaf::lang::Pointer** < **commands::Command** > **processCommitTransactionOnePhase** (**commands::TransactionInfo** *info AMQCPP_UNUSED)
- virtual **decaf::lang::Pointer** < **commands::Command** > **processCommitTransactionTwoPhase** (**commands::TransactionInfo** *info AMQCPP_UNUSED)
- virtual **decaf::lang::Pointer** < **commands::Command** > **processRollbackTransaction** (**commands::TransactionInfo** *info AMQCPP_UNUSED)
- virtual **decaf::lang::Pointer** < **commands::Command** > **processWireFormat** (**commands::WireFormatInfo** *info AMQCPP_UNUSED)
- virtual **decaf::lang::Pointer** < **commands::Command** > **processKeepAliveInfo** (**commands::KeepAliveInfo** *info AMQCPP_UNUSED)

- virtual **decaf::lang::Pointer** < **commands::Command** > **processShutdownInfo** (**commands::ShutdownInfo** *info AMQCPP_UNUSED)
- virtual **decaf::lang::Pointer** < **commands::Command** > **processFlushCommand** (**commands::FlushCommand** *command AMQCPP_UNUSED)
- virtual **decaf::lang::Pointer** < **commands::Command** > **processBrokerInfo** (**commands::BrokerInfo** *info AMQCPP_UNUSED)
- virtual **decaf::lang::Pointer** < **commands::Command** > **processRecoverTransactions** (**commands::TransactionInfo** *info AMQCPP_UNUSED)
- virtual **decaf::lang::Pointer** < **commands::Command** > **processForgetTransaction** (**commands::TransactionInfo** *info AMQCPP_UNUSED)
- virtual **decaf::lang::Pointer** < **commands::Command** > **processEndTransaction** (**commands::TransactionInfo** *info AMQCPP_UNUSED)
- virtual **decaf::lang::Pointer** < **commands::Command** > **processMessageDispatchNotification** (**commands::MessageDispatchNotification** *notification AMQCPP_UNUSED)
- virtual **decaf::lang::Pointer** < **commands::Command** > **processProducerAck** (**commands::ProducerAck** *ack AMQCPP_UNUSED)
- virtual **decaf::lang::Pointer** < **commands::Command** > **processMessageDispatch** (**commands::MessageDispatch** *dispatch AMQCPP_UNUSED)
- virtual **decaf::lang::Pointer** < **commands::Command** > **processControlCommand** (**commands::ControlCommand** *command AMQCPP_UNUSED)
- virtual **decaf::lang::Pointer** < **commands::Command** > **processConnectionError** (**commands::ConnectionError** *error AMQCPP_UNUSED)
- virtual **decaf::lang::Pointer** < **commands::Command** > **processConnectionControl** (**commands::ConnectionControl** *control AMQCPP_UNUSED)
- virtual **decaf::lang::Pointer** < **commands::Command** > **processConsumerControl** (**commands::ConsumerControl** *control AMQCPP_UNUSED)
- virtual **decaf::lang::Pointer** < **commands::Command** > **processBrokerError** (**commands::BrokerError** *error AMQCPP_UNUSED)
- virtual **decaf::lang::Pointer** < **commands::Command** > **processReplayCommand** (**commands::ReplayCommand** *replay AMQCPP_UNUSED)
- virtual **decaf::lang::Pointer** < **commands::Command** > **processResponse** (**commands::Response** *response AMQCPP_UNUSED)
- virtual **decaf::lang::Pointer** < **commands::Command** > **processConnectionInfo** (**commands::ConnectionInfo** *info AMQCPP_UNUSED)
- virtual **decaf::lang::Pointer** < **commands::Command** > **processSessionInfo** (**commands::SessionInfo** *info AMQCPP_UNUSED)
- virtual **decaf::lang::Pointer** < **commands::Command** > **processProducerInfo** (**commands::ProducerInfo** *info AMQCPP_UNUSED)
- virtual **decaf::lang::Pointer** < **commands::Command** > **processConsumerInfo** (**commands::ConsumerInfo** *info AMQCPP_UNUSED)
- virtual **decaf::lang::Pointer** < **commands::Command** > **processTransactionInfo** (**commands::TransactionInfo** *info)
- virtual **decaf::lang::Pointer** < **commands::Command** > **processRemoveInfo** (**commands::RemoveInfo** *info)

6.132.1 Detailed Description

Default Implementation of a **CommandVisitor** (p. 872) that returns NULL for all calls.

Since

3.0

6.132.2 Constructor & Destructor Documentation

6.132.2.1 `virtual activemq::state::CommandVisitorAdapter-
::~~CommandVisitorAdapter () [inline,
virtual]`

6.132.3 Member Function Documentation

6.132.3.1 `virtual decaf::lang::Pointer<commands::Command>
activemq::state::CommandVisitorAdapter::processBeginTransaction (
commands::TransactionInfo *info AMQCPP_UNUSED) [inline,
virtual]`

6.132.3.2 `virtual decaf::lang::Pointer<commands::Command>
activemq::state::CommandVisitorAdapter::processBrokerError
(commands::BrokerError *error AMQCPP_UNUSED) [inline,
virtual]`

6.132.3.3 `virtual decaf::lang::Pointer<commands::Command>
activemq::state::CommandVisitorAdapter::processBrokerInfo (
commands::BrokerInfo *info AMQCPP_UNUSED) [inline, virtual]`

6.132.3.4 `virtual decaf::lang::Pointer<commands::Command> activemq::state-
::CommandVisitorAdapter::processCommitTransactionOnePhase (
commands::TransactionInfo *info AMQCPP_UNUSED) [inline,
virtual]`

6.132.3.5 `virtual decaf::lang::Pointer<commands::Command> activemq::state-
::CommandVisitorAdapter::processCommitTransactionTwoPhase (
commands::TransactionInfo *info AMQCPP_UNUSED) [inline,
virtual]`

6.132.3.6 `virtual decaf::lang::Pointer<commands::Command>
activemq::state::CommandVisitorAdapter::processConnectionControl (
commands::ConnectionControl *control AMQCPP_UNUSED) [inline,
virtual]`

- 6.132.3.7 `virtual decaf::lang::Pointer<commands::Command>
activemq::state::CommandVisitorAdapter::processConnectionError (
commands::ConnectionError *error AMQCPP_UNUSED) [inline,
virtual]`
- 6.132.3.8 `virtual decaf::lang::Pointer<commands::Command>
activemq::state::CommandVisitorAdapter::processConnectionInfo (
commands::ConnectionInfo *info AMQCPP_UNUSED) [inline,
virtual]`
- 6.132.3.9 `virtual decaf::lang::Pointer<commands::Command>
activemq::state::CommandVisitorAdapter::processConsumerControl (
commands::ConsumerControl *control AMQCPP_UNUSED) [inline,
virtual]`
- 6.132.3.10 `virtual decaf::lang::Pointer<commands::Command>
activemq::state::CommandVisitorAdapter::processConsumerInfo (
commands::ConsumerInfo *info AMQCPP_UNUSED) [inline,
virtual]`
- 6.132.3.11 `virtual decaf::lang::Pointer<commands::Command>
activemq::state::CommandVisitorAdapter::processControlCommand (
commands::ControlCommand *command AMQCPP_UNUSED)
[inline, virtual]`
- 6.132.3.12 `virtual decaf::lang::Pointer<commands::Command>
activemq::state::CommandVisitorAdapter::processDestinationInfo (
commands::DestinationInfo *info AMQCPP_UNUSED) [inline,
virtual]`
- 6.132.3.13 `virtual decaf::lang::Pointer<commands::Command>
activemq::state::CommandVisitorAdapter::processEndTransaction (
commands::TransactionInfo *info AMQCPP_UNUSED) [inline,
virtual]`
- 6.132.3.14 `virtual decaf::lang::Pointer<commands::Command>
activemq::state::CommandVisitorAdapter::processFlushCommand (
commands::FlushCommand *command AMQCPP_UNUSED) [inline,
virtual]`
- 6.132.3.15 `virtual decaf::lang::Pointer<commands::Command>
activemq::state::CommandVisitorAdapter::processForgetTransaction (
commands::TransactionInfo *info AMQCPP_UNUSED) [inline,
virtual]`

- 6.132.3.16 `virtual decaf::lang::Pointer<commands::Command>
activemq::state::CommandVisitorAdapter::processKeepAliveInfo (
commands::KeepAliveInfo *info AMQCPP_UNUSED) [inline,
virtual]`
- 6.132.3.17 `virtual decaf::lang::Pointer<commands::Command>
activemq::state::CommandVisitorAdapter::processMessage (
commands::Message *send AMQCPP_UNUSED) [inline, virtual]`
- 6.132.3.18 `virtual decaf::lang::Pointer<commands::Command>
activemq::state::CommandVisitorAdapter::processMessageAck
(commands::MessageAck *ack AMQCPP_UNUSED) [inline,
virtual]`
- 6.132.3.19 `virtual decaf::lang::Pointer<commands::Command>
activemq::state::CommandVisitorAdapter::processMessageDispatch (
commands::MessageDispatch *dispatch AMQCPP_UNUSED) [inline,
virtual]`
- 6.132.3.20 `virtual decaf::lang::Pointer<commands::Command> activemq::state-
::CommandVisitorAdapter::processMessageDispatchNotification (
commands::MessageDispatchNotification *notification AMQCPP_UNUSED)
[inline, virtual]`
- 6.132.3.21 `virtual decaf::lang::Pointer<commands::Command>
activemq::state::CommandVisitorAdapter::processMessagePull
(commands::MessagePull *pull AMQCPP_UNUSED) [inline,
virtual]`
- 6.132.3.22 `virtual decaf::lang::Pointer<commands::Command> activemq-
::state::CommandVisitorAdapter::processPrepareTransaction (
commands::TransactionInfo *info AMQCPP_UNUSED) [inline,
virtual]`
- 6.132.3.23 `virtual decaf::lang::Pointer<commands::Command>
activemq::state::CommandVisitorAdapter::processProducerAck
(commands::ProducerAck *ack AMQCPP_UNUSED) [inline,
virtual]`
- 6.132.3.24 `virtual decaf::lang::Pointer<commands::Command>
activemq::state::CommandVisitorAdapter::processProducerInfo
(commands::ProducerInfo *info AMQCPP_UNUSED) [inline,
virtual]`
- 6.132.3.25 `virtual decaf::lang::Pointer<commands::Command> activemq-
::state::CommandVisitorAdapter::processRecoverTransactions
(commands::TransactionInfo *info AMQCPP_UNUSED) [inline,
virtual]`

- 6.132.3.26 `virtual decaf::lang::Pointer<commands::Command> activemq::state::CommandVisitorAdapter::processRemoveConnection (commands::ConnectionId *id AMQCPP_UNUSED) [inline, virtual]`
- 6.132.3.27 `virtual decaf::lang::Pointer<commands::Command> activemq::state::CommandVisitorAdapter::processRemoveConsumer (commands::ConsumerId *id AMQCPP_UNUSED) [inline, virtual]`
- 6.132.3.28 `virtual decaf::lang::Pointer<commands::Command> activemq::state::CommandVisitorAdapter::processRemoveDestination (commands::DestinationInfo *info AMQCPP_UNUSED) [inline, virtual]`
- 6.132.3.29 `virtual decaf::lang::Pointer<commands::Command> activemq::state::CommandVisitorAdapter::processRemoveInfo (commands::RemoveInfo *info) [inline, virtual]`

Implements **activemq::state::CommandVisitor** (p. 877).

References `activemq::commands::RemoveInfo::getObjectId()`, `activemq::commands::ConnectionId::ID_CONNECTIONID`, `activemq::commands::ConsumerId::ID_CONSUMERID`, `activemq::commands::ProducerId::ID_PRODUCERID`, and `activemq::commands::SessionId::ID_SESSIONID`.

- 6.132.3.30 `virtual decaf::lang::Pointer<commands::Command> activemq::state::CommandVisitorAdapter::processRemoveProducer (commands::ProducerId *id AMQCPP_UNUSED) [inline, virtual]`
- 6.132.3.31 `virtual decaf::lang::Pointer<commands::Command> activemq::state::CommandVisitorAdapter::processRemoveSession (commands::SessionId *id AMQCPP_UNUSED) [inline, virtual]`
- 6.132.3.32 `virtual decaf::lang::Pointer<commands::Command> activemq::state::CommandVisitorAdapter::processRemoveSubscriptionInfo (commands::RemoveSubscriptionInfo *info AMQCPP_UNUSED) [inline, virtual]`
- 6.132.3.33 `virtual decaf::lang::Pointer<commands::Command> activemq::state::CommandVisitorAdapter::processReplayCommand (commands::ReplayCommand *replay AMQCPP_UNUSED) [inline, virtual]`
- 6.132.3.34 `virtual decaf::lang::Pointer<commands::Command> activemq::state::CommandVisitorAdapter::processResponse (commands::Response *response AMQCPP_UNUSED) [inline, virtual]`

- 6.132.3.35 `virtual decaf::lang::Pointer<commands::Command> activemq-
::state::CommandVisitorAdapter::processRollbackTransaction
(commands::TransactionInfo *info AMQCPP_UNUSED) [inline,
virtual]`
- 6.132.3.36 `virtual decaf::lang::Pointer<commands::Command>
activemq::state::CommandVisitorAdapter::processSessionInfo
(commands::SessionInfo *info AMQCPP_UNUSED) [inline,
virtual]`
- 6.132.3.37 `virtual decaf::lang::Pointer<commands::Command>
activemq::state::CommandVisitorAdapter::processShutdownInfo (
commands::ShutdownInfo *info AMQCPP_UNUSED) [inline,
virtual]`
- 6.132.3.38 `virtual decaf::lang::Pointer<commands::Command>
activemq::state::CommandVisitorAdapter::processTransactionInfo (
commands::TransactionInfo * info) [inline, virtual]`

Implements **activemq::state::CommandVisitor** (p. 878).

References `activemq::commands::TransactionInfo::getType()`, `activemq::core::ActiveMQConstants::TRANSACTION_STATE_BEGIN`, `activemq::core::ActiveMQConstants::TRANSACTION_STATE_COMMITONEPHASE`, `activemq::core::ActiveMQConstants::TRANSACTION_STATE_COMMITTWOPHASE`, `activemq::core::ActiveMQConstants::TRANSACTION_STATE_END`, `activemq::core::ActiveMQConstants::TRANSACTION_STATE_FORGET`, `activemq::core::ActiveMQConstants::TRANSACTION_STATE_PREPARE`, `activemq::core::ActiveMQConstants::TRANSACTION_STATE_RECOVER`, and `activemq::core::ActiveMQConstants::TRANSACTION_STATE_ROLLBACK`.

- 6.132.3.39 `virtual decaf::lang::Pointer<commands::Command>
activemq::state::CommandVisitorAdapter::processWireFormat
(commands::WireFormatInfo *info AMQCPP_UNUSED) [inline,
virtual]`

The documentation for this class was generated from the following file:

- `src/main/activemq/state/CommandVisitorAdapter.h`

6.133 decaf::lang::Comparable< T > Class Template Reference

This interface imposes a total ordering on the objects of each class that implements it.

```
#include <src/main/decaf/lang/Comparable.h>
```

Public Member Functions

- virtual `~Comparable()`
- virtual `int compareTo (const T &value) const =0`
Compares this object with the specified object for order.
- virtual `bool equals (const T &value) const =0`
- virtual `bool operator== (const T &value) const =0`
Compares equality between this object and the one passed.
- virtual `bool operator< (const T &value) const =0`
Compares this object to another and returns true if this object is considered to be less than the one passed.

6.133.1 Detailed Description

```
template<typename T>class decaf::lang::Comparable< T >
```

This interface imposes a total ordering on the objects of each class that implements it.

This ordering is referred to as the class's natural ordering, and the class's `compareTo` method is referred to as its natural comparison method.

6.133.2 Constructor & Destructor Documentation

```
6.133.2.1 template<typename T> virtual decaf::lang::Comparable< T
>::~~Comparable( ) [inline, virtual]
```

6.133.3 Member Function Documentation

```
6.133.3.1 template<typename T> virtual int decaf::lang::Comparable< T
>::compareTo ( const T & value ) const [pure virtual]
```

Compares this object with the specified object for order.

Returns a negative integer, zero, or a positive integer as this object is less than, equal to, or greater than the specified object.

In the foregoing description, the notation `sgn(expression)` designates the mathematical signum function, which is defined to return one of -1, 0, or 1 according to whether the value of expression is negative, zero or positive. The implementor must ensure `sgn(x.compareTo(y)) == -sgn(y.compareTo(x))` for all x and y. (This implies that `x.compareTo(y)` must throw an exception iff `y.compareTo(x)` throws an exception.)

The implementor must also ensure that the relation is transitive: `(x.compareTo(y)>0 && y.compareTo(z)>0)` implies `x.compareTo(z)>0`.

Finally, the implementer must ensure that `x.compareTo(y)==0` implies that `sgn(x.compareTo(z)) == sgn(y.compareTo(z))`, for all z.

It is strongly recommended, but not strictly required that `(x.compareTo(y)==0) == (x.equals(y))`. Generally speaking, any class that implements the **Comparable** (p. 885)

interface and violates this condition should clearly indicate this fact. The recommended language is "Note: this class has a natural ordering that is inconsistent with equals."

Parameters

<i>value</i>	- the Object to be compared.
--------------	------------------------------

Returns

a negative integer, zero, or a positive integer as this object is less than, equal to, or greater than the specified object.

Implemented in **decaf::lang::Double** (p. 1239), **decaf::lang::Float** (p. 1348), **decaf::lang::Integer** (p. 1504), **decaf::lang::Long** (p. 1730), **decaf::lang::Boolean** (p. 547), **decaf::lang::Short** (p. 2401), **decaf::lang::Byte** (p. 617), **decaf::lang::Character** (p. 767), **decaf::lang::Double** (p. 1239), **decaf::lang::Float** (p. 1348), **decaf::lang::Boolean** (p. 547), **decaf::lang::Integer** (p. 1503), **decaf::lang::Long** (p. 1729), **decaf::lang::Byte** (p. 616), **decaf::lang::Short** (p. 2401), and **decaf::lang::Character** (p. 767).

6.133.3.2 `template<typename T> virtual bool decaf::lang::Comparable< T >::equals (const T & value) const [pure virtual]`

Returns

true if this value is considered equal to the passed value.

Implemented in **decaf::lang::Byte** (p. 618), **decaf::lang::Boolean** (p. 548), **decaf::lang::Character** (p. 768), **decaf::lang::Byte** (p. 618), **decaf::lang::Double** (p. 1241), **decaf::lang::Float** (p. 1349), **decaf::lang::Character** (p. 768), **decaf::lang::Integer** (p. 1505), **decaf::lang::Long** (p. 1731), **decaf::lang::Short** (p. 2402), **decaf::lang::Boolean** (p. 547), **decaf::lang::Double** (p. 1241), **decaf::lang::Float** (p. 1349), **decaf::lang::Integer** (p. 1505), **decaf::lang::Long** (p. 1731), and **decaf::lang::Short** (p. 2402).

6.133.3.3 `template<typename T> virtual bool decaf::lang::Comparable< T >::operator< (const T & value) const [pure virtual]`

Compares this object to another and returns true if this object is considered to be less than the one passed.

This

Parameters

<i>value</i>	- the value to be compared to this one.
--------------	---

Returns

true if this object is equal to the one passed.

Implemented in **decaf::lang::Double** (p. 1244), **decaf::lang::Float** (p. 1352), **decaf::lang::Integer** (p. 1508), **decaf::lang::Long** (p. 1734), **decaf::lang::Short** (p. 2404), **decaf::lang::Boolean** (p. 548), **decaf::lang::Byte** (p. 619), **decaf::lang::Character** (p. 771), **decaf::lang::Double** (p. 1243), **decaf::lang::Float** (p. 1352), **decaf::lang::Integer** (p. 1507), **decaf::lang::Long** (p. 1734), **decaf::lang::Short** (p. 2403), **decaf::lang::Boolean** (p. 548), **decaf::lang::Byte** (p. 619), and **decaf::lang::Character** (p. 770).

```
6.133.3.4  template<typename T> virtual bool decaf::lang::Comparable< T >::operator==
           ( const T & value ) const    [pure virtual]
```

Compares equality between this object and the one passed.

Parameters

<i>value</i>	- the value to be compared to this one.
--------------	---

Returns

true if this object is equal to the one passed.

Implemented in **decaf::lang::Double** (p. 1244), **decaf::lang::Float** (p. 1353), **decaf::lang::Integer** (p. 1508), **decaf::lang::Long** (p. 1735), **decaf::lang::Short** (p. 2404), **decaf::lang::Boolean** (p. 549), **decaf::lang::Byte** (p. 620), **decaf::lang::Character** (p. 771), **decaf::lang::Double** (p. 1244), **decaf::lang::Float** (p. 1353), **decaf::lang::Integer** (p. 1508), **decaf::lang::Long** (p. 1734), **decaf::lang::Boolean** (p. 548), **decaf::lang::Short** (p. 2404), **decaf::lang::Byte** (p. 620), and **decaf::lang::Character** (p. 771).

The documentation for this class was generated from the following file:

- `src/main/decaf/lang/Comparable.h`

6.134 decaf::util::Comparator< T > Class Template Reference

A comparison function, which imposes a total ordering on some collection of objects.

```
#include <src/main/decaf/util/Comparator.h>
```

Public Member Functions

- virtual **~Comparator** ()
- virtual bool **operator()** (const T &left, const T &right) const =0

*Implementation of the Binary function interface as a means of allowing a **Comparator** (p. 888) to be passed to an STL **Map** (p. 1768) for use as the sorting criteria.*

- virtual int **compare** (const T &o1, const T &o2) const =0

Compares its two arguments for order.

6.134.1 Detailed Description

```
template<typename T>class decaf::util::Comparator< T >
```

A comparison function, which imposes a total ordering on some collection of objects.

Comparators can be passed to a sort method (such as Collections.sort) to allow precise control over the sort order. Comparators can also be used to control the order of certain data structures.

The ordering imposed by a **Comparator** (p. 888) c on a set of elements S is said to be consistent with equals if and only if (compare(e1, e2) == 0) has the same boolean value as (e1 == e2) for every e1 and e2 in S.

Since

1.0

6.134.2 Constructor & Destructor Documentation

```
6.134.2.1 template<typename T> virtual decaf::util::Comparator< T >::~Comparator (
) [inline, virtual]
```

6.134.3 Member Function Documentation

```
6.134.3.1 template<typename T> virtual int decaf::util::Comparator< T >::compare (
const T & o1, const T & o2 ) const [pure virtual]
```

Compares its two arguments for order.

Returns a negative integer, zero, or a positive integer as the first argument is less than, equal to, or greater than the second.

The implementor must ensure that $\text{sgn}(\text{compare}(x, y)) == -\text{sgn}(\text{compare}(y, x))$ for all x and y. (This implies that compare(x, y) must throw an exception if and only if compare(y, x) throws an exception.)

The implementor must also ensure that the relation is transitive: $((\text{compare}(x, y) > 0) \ \&\& \ (\text{compare}(y, z) > 0))$ implies $\text{compare}(x, z) > 0$.

Finally, the implementer must ensure that $\text{compare}(x, y) == 0$ implies that $\text{sgn}(\text{compare}(x, z)) == \text{sgn}(\text{compare}(y, z))$ for all z.

It is generally the case, but not strictly required that $(\text{compare}(x, y) == 0) == (x == y)$. Generally speaking, any comparator that violates this condition should clearly indicate this fact. The recommended language is "Note: this comparator imposes orderings that are inconsistent with equals."

Parameters

<i>o1</i>	The first object to be compared
<i>o2</i>	The second object to be compared

Returns

a negative integer, zero, or a positive integer as the first argument is less than, equal to, or greater than the second.

Implemented in **decaf::util::comparators::Less< E >** (p. 1613).

6.134.3.2 `template<typename T> virtual bool decaf::util::Comparator< T >::operator() (const T & left, const T & right) const` [pure virtual]

Implementation of the Binary function interface as a means of allowing a **Comparator** (p. 888) to be passed to an STL **Map** (p. 1768) for use as the sorting criteria.

Parameters

<i>left</i>	The Left hand side operand.
<i>right</i>	The Right hand side operand.

Returns

true if the vale of left is less than the value of right.

Implemented in **decaf::util::comparators::Less< E >** (p. 1614).

The documentation for this class was generated from the following file:

- `src/main/decaf/util/Comparator.h`

6.135 activemq::util::CompositeData Class Reference

Represents a Composite URI.

```
#include <src/main/activemq/util/CompositeData.h>
```

Public Member Functions

- **CompositeData** ()
- virtual **~CompositeData** ()
- **LinkedList< URI > &getComponents** ()
- const **LinkedList< URI > &getComponents** () const
- void **setComponents** (const **LinkedList< URI > &components**)
- std::string **getFragment** () const

- void **setFragment** (const std::string &fragment)
- const **Properties** & **getParameters** () const
- void **setParameters** (const **Properties** ¶meters)
- std::string **getScheme** () const
- void **setScheme** (const std::string &scheme)
- std::string **getPath** () const
- void **setPath** (const std::string &path)
- std::string **getHost** () const
- void **setHost** (const std::string &host)
- **URI toURI** () const

6.135.1 Detailed Description

Represents a Composite URI.

Since

3.0

6.135.2 Constructor & Destructor Documentation

6.135.2.1 **activemq::util::CompositeData::CompositeData** ()

6.135.2.2 **virtual activemq::util::CompositeData::~~CompositeData** ()
[virtual]

6.135.3 Member Function Documentation

6.135.3.1 **LinkedList<URI>& activemq::util::CompositeData::getComponents** ()
[inline]

6.135.3.2 **const LinkedList<URI>& activemq::util::CompositeData::getComponents** () const [inline]

6.135.3.3 **std::string activemq::util::CompositeData::getFragment** () const
[inline]

6.135.3.4 **std::string activemq::util::CompositeData::getHost** () const [inline]

6.135.3.5 **const Properties& activemq::util::CompositeData::getParameters** () const [inline]

6.135.3.6 **std::string activemq::util::CompositeData::getPath** () const [inline]

6.135.3.7 **std::string activemq::util::CompositeData::getScheme** () const
[inline]

- 6.135.3.8 `void activemq::util::CompositeData::setComponents (const LinkedList< URI > & components) [inline]`
- 6.135.3.9 `void activemq::util::CompositeData::setFragment (const std::string & fragment) [inline]`
- 6.135.3.10 `void activemq::util::CompositeData::setHost (const std::string & host) [inline]`
- 6.135.3.11 `void activemq::util::CompositeData::setParameters (const Properties & parameters) [inline]`
- 6.135.3.12 `void activemq::util::CompositeData::setPath (const std::string & path) [inline]`
- 6.135.3.13 `void activemq::util::CompositeData::setScheme (const std::string & scheme) [inline]`
- 6.135.3.14 `URI activemq::util::CompositeData::toURI () const`

Exceptions

<i>decaf::net::URI-SyntaxException</i> (p. 2856)
--

The documentation for this class was generated from the following file:

- `src/main/activemq/util/CompositeData.h`

6.136 `activemq::threads::CompositeTask` Class Reference

Represents a single task that can be part of a set of Tasks that are contained in a **CompositeTaskRunner** (p. 893).

```
#include <src/main/activemq/threads/CompositeTask.h>
```

Inheritance diagram for `activemq::threads::CompositeTask`:

Public Member Functions

- virtual `~CompositeTask ()`
- virtual bool **isPending** () const =0

*Indicates whether this task has any pending work that needs to be done, if not then it is skipped and the next **Task** (p. 2676) in the **CompositeTaskRunner** (p. 893)'s list of tasks is checked, if none of the tasks have any pending work to do, then the runner can go to sleep until it awakened by a call to *wakeup*.*

6.136.1 Detailed Description

Represents a single task that can be part of a set of Tasks that are contained in a **CompositeTaskRunner** (p. 893).

Since

3.0

6.136.2 Constructor & Destructor Documentation

6.136.2.1 virtual **activemq::threads::CompositeTask::~CompositeTask** ()
[inline, virtual]

6.136.3 Member Function Documentation

6.136.3.1 virtual bool **activemq::threads::CompositeTask::isPending** () const
[pure virtual]

Indicates whether this task has any pending work that needs to be done, if not then it is skipped and the next **Task** (p. 2676) in the **CompositeTaskRunner** (p. 893)'s list of tasks is checked, if none of the tasks have any pending work to do, then the runner can go to sleep until it awakened by a call to `wakeup`.

Since

3.0

Implemented in **activemq::transport::failover::FailoverTransport** (p. 1311), **activemq::transport::failover::BackupTransportPool** (p. 491), and **activemq::transport::failover::CloseTransportsTask** (p. 818).

The documentation for this class was generated from the following file:

- `src/main/activemq/threads/CompositeTask.h`

6.137 activemq::threads::CompositeTaskRunner Class Reference

A **Task** (p. 2676) Runner that can contain one or more CompositeTasks that are each checked for pending work and run if any is present in the order that the tasks were added.

```
#include <src/main/activemq/threads/CompositeTaskRunner.h>
```

Inheritance diagram for `activemq::threads::CompositeTaskRunner`:

Public Member Functions

- **CompositeTaskRunner** ()
- virtual **~CompositeTaskRunner** ()
- void **addTask** (**CompositeTask** *task)
*Adds a new **CompositeTask** (p. 892) to the Set of Tasks that this class manages.*
- void **removeTask** (**CompositeTask** *task)
*Removes a **CompositeTask** (p. 892) that was added previously.*
- virtual void **shutdown** (unsigned int timeout)
Shutdown after a timeout, does not guarantee that the task's iterate method has completed and the thread halted.
- virtual void **shutdown** ()
*Shutdown once the task has finished and the **TaskRunner** (p. 2677)'s thread has exited.*
- virtual void **wakeup** ()
*Signal the **TaskRunner** (p. 2677) to wakeup and execute another iteration cycle on the task, the **Task** (p. 2676) instance will be run until its iterate method has returned false indicating it is done.*

Protected Member Functions

- virtual void **run** ()
Run method - called by the Thread class in the context of the thread.
- virtual bool **iterate** ()
Perform one iteration of work, returns true if the task needs to run again to complete or false to indicate that the task is now complete.

6.137.1 Detailed Description

A **Task** (p. 2676) Runner that can contain one or more CompositeTasks that are each checked for pending work and run if any is present in the order that the tasks were added.

Since

3.0

6.137.2 Constructor & Destructor Documentation

6.137.2.1 **activemq::threads::CompositeTaskRunner::CompositeTaskRunner** ()

6.137.2.2 **virtual activemq::threads::CompositeTaskRunner::~~CompositeTaskRunner** () `[virtual]`

6.137.3 Member Function Documentation

6.137.3.1 void **activemq::threads::CompositeTaskRunner::addTask** (
CompositeTask * *task*)

Adds a new **CompositeTask** (p. 892) to the Set of Tasks that this class manages.

Parameters

<i>task</i>	- Pointer to a CompositeTask (p. 892) instance.
-------------	--

6.137.3.2 virtual bool **activemq::threads::CompositeTaskRunner::iterate** ()
[protected, virtual]

Perform one iteration of work, returns true if the task needs to run again to complete or false to indicate that the task is now complete.

Returns

true if the task should be run again or false if the task has completed and the runner should wait for a wakeup call.

Implements **activemq::threads::Task** (p. 2676).

6.137.3.3 void **activemq::threads::CompositeTaskRunner::removeTask** (
CompositeTask * *task*)

Removes a **CompositeTask** (p. 892) that was added previously.

Parameters

<i>task</i>	- Pointer to a CompositeTask (p. 892) instance.
-------------	--

6.137.3.4 virtual void **activemq::threads::CompositeTaskRunner::run** ()
[protected, virtual]

Run method - called by the Thread class in the context of the thread.

Implements **decaf::lang::Runnable** (p. 2312).

6.137.3.5 virtual void **activemq::threads::CompositeTaskRunner::shutdown** (
unsigned int *timeout*) [virtual]

Shutdown after a timeout, does not guarantee that the task's iterate method has completed and the thread halted.

Parameters

<i>timeout</i>	- Time in Milliseconds to wait for the task to stop.
----------------	--

Implements **activemq::threads::TaskRunner** (p. 2677).

6.137.3.6 virtual void **activemq::threads::CompositeTaskRunner::shutdown** ()
[virtual]

Shutdown once the task has finished and the **TaskRunner** (p. 2677)'s thread has exited.

Implements **activemq::threads::TaskRunner** (p. 2678).

6.137.3.7 virtual void **activemq::threads::CompositeTaskRunner::wakeup** ()
[virtual]

Signal the **TaskRunner** (p. 2677) to wakeup and execute another iteration cycle on the task, the **Task** (p. 2676) instance will be run until its iterate method has returned false indicating it is done.

Implements **activemq::threads::TaskRunner** (p. 2678).

The documentation for this class was generated from the following file:

- src/main/activemq/threads/**CompositeTaskRunner.h**

6.138 activemq::transport::CompositeTransport Class Reference

A Composite **Transport** (p. 2790) is a **Transport** (p. 2790) implementation that is composed of several Transports.

```
#include <src/main/activemq/transport/CompositeTransport.h>
```

Inheritance diagram for **activemq::transport::CompositeTransport**:

Public Member Functions

- virtual **~CompositeTransport** ()
- virtual void **addURI** (bool rebalance, const **List**< **URI** > &uris)=0
*Add a URI to the list of URI's that will represent the set of Transports that this **Transport** (p. 2790) is a composite of.*
- virtual void **removeURI** (bool rebalance, const **List**< **URI** > &uris)=0
*Remove a URI from the set of URI's that represents the set of Transports that this **Transport** (p. 2790) is composed of, removing a URI for which the composite has created a connected **Transport** (p. 2790) should result in that **Transport** (p. 2790) being disposed of.*

6.138.1 Detailed Description

A Composite **Transport** (p. 2790) is a **Transport** (p. 2790) implementation that is composed of several Transports.

The composition could be such that only one **Transport** (p. 2790) exists for each URI that is composed or there could be many active Transports working at once.

Since

3.0

6.138.2 Constructor & Destructor Documentation

6.138.2.1 virtual **activemq::transport::CompositeTransport::~~CompositeTransport** () [inline, virtual]

6.138.3 Member Function Documentation

6.138.3.1 virtual void **activemq::transport::CompositeTransport::addURI** (bool *rebalance*, const List< URI > & *uris*) [pure virtual]

Add a URI to the list of URI's that will represent the set of Transports that this **Transport** (p. 2790) is a composite of.

Parameters

<i>rebalance</i>	Indicates if the addition should cause a forced reconnect or not.
<i>uris</i>	The new URI set to add to the set this composite maintains.

Implemented in **activemq::transport::failover::FailoverTransport** (p. 1308).

6.138.3.2 virtual void **activemq::transport::CompositeTransport::removeURI** (bool *rebalance*, const List< URI > & *uris*) [pure virtual]

Remove a URI from the set of URI's that represents the set of Transports that this **Transport** (p. 2790) is composed of, removing a URI for which the composite has created a connected **Transport** (p. 2790) should result in that **Transport** (p. 2790) being disposed of.

Parameters

<i>rebalance</i>	Indicates if the removal should cause a forced reconnect or not.
<i>uris</i>	The new URI set to remove to the set this composite maintains.

Implemented in **activemq::transport::failover::FailoverTransport** (p. 1314).

The documentation for this class was generated from the following file:

- src/main/activemq/transport/**CompositeTransport.h**

6.139 decaf::util::concurrent::ConcurrentMap< K, V, COMPARATOR > Class Template Reference

Interface for a **Map** (p. 1768) type that provides additional atomic putIfAbsent, remove, and replace methods alongside the already available **Map** (p. 1768) interface.

```
#include <src/main/decaf/util/concurrent/ConcurrentMap.h>
```

Inheritance diagram for decaf::util::concurrent::ConcurrentMap< K, V, COMPARATOR >:

Public Member Functions

- virtual **~ConcurrentMap** ()
- virtual bool **putIfAbsent** (const K &key, const V &value)=0
If the specified key is not already associated with a value, associate it with the given value.
- virtual bool **remove** (const K &key, const V &value)=0
Remove entry for key only if currently mapped to given value.
- virtual bool **replace** (const K &key, const V &oldValue, const V &newValue)=0
Replace entry for key only if currently mapped to given value.
- virtual V **replace** (const K &key, const V &value)=0
Replace entry for key only if currently mapped to some value.

6.139.1 Detailed Description

```
template<typename K, typename V, typename COMPARATOR>class decaf::util::concurrent::ConcurrentMap< K, V, COMPARATOR >
```

Interface for a **Map** (p. 1768) type that provides additional atomic putIfAbsent, remove, and replace methods alongside the already available **Map** (p. 1768) interface.

Since

1.0

6.139.2 Constructor & Destructor Documentation

6.139.2.1 `template<typename K, typename V, typename COMPARATOR> virtual
 decaf::util::concurrent::ConcurrentMap< K, V, COMPARATOR
 >::~~ConcurrentMap() [inline, virtual]`

6.139.3 Member Function Documentation

6.139.3.1 `template<typename K, typename V, typename COMPARATOR> virtual
 bool decaf::util::concurrent::ConcurrentMap< K, V, COMPARATOR
 >::putIfAbsent (const K & key, const V & value) [pure virtual]`

If the specified key is not already associated with a value, associate it with the given value.

This is equivalent to

```
if( !map.containsKey( key ) ) {
    map.put( key, value );
    return true;
} else {
    return false;
}
```

except that the action is performed atomically.

Parameters

<i>key</i>	The key to map the value to.
<i>value</i>	The value to map to the given key.

Returns

true if the put operation was performed otherwise return false which indicates there was a value previously mapped to the key.

Exceptions

<i>Unsupported- OperationException</i>	if the put operation is not supported by this map
--	---

Implemented in `decaf::util::concurrent::ConcurrentStlMap< K, V, COMPARATOR >` (p.915), `decaf::util::concurrent::ConcurrentStlMap< Pointer< MessageId >, Pointer< Message >, MessageId::COMPARATOR >` (p.915), `decaf::util::concurrent::ConcurrentStlMap< Pointer< ConnectionId >, Pointer< ConnectionState >, ConnectionId::COMPARATOR >` (p.915), `decaf::util::concurrent::ConcurrentStlMap< Pointer< ConsumerId >, Pointer< ConsumerState >, ConsumerId::COMPARATOR >` (p.915), `decaf::util::concurrent::ConcurrentStlMap< Pointer< SessionId >, Pointer< SessionState >, SessionId::COMPARATOR >` (p.915), `decaf::util::concurrent::ConcurrentStlMap< Pointer< LocalTransactionId >, Pointer< TransactionState >, LocalTransaction-`

Id::COMPARATOR > (p.915), and **decaf::util::concurrent::ConcurrentStlMap< Pointer< ProducerId >, Pointer< ProducerState >, ProducerId::COMPARATOR >** (p.915).

6.139.3.2 `template<typename K, typename V, typename COMPARATOR> virtual bool
decaf::util::concurrent::ConcurrentMap< K, V, COMPARATOR >::remove (
const K & key, const V & value) [pure virtual]`

Remove entry for key only if currently mapped to given value.

Acts as

```
if( ( map.containsKey( key ) && ( map.get( key ) == value ) ) ) {  
    map.remove( key );  
    return true;  
} else {  
    return false;  
}
```

except that the action is performed atomically.

Parameters

<i>key</i>	key with which the specified value is associated.
<i>value</i>	value associated with the specified key.

Returns

true if the value was removed, false otherwise

Implemented in **decaf::util::concurrent::ConcurrentStlMap< K, V, COMPARATOR >** (p.917), **decaf::util::concurrent::ConcurrentStlMap< Pointer< MessageId >, Pointer< Message >, MessageId::COMPARATOR >** (p.917), **decaf::util::concurrent::ConcurrentStlMap< Pointer< ConnectionId >, Pointer< ConnectionState >, ConnectionId::COMPARATOR >** (p.917), **decaf::util::concurrent::ConcurrentStlMap< Pointer< ConsumerId >, Pointer< ConsumerState >, ConsumerId::COMPARATOR >** (p.917), **decaf::util::concurrent::ConcurrentStlMap< Pointer< SessionId >, Pointer< SessionState >, SessionId::COMPARATOR >** (p.917), **decaf::util::concurrent::ConcurrentStlMap< Pointer< LocalTransactionId >, Pointer< TransactionState >, LocalTransactionId::COMPARATOR >** (p.917), and **decaf::util::concurrent::ConcurrentStlMap< Pointer< ProducerId >, Pointer< ProducerState >, ProducerId::COMPARATOR >** (p.917).

6.139.3.3 `template<typename K, typename V, typename COMPARATOR> virtual bool
decaf::util::concurrent::ConcurrentMap< K, V, COMPARATOR >::replace (
const K & key, const V & oldValue, const V & newValue) [pure virtual]`

Replace entry for key only if currently mapped to given value.

Acts as

```
if( ( map.containsKey( key ) && ( map.get( key ) == oldValue ) ) ) {
    map.put( key, newValue );
    return true;
} else {
    return false;
}
```

except that the action is performed atomically.

Parameters

<i>key</i>	key with which the specified value is associated.
<i>oldValue</i>	value expected to be associated with the specified key.
<i>newValue</i>	value to be associated with the specified key.

Returns

true if the value was replaced

Implemented in **decaf::util::concurrent::ConcurrentStlMap< K, V, COMPARATOR >** (p.917), **decaf::util::concurrent::ConcurrentStlMap< Pointer< MessageId >, Pointer< Message >, MessageId::COMPARATOR >** (p.917), **decaf::util::concurrent::ConcurrentStlMap< Pointer< ConnectionId >, Pointer< ConnectionState >, ConnectionId::COMPARATOR >** (p.917), **decaf::util::concurrent::ConcurrentStlMap< Pointer< ConsumerId >, Pointer< ConsumerState >, ConsumerId::COMPARATOR >** (p.917), **decaf::util::concurrent::ConcurrentStlMap< Pointer< SessionId >, Pointer< SessionState >, SessionId::COMPARATOR >** (p.917), **decaf::util::concurrent::ConcurrentStlMap< Pointer< LocalTransactionId >, Pointer< TransactionState >, LocalTransactionId::COMPARATOR >** (p.917), and **decaf::util::concurrent::ConcurrentStlMap< Pointer< ProducerId >, Pointer< ProducerState >, ProducerId::COMPARATOR >** (p.917).

6.139.3.4 `template<typename K, typename V, typename COMPARATOR> virtual V
 decaf::util::concurrent::ConcurrentMap< K, V, COMPARATOR >::replace (
 const K & key, const V & value) [pure virtual]`

Replace entry for key only if currently mapped to some value.

Acts as

```
if( ( map.containsKey( key ) ) ) {
    return map.put( key, value );
} else {
    throw NoSuchElementException (p.1984) (...);
};
```

except that the action is performed atomically.

Parameters

<i>key</i>	key with which the specified value is associated.
<i>value</i>	value to be associated with the specified key.

Returns

copy of the previous value associated with specified key, or throws an **NoSuchElementException** (p. 1984) if there was no mapping for key.

Exceptions

<i>NoSuchElementException</i> (p. 1984)	if there was no previous mapping.
---	-----------------------------------

Implemented in **decaf::util::concurrent::ConcurrentStlMap< K, V, COMPARATOR >** (p. 918), **decaf::util::concurrent::ConcurrentStlMap< Pointer< MessageId >, Pointer< Message >, MessageId::COMPARATOR >** (p. 918), **decaf::util::concurrent::ConcurrentStlMap< Pointer< ConnectionId >, Pointer< ConnectionState >, ConnectionId::COMPARATOR >** (p. 918), **decaf::util::concurrent::ConcurrentStlMap< Pointer< ConsumerId >, Pointer< ConsumerState >, ConsumerId::COMPARATOR >** (p. 918), **decaf::util::concurrent::ConcurrentStlMap< Pointer< SessionId >, Pointer< SessionState >, SessionId::COMPARATOR >** (p. 918), **decaf::util::concurrent::ConcurrentStlMap< Pointer< LocalTransactionId >, Pointer< TransactionState >, LocalTransactionId::COMPARATOR >** (p. 918), and **decaf::util::concurrent::ConcurrentStlMap< Pointer< ProducerId >, Pointer< ProducerState >, ProducerId::COMPARATOR >** (p. 918).

The documentation for this class was generated from the following file:

- `src/main/decaf/util/concurrent/ConcurrentMap.h`

6.140 decaf::util::ConcurrentModificationException Class Reference

```
#include <src/main/decaf/util/ConcurrentModification-Exception.h>
```

Inheritance diagram for **decaf::util::ConcurrentModificationException**:

Public Member Functions

- **ConcurrentModificationException** () throw ()

Default Constructor.

- **ConcurrentModificationException** (const lang::Exception &ex) throw ()

Copy Constructor.

- **ConcurrentModificationException** (const ConcurrentModificationException &ex) throw ()

Copy Constructor.

- **ConcurrentModificationException** (const char *file, const int lineNumber, const std::exception *cause, const char *msg,...) throw ()

Constructor - Initializes the file name and line number where this message occurred.

- **ConcurrentModificationException** (const std::exception *cause) throw ()

Constructor.

- **ConcurrentModificationException** (const char *file, const int lineNumber, const char *msg,...) throw ()

Constructor.

- virtual **ConcurrentModificationException** * clone () const

Clones this exception.

- virtual ~**ConcurrentModificationException** () throw ()

6.140.1 Constructor & Destructor Documentation

6.140.1.1 decaf::util::ConcurrentModificationException::ConcurrentModificationException () throw ()

Default Constructor.

6.140.1.2 decaf::util::ConcurrentModificationException::ConcurrentModificationException (const lang::Exception & ex) throw () [inline]

Copy Constructor.

Parameters

<i>ex</i>	the exception to copy
-----------	-----------------------

6.140.1.3 decaf::util::ConcurrentModificationException::ConcurrentModificationException (const ConcurrentModificationException & ex) throw () [inline]

Copy Constructor.

Parameters

<i>ex</i>	the exception to copy, which is an instance of this type
-----------	--

6.140.1.4 `decaf::util::ConcurrentModificationException::ConcurrentModificationException (const char * file, const int lineNumber, const std::exception * cause, const char * msg, ...) throw ()` `[inline]`

Constructor - Initializes the file name and line number where this message occurred.

Sets the message to report, using an optional list of arguments to parse into the message

Parameters

<i>file</i>	The file name where exception occurs
<i>lineNumber</i>	The line number where the exception occurred.
<i>cause</i>	The exception that was the cause for this one to be thrown.
<i>msg</i>	The message to report
...	list of primitives that are formatted into the message

6.140.1.5 `decaf::util::ConcurrentModificationException::ConcurrentModificationException (const std::exception * cause) throw ()` `[inline]`

Constructor.

Parameters

<i>cause</i>	Pointer to the exception that caused this one to be thrown, the object is cloned caller retains ownership.
--------------	--

6.140.1.6 `decaf::util::ConcurrentModificationException::ConcurrentModificationException (const char * file, const int lineNumber, const char * msg, ...) throw ()` `[inline]`

Constructor.

Parameters

<i>file</i>	The file name where exception occurs
<i>lineNumber</i>	The line number where the exception occurred.
<i>msg</i>	The message to report
...	list of primitives that are formatted into the message

6.140.1.7 `virtual decaf::util::ConcurrentModificationException::~ConcurrentModificationException () throw ()` `[virtual]`

6.140.2 Member Function Documentation

6.141 decaf::util::concurrent::ConcurrentStlMap< K, V, COMPARATOR > Class Template Reference 905

6.140.2.1 virtual **ConcurrentModificationException*** decaf::util::
ConcurrentModificationException::clone () const [inline,
virtual]

Clones this exception.

This is useful for cases where you need to preserve the type of the original exception as well as the message. All subclasses should override.

Reimplemented from **decaf::lang::exceptions::RuntimeException** (p. 2317).

The documentation for this class was generated from the following file:

- src/main/decaf/util/**ConcurrentModificationException.h**

6.141 decaf::util::concurrent::ConcurrentStlMap< K, V, COMPARATOR > Class Template Reference

Map (p. 1768) template that wraps around a std::map to provide a more user-friendly interface and to provide common functions that do not exist in std::map.

```
#include <src/main/decaf/util/concurrent/ConcurrentStl-  
Map.h>
```

Inheritance diagram for decaf::util::concurrent::ConcurrentStlMap< K, V, COMPARATOR >:

Public Member Functions

- **ConcurrentStlMap** ()
Default constructor - does nothing.
- **ConcurrentStlMap** (const **ConcurrentStlMap** &source)
Copy constructor - copies the content of the given map into this one.
- **ConcurrentStlMap** (const **Map**< K, V, COMPARATOR > &source)
Copy constructor - copies the content of the given map into this one.
- virtual ~**ConcurrentStlMap** ()
- virtual bool **equals** (const **ConcurrentStlMap** &source) const
- virtual bool **equals** (const **Map**< K, V, COMPARATOR > &source) const
Comparison, equality is dependent on the method of determining if the element are equal.
- virtual void **copy** (const **ConcurrentStlMap** &source)
- virtual void **copy** (const **Map**< K, V, COMPARATOR > &source)
Copies the content of the source map into this map.

- virtual void **clear** () throw (decaf::lang::exceptions::UnsupportedOperationException)

Removes all keys and values from this map.

Exceptions

UnsupportedOperationException	<i>if this map is unmodifiable.</i>
-------------------------------	-------------------------------------

- virtual bool **containsKey** (const K &key) const

Indicates whether or this map contains a value for the given key.

Parameters

key	<i>The key to look up.</i>
-----	----------------------------

Returns

true if this map contains the value, otherwise false.

- virtual bool **containsValue** (const V &value) const

Indicates whether or this map contains a value for the given value, i.e. they are equal, this is done by operator== so the types must pass equivalence testing in this manner.

Parameters

value	<i>The Value to look up.</i>
-------	------------------------------

Returns

true if this map contains the value, otherwise false.

- virtual bool **isEmpty** () const

Returns

*if the **Map** (p. 1768) contains any element or not, TRUE or FALSE*

- virtual int **size** () const

Returns

The number of elements (key/value pairs) in this map.

- virtual V & **get** (const K &key)

*Gets the value mapped to the specified key in the **Map** (p. 1768).*

*If there is no element in the map whose key is equivalent to the key provided then a **NoSuchElementException** (p. 1984) is thrown.*

Parameters

key	<i>The search key.</i>
-----	------------------------

Returns

A reference to the value for the given key.

Exceptions

NoSuchElementException (p. 1984)	<i>if the key requests doesn't exist in the Map (p. 1768).</i>
--	---

- virtual const V & **get** (const K &key) const

*Gets the value mapped to the specified key in the **Map** (p. 1768).*

*If there is no element in the map whose key is equivalent to the key provided then a **NoSuchElementException** (p. 1984) is thrown.*

6.141 decaf::util::concurrent::ConcurrentStlMap< K, V, COMPARATOR > Class Template Reference 907

Parameters

key	<i>The search key.</i>
-----	------------------------

Returns

A {const} reference to the value for the given key.

Exceptions

NoSuchElementException (p. 1984)	<i>if the key requests doesn't exist in the Map (p. 1768).</i>
--	---

- virtual void **put** (const K &key, const V &value)

Sets the value for the specified key.

Parameters

key	<i>The target key.</i>
value	<i>The value to be set.</i>

Exceptions

UnsupportedOperationException	<i>if this map is unmodifiable.</i>
-------------------------------	-------------------------------------

- virtual void **putAll** (const **ConcurrentStlMap**< K, V, COMPARATOR > &other)
- virtual void **putAll** (const **Map**< K, V, COMPARATOR > &other)

*Stores a copy of the Mappings contained in the other **Map** (p. 1768) in this one.*

Parameters

other	<i>A Map (p. 1768) instance whose elements are to all be inserted in this Map (p. 1768).</i>
-------	--

Exceptions

UnsupportedOperationException	<i>If the implementing class does not support the putAll operation.</i>
-------------------------------	---

- virtual V **remove** (const K &key)

Removes the value (key/value pair) for the specified key from the map, returns a copy of the value that was mapped to the key.

Parameters

key	<i>The search key.</i>
-----	------------------------

Returns

a copy of the element that was previously mapped to the given key

Exceptions

NoSuchElementException (p. 1984)	<i>if this key is not in the Map (p. 1768).</i>
UnsupportedOperationException	<i>if this map is unmodifiable.</i>

- virtual std::vector< K > **keySet** () const

Returns a **Set** (p. 2397) view of the mappings contained in this map. The set is backed by the map, so changes to the map are reflected in the set, and vice-versa. If the map is modified while an iteration over the set is in progress (except through the iterator's own remove operation, or through the setValue operation on a map entry returned by the iterator) the results of the iteration are undefined. The set supports element removal, which removes the corresponding mapping from the map, via the **Iterator.remove** (p. 1560), **Set.remove** (p. 861), **removeAll**, **retainAll** and **clear** operations. It does not support the **add** or **addAll** operations.

Returns

the entire set of keys in this map as a `std::vector`.

- virtual `std::vector< V > values () const`

Returns

the entire set of values in this map as a `std::vector`.

- bool **putIfAbsent** (const K &key, const V &value)

If the specified key is not already associated with a value, associate it with the given value.

- bool **remove** (const K &key, const V &value)

Remove entry for key only if currently mapped to given value.

- bool **replace** (const K &key, const V &oldValue, const V &newValue)

Replace entry for key only if currently mapped to given value.

- V **replace** (const K &key, const V &value)

Replace entry for key only if currently mapped to some value.

- virtual void **lock** ()

Locks the object.

- virtual bool **tryLock** ()

Attempts to **Lock** (p. 1682) the object, if the lock is already held by another thread than this method returns false.

- virtual void **unlock** ()

Unlocks the object.

- virtual void **wait** ()

Waits on a signal from this object, which is generated by a call to **Notify**.

- virtual void **wait** (long long millisecs)

Waits on a signal from this object, which is generated by a call to **Notify**.

- virtual void **wait** (long long millisecs, int nanos)

Waits on a signal from this object, which is generated by a call to **Notify**.

- virtual void **notify** ()

Signals a waiter on this object that it can now wake up and continue.

- virtual void **notifyAll** ()

Signals the waiters on this object that it can now wake up and continue.

6.141.1 Detailed Description

```
template<typename K, typename V, typename COMPARATOR = std::less<K>>class decaf::util-
::concurrent::ConcurrentStlMap< K, V, COMPARATOR >
```

Map (p. 1768) template that wraps around a std::map to provide a more user-friendly interface and to provide common functions that do not exist in std::map.

This version of **Map** (p. 1768) extends the **ConcurrentMap** (p. 898) interface and implements all the methods defined in that interface. Unlike a Java ConcurrentHashMap this implementations synchronizes all methods such that any call to this class will block if another thread is already holding a lock, much like the Java HashTable.

Since

1.0

6.141.2 Constructor & Destructor Documentation

6.141.2.1 `template<typename K, typename V, typename COMPARATOR = std::less<K>>
 decaf::util::concurrent::ConcurrentStlMap< K, V, COMPARATOR
 >::ConcurrentStlMap () [inline]`

Default constructor - does nothing.

6.141.2.2 `template<typename K, typename V, typename COMPARATOR = std::less<K>>
 decaf::util::concurrent::ConcurrentStlMap< K, V, COMPARATOR
 >::ConcurrentStlMap (const ConcurrentStlMap< K, V, COMPARATOR > &
 source) [inline]`

Copy constructor - copies the content of the given map into this one.

Parameters

<i>source</i>	The source map.
---------------	-----------------

6.141.2.3 `template<typename K, typename V, typename COMPARATOR = std::less<K>>
 decaf::util::concurrent::ConcurrentStlMap< K, V, COMPARATOR
 >::ConcurrentStlMap (const Map< K, V, COMPARATOR > & source)
 [inline]`

Copy constructor - copies the content of the given map into this one.

Parameters

<i>source</i>	The source map.
---------------	-----------------

6.141.2.4 `template<typename K, typename V, typename COMPARATOR = std::less<K>>
virtual decaf::util::concurrent::ConcurrentStlMap< K, V, COMPARATOR
>::~~ConcurrentStlMap () [inline, virtual]`

6.141.3 Member Function Documentation

6.141.3.1 `template<typename K, typename V, typename COMPARATOR = std::less<K>>
virtual void decaf::util::concurrent::ConcurrentStlMap< K, V,
COMPARATOR >::clear () throw (decaf::lang::exceptions-
::UnsupportedOperationException) [inline,
virtual]`

Removes all keys and values from this map.

Exceptions

<i>Unsupported- OperationException</i>	if this map is unmodifiable.
--	------------------------------

Implements **decaf::util::Map< K, V, COMPARATOR >** (p. 1770).

Referenced by `decaf::util::concurrent::ConcurrentStlMap< Pointer< ProducerId >,
Pointer< ProducerState >, ProducerId::COMPARATOR >::copy()`.

6.141.3.2 `template<typename K, typename V, typename COMPARATOR = std::less<K>>
virtual bool decaf::util::concurrent::ConcurrentStlMap< K, V, COMPARATOR
>::containsKey (const K & key) const [inline, virtual]`

Indicates whether or this map contains a value for the given key.

Parameters

<i>key</i>	The key to look up.
------------	---------------------

Returns

true if this map contains the value, otherwise false.

Implements **decaf::util::Map< K, V, COMPARATOR >** (p. 1771).

Referenced by `decaf::util::concurrent::ConcurrentStlMap< Pointer< ProducerId >,
Pointer< ProducerState >, ProducerId::COMPARATOR >::equals()`, `decaf::util-
::concurrent::ConcurrentStlMap< Pointer< ProducerId >, Pointer< ProducerState >,
ProducerId::COMPARATOR >::putIfAbsent()`, `decaf::util::concurrent::ConcurrentStl-
Map< Pointer< ProducerId >, Pointer< ProducerState >, ProducerId::COMPARAT-
OR >::remove()`, and `decaf::util::concurrent::ConcurrentStlMap< Pointer< ProducerId
>, Pointer< ProducerState >, ProducerId::COMPARATOR >::replace()`.

6.141 decaf::util::concurrent::ConcurrentStlMap< K, V, COMPARATOR > Class Template Reference 911

6.141.3.3 `template<typename K, typename V, typename COMPARATOR = std::less<K>>
virtual bool decaf::util::concurrent::ConcurrentStlMap< K, V, COMPARATOR
>::containsValue (const V & value) const [inline, virtual]`

Indicates whether or this map contains a value for the given value, i.e.

they are equal, this is done by operator== so the types must pass equivalence testing in this manner.

Parameters

<i>value</i>	The Value to look up.
--------------	-----------------------

Returns

true if this map contains the value, otherwise false.

Implements **decaf::util::Map< K, V, COMPARATOR >** (p. 1771).

6.141.3.4 `template<typename K, typename V, typename COMPARATOR = std::less<K>>
virtual void decaf::util::concurrent::ConcurrentStlMap< K, V, COMPARATOR
>::copy (const ConcurrentStlMap< K, V, COMPARATOR > & source)
[inline, virtual]`

Referenced by `decaf::util::concurrent::ConcurrentStlMap< Pointer< ProducerId >, Pointer< ProducerState >, ProducerId::COMPARATOR >::ConcurrentStlMap()`.

6.141.3.5 `template<typename K, typename V, typename COMPARATOR = std::less<K>>
virtual void decaf::util::concurrent::ConcurrentStlMap< K, V, COMPARATOR
>::copy (const Map< K, V, COMPARATOR > & source) [inline,
virtual]`

Copies the content of the source map into this map.

Erases all existing data in this map.

Parameters

<i>source</i>	The source object to copy from.
---------------	---------------------------------

Implements **decaf::util::Map< K, V, COMPARATOR >** (p. 1772).

6.141.3.6 `template<typename K, typename V, typename COMPARATOR = std::less<K>>
virtual bool decaf::util::concurrent::ConcurrentStlMap< K, V, COMPARATOR
>::equals (const ConcurrentStlMap< K, V, COMPARATOR > & source) const
[inline, virtual]`

```
6.141.3.7  template<typename K, typename V, typename COMPARATOR = std::less<K>>
            virtual bool decaf::util::concurrent::ConcurrentStlMap< K, V, COMPARATOR
            >::equals ( const Map< K, V, COMPARATOR > & source ) const  [inline,
            virtual]
```

Comparison, equality is dependent on the method of determining if the element are equal.

Parameters

<i>source</i>	- Map (p. 1768) to compare to this one.
---------------	--

Returns

true if the **Map** (p. 1768) passed is equal in value to this one.

Implements **decaf::util::Map< K, V, COMPARATOR >** (p. 1772).

```
6.141.3.8  template<typename K, typename V, typename COMPARATOR = std::less<K>>
            virtual V& decaf::util::concurrent::ConcurrentStlMap< K, V, COMPARATOR
            >::get ( const K & key ) [inline, virtual]
```

Gets the value mapped to the specified key in the **Map** (p. 1768).

If there is no element in the map whose key is equivalent to the key provided then a **NoSuchElementException** (p. 1984) is thrown.

Parameters

<i>key</i>	The search key.
------------	-----------------

Returns

A reference to the value for the given key.

Exceptions

NoSuchElementException (p. 1984)	if the key requests doesn't exist in the Map (p. 1768).
--	--

Implements **decaf::util::Map< K, V, COMPARATOR >** (p. 1773).

```
6.141.3.9  template<typename K, typename V, typename COMPARATOR = std::less<K>>
            virtual const V& decaf::util::concurrent::ConcurrentStlMap< K, V,
            COMPARATOR >::get ( const K & key ) const  [inline, virtual]
```

Gets the value mapped to the specified key in the **Map** (p. 1768).

6.141 `decaf::util::concurrent::ConcurrentStlMap< K, V, COMPARATOR >` Class Template Reference 913

If there is no element in the map whose key is equivalent to the key provided then a **NoSuchElementException** (p. 1984) is thrown.

Parameters

<i>key</i>	The search key.
------------	-----------------

Returns

A {const} reference to the value for the given key.

Exceptions

<i>NoSuchElementException</i> (p. 1984)	if the key requests doesn't exist in the Map (p. 1768).
---	--

Implements `decaf::util::Map< K, V, COMPARATOR >` (p. 1774).

```
6.141.3.10  template<typename K, typename V, typename COMPARATOR = std::less<K>>
            virtual bool decaf::util::concurrent::ConcurrentStlMap< K, V,
            COMPARATOR>::isEmpty( ) const [inline, virtual]
```

Returns

if the **Map** (p. 1768) contains any element or not, TRUE or FALSE

Implements `decaf::util::Map< K, V, COMPARATOR >` (p. 1774).

```
6.141.3.11  template<typename K, typename V, typename COMPARATOR = std::less<K>>
            virtual std::vector<K> decaf::util::concurrent::ConcurrentStlMap< K, V,
            COMPARATOR>::keySet( ) const [inline, virtual]
```

Returns a **Set** (p. 2397) view of the mappings contained in this map.

The set is backed by the map, so changes to the map are reflected in the set, and vice-versa. If the map is modified while an iteration over the set is in progress (except through the iterator's own remove operation, or through the setValue operation on a map entry returned by the iterator) the results of the iteration are undefined. The set supports element removal, which removes the corresponding mapping from the map, via the **Iterator.remove** (p. 1560), **Set.remove** (p. 861), **removeAll**, **retainAll** and **clear** operations. It does not support the **add** or **addAll** operations.

Returns

the entire set of keys in this map as a `std::vector`.

Implements `decaf::util::Map< K, V, COMPARATOR >` (p. 1775).

6.141.3.12 `template<typename K, typename V, typename COMPARATOR = std::less<K>>
virtual void decaf::util::concurrent::ConcurrentStlMap< K, V,
COMPARATOR >::lock () [inline, virtual]`

Locks the object.

Exceptions

<i>RuntimeException</i>	if an error occurs while locking the object.
-------------------------	--

Implements **decaf::util::concurrent::Synchronizable** (p. 2640).

6.141.3.13 `template<typename K, typename V, typename COMPARATOR = std::less<K>>
virtual void decaf::util::concurrent::ConcurrentStlMap< K, V,
COMPARATOR >::notify () [inline, virtual]`

Signals a waiter on this object that it can now wake up and continue.

Must have this object locked before calling.

Exceptions

<i>IllegalMonitorState-Exception</i>	- if the current thread is not the owner of the the Synchronizable (p. 2639) Object.
<i>RuntimeException</i>	if an error occurs while notifying one of the waiting threads.

Implements **decaf::util::concurrent::Synchronizable** (p. 2641).

6.141.3.14 `template<typename K, typename V, typename COMPARATOR = std::less<K>>
virtual void decaf::util::concurrent::ConcurrentStlMap< K, V,
COMPARATOR >::notifyAll () [inline, virtual]`

Signals the waiters on this object that it can now wake up and continue.

Must have this object locked before calling.

Exceptions

<i>IllegalMonitorState-Exception</i>	- if the current thread is not the owner of the the Synchronizable (p. 2639) Object.
<i>RuntimeException</i>	if an error occurs while notifying the waiting threads.

Implements **decaf::util::concurrent::Synchronizable** (p. 2642).

6.141 decaf::util::concurrent::ConcurrentStlMap< K, V, COMPARATOR > Class Template Reference 915

6.141.3.15 `template<typename K, typename V, typename COMPARATOR = std::less<K>>
virtual void decaf::util::concurrent::ConcurrentStlMap< K, V,
COMPARATOR >::put (const K & key, const V & value) [inline,
virtual]`

Sets the value for the specified key.

Parameters

<i>key</i>	The target key.
<i>value</i>	The value to be set.

Exceptions

<i>Unsupported- OperationException</i>	if this map is unmodifiable.
--	------------------------------

Implements **decaf::util::Map< K, V, COMPARATOR >** (p. 1776).

Referenced by `decaf::util::concurrent::ConcurrentStlMap< Pointer< ProducerId >, Pointer< ProducerState >, ProducerId::COMPARATOR >::putAll()`, `decaf::util::concurrent::ConcurrentStlMap< Pointer< ProducerId >, Pointer< ProducerState >, ProducerId::COMPARATOR >::putIfAbsent()`, and `decaf::util::concurrent::ConcurrentStlMap< Pointer< ProducerId >, Pointer< ProducerState >, ProducerId::COMPARATOR >::replace()`.

6.141.3.16 `template<typename K, typename V, typename COMPARATOR = std::less<K>>
virtual void decaf::util::concurrent::ConcurrentStlMap< K, V,
COMPARATOR >::putAll (const ConcurrentStlMap< K, V, COMPARATOR > &
other) [inline, virtual]`

Referenced by `decaf::util::concurrent::ConcurrentStlMap< Pointer< ProducerId >, Pointer< ProducerState >, ProducerId::COMPARATOR >::copy()`.

6.141.3.17 `template<typename K, typename V, typename COMPARATOR = std::less<K>>
virtual void decaf::util::concurrent::ConcurrentStlMap< K, V,
COMPARATOR >::putAll (const Map< K, V, COMPARATOR > & other)
[inline, virtual]`

Stores a copy of the Mappings contained in the other **Map** (p. 1768) in this one.

Parameters

<i>other</i>	A Map (p. 1768) instance whose elements are to all be inserted in this Map (p. 1768).
--------------	---

Exceptions

<i>Unsupported- OperationException</i>	If the implementing class does not support the putAll operation.
--	--

Implements **decaf::util::Map< K, V, COMPARATOR >** (p. 1777).

6.141.3.18 `template<typename K, typename V, typename COMPARATOR = std::less<K>>
bool decaf::util::concurrent::ConcurrentStlMap< K, V, COMPARATOR
>::putIfAbsent (const K & key, const V & value) [inline, virtual]`

If the specified key is not already associated with a value, associate it with the given value.

This is equivalent to

```
if( !map.containsKey( key ) ) {
    map.put( key, value );
    return true;
} else {
    return false;
}
```

except that the action is performed atomically.

Parameters

<i>key</i>	The key to map the value to.
<i>value</i>	The value to map to the given key.

Returns

true if the put operation was performed otherwise return false which indicates there was a value previously mapped to the key.

Exceptions

<i>Unsupported- OperationException</i>	if the put operation is not supported by this map
--	---

Implements **decaf::util::concurrent::ConcurrentMap< K, V, COMPARATOR >** (p. 899).

6.141 decaf::util::concurrent::ConcurrentStlMap< K, V, COMPARATOR > Class Template Reference 917

6.141.3.19 `template<typename K, typename V, typename COMPARATOR = std::less<K>>
virtual V decaf::util::concurrent::ConcurrentStlMap< K, V, COMPARATOR
>::remove (const K & key) [inline, virtual]`

Removes the value (key/value pair) for the specified key from the map, returns a copy of the value that was mapped to the key.

Parameters

<i>key</i>	The search key.
------------	-----------------

Returns

a copy of the element that was previously mapped to the given key

Exceptions

<i>NoSuchElementException</i> (p. 1984)	if this key is not in the Map (p. 1768).
<i>UnsupportedOperationException</i>	if this map is unmodifiable.

Implements **decaf::util::Map< K, V, COMPARATOR >** (p. 1777).

6.141.3.20 `template<typename K, typename V, typename COMPARATOR = std::less<K>>
bool decaf::util::concurrent::ConcurrentStlMap< K, V, COMPARATOR
>::remove (const K & key, const V & value) [inline, virtual]`

Remove entry for key only if currently mapped to given value.

Acts as

```
if( map.containsKey( key ) && ( map.get( key ) == value ) ) {  
    map.remove( key );  
    return true;  
} else {  
    return false;  
}
```

except that the action is performed atomically.

Parameters

<i>key</i>	key with which the specified value is associated.
<i>value</i>	value associated with the specified key.

Returns

true if the value was removed, false otherwise

Implements **decaf::util::concurrent::ConcurrentMap**< K, V, COMPARATOR > (p. 900).

```
6.141.3.21  template<typename K, typename V, typename COMPARATOR = std::less<K>>
             bool decaf::util::concurrent::ConcurrentStlMap< K, V, COMPARATOR
             >::replace ( const K & key, const V & oldValue, const V & newValue )
             [inline, virtual]
```

Replace entry for key only if currently mapped to given value.

Acts as

```
if( map.containsKey( key ) && ( map.get( key ) == oldValue ) ) {
    map.put( key, newValue );
    return true;
} else {
    return false;
}
```

except that the action is performed atomically.

Parameters

<i>key</i>	key with which the specified value is associated.
<i>oldValue</i>	value expected to be associated with the specified key.
<i>newValue</i>	value to be associated with the specified key.

Returns

true if the value was replaced

Implements **decaf::util::concurrent::ConcurrentMap**< K, V, COMPARATOR > (p. 900).

```
6.141.3.22  template<typename K, typename V, typename COMPARATOR = std::less<K>>
             V decaf::util::concurrent::ConcurrentStlMap< K, V, COMPARATOR
             >::replace ( const K & key, const V & value ) [inline, virtual]
```

Replace entry for key only if currently mapped to some value.

Acts as

```
if( map.containsKey( key ) ) {
    return map.put( key, value );
} else {
```

6.141 `decaf::util::concurrent::ConcurrentStlMap< K, V, COMPARATOR >` Class Template Reference 919

```
        throw NoSuchElementException (p.1984) (...);  
    };
```

except that the action is performed atomically.

Parameters

<i>key</i>	key with which the specified value is associated.
<i>value</i>	value to be associated with the specified key.

Returns

copy of the previous value associated with specified key, or throws an **NoSuchElementException** (p. 1984) if there was no mapping for key.

Exceptions

<i>NoSuchElementException</i> (p. 1984)	if there was no previous mapping.
---	-----------------------------------

Implements `decaf::util::concurrent::ConcurrentMap< K, V, COMPARATOR >` (p.901).

```
6.141.3.23  template<typename K, typename V, typename COMPARATOR = std::less<K>>  
            virtual int decaf::util::concurrent::ConcurrentStlMap< K, V, COMPARATOR  
            >::size( ) const [inline, virtual]
```

Returns

The number of elements (key/value pairs) in this map.

Implements `decaf::util::Map< K, V, COMPARATOR >` (p. 1778).

```
6.141.3.24  template<typename K, typename V, typename COMPARATOR = std::less<K>>  
            virtual bool decaf::util::concurrent::ConcurrentStlMap< K, V,  
            COMPARATOR>::tryLock( ) [inline, virtual]
```

Attempts to **Lock** (p. 1682) the object, if the lock is already held by another thread than this method returns false.

Returns

true if the lock was acquired, false if it is already held by another thread.

Exceptions

<i>RuntimeException</i>	if an error occurs while locking the object.
-------------------------	--

Implements **decaf::util::concurrent::Synchronizable** (p. 2643).

```
6.141.3.25  template<typename K, typename V, typename COMPARATOR = std::less<K>>
            virtual void decaf::util::concurrent::ConcurrentStlMap< K, V,
            COMPARATOR >::unlock ( ) [inline, virtual]
```

Unlocks the object.

Exceptions

<i>RuntimeException</i>	if an error occurs while unlocking the object.
-------------------------	--

Implements **decaf::util::concurrent::Synchronizable** (p. 2645).

```
6.141.3.26  template<typename K, typename V, typename COMPARATOR = std::less<K>>
            virtual std::vector<V> decaf::util::concurrent::ConcurrentStlMap< K, V,
            COMPARATOR >::values ( ) const [inline, virtual]
```

Returns

the entire set of values in this map as a std::vector.

Implements **decaf::util::Map< K, V, COMPARATOR >** (p. 1779).

Referenced by **decaf::util::concurrent::ConcurrentStlMap< Pointer< ProducerId >, Pointer< ProducerState >, ProducerId::COMPARATOR >::values()**.

```
6.141.3.27  template<typename K, typename V, typename COMPARATOR = std::less<K>>
            virtual void decaf::util::concurrent::ConcurrentStlMap< K, V,
            COMPARATOR >::wait ( ) [inline, virtual]
```

Waits on a signal from this object, which is generated by a call to Notify.

Must have this object locked before calling.

Exceptions

<i>RuntimeException</i>	if an error occurs while waiting on the object.
<i>InterruptedException</i>	if the wait is interrupted before it completes.
<i>IllegalMonitorState-Exception</i>	- if the current thread is not the owner of the the Synchronizable (p. 2639) Object.

Implements **decaf::util::concurrent::Synchronizable** (p. 2646).

6.141 decaf::util::concurrent::ConcurrentStlMap< K, V, COMPARATOR > Class Template Reference 921

6.141.3.28 `template<typename K, typename V, typename COMPARATOR = std::less<K>>
virtual void decaf::util::concurrent::ConcurrentStlMap< K, V,
COMPARATOR >::wait (long long millisecs) [inline, virtual]`

Waits on a signal from this object, which is generated by a call to Notify.

Must have this object locked before calling. This wait will timeout after the specified time interval.

Parameters

<i>millisecs</i>	the time in milliseconds to wait, or WAIT_INFINITE
------------------	--

Exceptions

<i>RuntimeException</i>	if an error occurs while waiting on the object.
<i>InterruptedException</i>	if the wait is interrupted before it completes.
<i>IllegalMonitorState-Exception</i>	- if the current thread is not the owner of the the Synchronizable (p. 2639) Object.

Implements **decaf::util::concurrent::Synchronizable** (p. 2647).

6.141.3.29 `template<typename K, typename V, typename COMPARATOR = std::less<K>>
virtual void decaf::util::concurrent::ConcurrentStlMap< K, V,
COMPARATOR >::wait (long long millisecs, int nanos) [inline,
virtual]`

Waits on a signal from this object, which is generated by a call to Notify.

Must have this object locked before calling. This wait will timeout after the specified time interval. This method is similar to the one argument wait function except that it add a finer grained control over the amount of time that it waits by adding in the additional nanosecond argument.

NOTE: The ability to wait accurately at a nanosecond scale depends on the platform and OS that the Decaf API is running on, some systems do not provide an accurate enough clock to provide this level of granularity.

Parameters

<i>millisecs</i>	the time in milliseconds to wait, or WAIT_INFINITE
<i>nanos</i>	additional time in nanoseconds with a range of 0-999999

Exceptions

<i>IllegalArgument-Exception</i>	if an error occurs or the nanos argument is not in the range of [0-999999]
<i>RuntimeException</i>	if an error occurs while waiting on the object.
<i>InterruptedException</i>	if the wait is interrupted before it completes.
<i>IllegalMonitorState-Exception</i>	- if the current thread is not the owner of the the Synchronizable (p. 2639) Object.

Implements **decaf::util::concurrent::Synchronizable** (p. 2648).

The documentation for this class was generated from the following file:

- src/main/decaf/util/concurrent/**ConcurrentStlMap.h**

6.142 decaf::util::concurrent::locks::Condition Class Reference

Condition (p. 921) factors out the **Mutex** (p. 1960) monitor methods (wait, notify and notifyAll) into distinct objects to give the effect of having multiple wait-sets per object, by combining them with the use of arbitrary **Lock** (p. 1684) implementations.

```
#include <src/main/decaf/util/concurrent/locks/Condition.h>
```

Public Member Functions

- virtual **~Condition** ()
- virtual void **await** ()=0
Causes the current thread to wait until it is signaled or interrupted.
- virtual void **awaitUninterruptibly** ()=0
Causes the current thread to wait until it is signalled.
- virtual long long **awaitNanos** (long long nanosTimeout)=0
Causes the current thread to wait until it is signaled or interrupted, or the specified waiting time elapses.
- virtual bool **await** (long long time, const **TimeUnit** &unit)=0
Causes the current thread to wait until it is signaled or interrupted, or the specified waiting time elapses.
- virtual bool **awaitUntil** (const **Date** &deadline)=0
- virtual void **signal** ()=0
Wakes up one waiting thread.
- virtual void **signalAll** ()=0
Wakes up all waiting threads.

6.142.1 Detailed Description

Condition (p. 921) factors out the **Mutex** (p. 1960) monitor methods (wait, notify and notifyAll) into distinct objects to give the effect of having multiple wait-sets per object, by combining them with the use of arbitrary **Lock** (p. 1684) implementations.

Where a **Lock** (p. 1684) replaces the use of synchronized statements, a **Condition** (p. 921) replaces the use of the Object monitor methods.

Conditions (also known as condition queues or condition variables) provide a means for one thread to suspend execution (to "wait") until notified by another thread that some state condition may now be true. Because access to this shared state information occurs

in different threads, it must be protected, so a lock of some form is associated with the condition. The key property that waiting for a condition provides is that it atomically releases the associated lock and suspends the current thread.

A **Condition** (p. 921) instance is intrinsically bound to a lock. To obtain a **Condition** (p. 921) instance for a particular **Lock** (p. 1684) instance use its `newCondition()` method.

As an example, suppose we have a bounded buffer which supports put and take methods. If a take is attempted on an empty buffer, then the thread will block until an item becomes available; if a put is attempted on a full buffer, then the thread will block until a space becomes available. We would like to keep waiting put threads and take threads in separate wait-sets so that we can use the optimization of only notifying a single thread at a time when items or spaces become available in the buffer. This can be achieved using two **Condition** (p. 921) instances.

```
class BoundedBuffer { Lock* lock = new ReentrantLock(); Condition* notFull = lock->newCondition(); Condition* notEmpty = lock->newCondition();
```

```
Object* items = new Object[100]; int putptr, takeptr, count;
```

```
public void put( Object* x ) throw( InterruptedException ) { lock->lock(); try { while( count == 100 ) notFull->await() (p. 923); items[putptr] = x; if (++putptr == 100) putptr = 0; ++count; notEmpty->signal() (p. 927); } catch(...) { lock->unlock(); } }
```

```
public Object take() throw( InterruptedException ) { lock->lock(); try { while(count == 0) notEmpty->await() (p. 923); Object x = items[takeptr]; if (++takeptr == 100) takeptr = 0; --count; notFull->signal() (p. 927); return x; } catch(...) { lock->unlock(); } }
```

(The `ArrayBlockingQueue` class provides this functionality, so there is no reason to implement this sample usage class.)

Implementation Considerations

When waiting upon a **Condition** (p. 921), a "spurious wakeup" is permitted to occur, in general, as a concession to the underlying platform semantics. This has little practical impact on most application programs as a **Condition** (p. 921) should always be waited upon in a loop, testing the state predicate that is being waited for. An implementation is free to remove the possibility of spurious wakeups but it is recommended that applications programmers always assume that they can occur and so always wait in a loop.

The three forms of condition waiting (interruptible, non-interruptible, and timed) may differ in their ease of implementation on some platforms and in their performance characteristics. In particular, it may be difficult to provide these features and maintain specific semantics such as ordering guarantees. Further, the ability to interrupt the actual suspension of the thread may not always be feasible to implement on all platforms.

Consequently, an implementation is not required to define exactly the same guarantees or semantics for all three forms of waiting, nor is it required to support interruption of the actual suspension of the thread.

An implementation is required to clearly document the semantics and guarantees provided by each of the waiting methods, and when an implementation does support interruption of thread suspension then it must obey the interruption semantics as defined in this interface.

As interruption generally implies cancellation, and checks for interruption are often infrequent, an implementation can favor responding to an interrupt over normal method return. This is true even if it can be shown that the interrupt occurred after another action may have unblocked the thread. An implementation should document this behavior.

Since

1.0

6.142.2 Constructor & Destructor Documentation

6.142.2.1 `virtual decaf::util::concurrent::locks::Condition::~Condition ()`
`[inline, virtual]`

6.142.3 Member Function Documentation

6.142.3.1 `virtual void decaf::util::concurrent::locks::Condition::await ()` `[pure virtual]`

Causes the current thread to wait until it is signaled or interrupted.

The lock associated with this **Condition** (p. 921) is atomically released and the current thread becomes disabled for thread scheduling purposes and lies dormant until one of four things happens:

- * Some other thread invokes the **signal()** (p. 927) method for this **Condition** (p. 921) and the current thread happens to be chosen as the thread to be awakened; or
- * Some other thread invokes the **signalAll()** (p. 927) method for this **Condition** (p. 921); or
- * Some other thread interrupts the current thread, and interruption of thread suspension is supported; or
- * A "spurious wakeup" occurs.

In all cases, before this method can return the current thread must re-acquire the lock associated with this condition. When the thread returns it is guaranteed to hold this lock.

If the current thread:

- * has its interrupted status set on entry to this method; or
- * is interrupted while waiting and interruption of thread suspension is supported,

then `InterruptedException` is thrown and the current thread's interrupted status is cleared. It is not specified, in the first case, whether or not the test for interruption occurs before the lock is released.

Implementation Considerations

The current thread is assumed to hold the lock associated with this **Condition** (p. 921) when this method is called. It is up to the implementation to determine if this is the case and if not, how to respond. Typically, an exception will be thrown (such as `IllegalMonitorStateException`) and the implementation must document that fact.

An implementation can favor responding to an interrupt over normal method return in response to a signal. In that case the implementation must ensure that the signal is redirected to another waiting thread, if there is one.

Exceptions

<i>RuntimeException</i>	if an unexpected error occurs while trying to wait on the Condition (p. 921).
<i>InterruptedException</i>	if the current thread is interrupted (and interruption of thread suspension is supported)
<i>IllegalMonitorStateException</i>	if the caller is not the lock owner.

6.142.3.2 `virtual bool decaf::util::concurrent::locks::Condition::await (long long time,
const TimeUnit & unit) [pure virtual]`

Causes the current thread to wait until it is signaled or interrupted, or the specified waiting time elapses.

This method is behaviorally equivalent to:

`awaitNanos(unit.toNanos(time)) > 0`

Parameters

<i>time</i>	- the maximum time to wait
<i>unit</i>	- the time unit of the time argument

Returns

false if the waiting time detectably elapsed before return from the method, else true

Exceptions

<i>RuntimeException</i>	if an unexpected error occurs while trying to wait on the Condition (p. 921).
<i>InterruptedException</i>	if the current thread is interrupted (and interruption of thread suspension is supported)
<i>IllegalMonitorStateException</i>	if the caller is not the lock owner.

6.142.3.3 `virtual long long decaf::util::concurrent::locks::Condition::awaitNanos (long long nanosTimeout) [pure virtual]`

Causes the current thread to wait until it is signaled or interrupted, or the specified waiting time elapses.

The lock associated with this condition is atomically released and the current thread becomes disabled for thread scheduling purposes and lies dormant until one of five things happens:

* Some other thread invokes the **signal()** (p. 927) method for this **Condition** (p. 921) and the current thread happens to be chosen as the thread to be awakened; or * Some

other thread invokes the **signalAll()** (p. 927) method for this **Condition** (p. 921); or * Some other thread interrupts the current thread, and interruption of thread suspension is supported; or * The specified waiting time elapses; or * A "spurious wakeup" occurs.

In all cases, before this method can return the current thread must re-acquire the lock associated with this condition. When the thread returns it is guaranteed to hold this lock.

If the current thread:

* has its interrupted status set on entry to this method; or * is interrupted while waiting and interruption of thread suspension is supported,

then `InterruptedException` is thrown and the current thread's interrupted status is cleared. It is not specified, in the first case, whether or not the test for interruption occurs before the lock is released.

The method returns an estimate of the number of nanoseconds remaining to wait given the supplied `nanosTimeout` value upon return, or a value less than or equal to zero if it timed out. This value can be used to determine whether and how long to re-wait in cases where the wait returns but an awaited condition still does not hold. Typical uses of this method take the following form:

```
synchronized boolean aMethod( long timeout, const TimeUnit& unit ) { long nanos-
Timeout = unit.toNanos(timeout); while (!conditionBeingWaitedFor) { if (nanosTimeout
> 0) nanosTimeout = theCondition->awaitNanos(nanosTimeout); else return false; } //
... }
```

Design note: This method requires a nanosecond argument so as to avoid truncation errors in reporting remaining times. Such precision loss would make it difficult for programmers to ensure that total waiting times are not systematically shorter than specified when re-waits occur.

Implementation Considerations

The current thread is assumed to hold the lock associated with this **Condition** (p. 921) when this method is called. It is up to the implementation to determine if this is the case and if not, how to respond. Typically, an exception will be thrown (such as `IllegalMonitorStateException`) and the implementation must document that fact.

An implementation can favor responding to an interrupt over normal method return in response to a signal, or over indicating the elapse of the specified waiting time. In either case the implementation must ensure that the signal is redirected to another waiting thread, if there is one.

Parameters

<i>nanos-Timeout</i>	- the maximum time to wait, in nanoseconds
----------------------	--

Returns

an estimate of the `nanosTimeout` value minus the time spent waiting upon return from this method. A positive value may be used as the argument to a subsequent call to this method to finish waiting out the desired time. A value less than or equal to zero indicates that no time remains.

Exceptions

<i>RuntimeException</i>	if an unexpected error occurs while trying to wait on the Condition (p. 921).
<i>InterruptedException</i>	if the current thread is interrupted (and interruption of thread suspension is supported)
<i>IllegalMonitorStateException</i>	if the caller is not the lock owner.

6.142.3.4 virtual void decaf::util::concurrent::locks::Condition::awaitUninterruptibly () [pure virtual]

Causes the current thread to wait until it is signalled.

The lock associated with this condition is atomically released and the current thread becomes disabled for thread scheduling purposes and lies dormant until one of three things happens:

* Some other thread invokes the **signal()** (p. 927) method for this **Condition** (p. 921) and the current thread happens to be chosen as the thread to be awakened; or * Some other thread invokes the **signalAll()** (p. 927) method for this **Condition** (p. 921); or * A "spurious wakeup" occurs.

In all cases, before this method can return the current thread must re-acquire the lock associated with this condition. When the thread returns it is guaranteed to hold this lock.

If the current thread's interrupted status is set when it enters this method, or it is interrupted while waiting, it will continue to wait until signalled. When it finally returns from this method its interrupted status will still be set.

Implementation Considerations

The current thread is assumed to hold the lock associated with this **Condition** (p. 921) when this method is called. It is up to the implementation to determine if this is the case and if not, how to respond. Typically, an exception will be thrown (such as *IllegalMonitorStateException*) and the implementation must document that fact.

Exceptions

<i>RuntimeException</i>	if an unexpected error occurs while trying to wait on the Condition (p. 921).
<i>IllegalMonitorStateException</i>	if the caller is not the lock owner.

6.142.3.5 virtual bool decaf::util::concurrent::locks::Condition::awaitUntil (const Date & deadline) [pure virtual]

6.142.3.6 virtual void decaf::util::concurrent::locks::Condition::signal () [pure virtual]

Wakes up one waiting thread.

If any threads are waiting on this condition then one is selected for waking up. That thread must then re-acquire the lock before returning from await.

Exceptions

<i>RuntimeException</i>	if an unexpected error occurs while trying to wait on the Condition (p. 921).
-------------------------	--

6.142.3.7 `virtual void decaf::util::concurrent::locks::Condition::signalAll ()`
`[pure virtual]`

Wakes up all waiting threads.

If any threads are waiting on this condition then they are all woken up. Each thread must re-acquire the lock before it can return from await.

Exceptions

<i>RuntimeException</i>	if an unexpected error occurs while trying to wait on the Condition (p. 921).
-------------------------	--

The documentation for this class was generated from the following file:

- `src/main/decaf/util/concurrent/locks/Condition.h`

6.143 decaf::util::concurrent::ConditionHandle Class Reference

```
#include <src/main/decaf/internal/util/concurrent/unix/-  
ConditionHandle.h>
```

Public Member Functions

- **ConditionHandle** ()
- **~ConditionHandle** ()
- **ConditionHandle** ()
- **~ConditionHandle** ()

Data Fields

- `pthread_cond_t` **condition**
- **MutexHandle** * **mutex**
- `HANDLE` **semaphore**
- `CRITICAL_SECTION` **criticalSection**
- volatile unsigned int **numWaiting**
- volatile unsigned int **numWake**
- volatile unsigned int **generation**

6.143.1 Constructor & Destructor Documentation

6.143.1.1 `decaf::util::concurrent::ConditionHandle::ConditionHandle ()` `[inline]`

6.143.1.2 `decaf::util::concurrent::ConditionHandle::~~ConditionHandle ()`
`[inline]`

6.143.1.3 `decaf::util::concurrent::ConditionHandle::ConditionHandle ()` `[inline]`

6.143.1.4 `decaf::util::concurrent::ConditionHandle::~~ConditionHandle ()`
`[inline]`

6.143.2 Field Documentation

6.143.2.1 `pthread_cond_t decaf::util::concurrent::ConditionHandle::condition`

6.143.2.2 `CRITICAL_SECTION decaf::util::concurrent::ConditionHandle::critical-
Section`

6.143.2.3 `volatile unsigned int decaf::util::concurrent::ConditionHandle::generation`

6.143.2.4 `MutexHandle * decaf::util::concurrent::ConditionHandle::mutex`

6.143.2.5 `volatile unsigned int decaf::util::concurrent::ConditionHandle::numWaiting`

6.143.2.6 `volatile unsigned int decaf::util::concurrent::ConditionHandle::numWake`

6.143.2.7 `HANDLE decaf::util::concurrent::ConditionHandle::semaphore`

The documentation for this class was generated from the following files:

- `src/main/decaf/internal/util/concurrent/unix/ConditionHandle.h`
- `src/main/decaf/internal/util/concurrent/windows/ConditionHandle.h`

6.144 decaf::internal::util::concurrent::ConditionImpl Class - Reference

```
#include <src/main/decaf/internal/util/concurrent/Condition-  
Impl.h>
```

Static Public Member Functions

- static `decaf::util::concurrent::ConditionHandle * create (decaf::util-
::concurrent::MutexHandle *mutex)`

Creates the Condition object and attaches it to the given MutexHandle.

- static void **destroy** (**decaf::util::concurrent::ConditionHandle** *handle)
Destroy a previously create Condition instance.
- static void **wait** (**decaf::util::concurrent::ConditionHandle** *condition)
Waits for the condition to be signaled.
- static void **wait** (**decaf::util::concurrent::ConditionHandle** *condition, long long mills, long long nanos)
Waits for the condition to be signaled or for the time specified to ellapse.
- static void **notify** (**decaf::util::concurrent::ConditionHandle** *condition)
Signals one Thread that is waiting on this condition to wake up.
- static void **notifyAll** (**decaf::util::concurrent::ConditionHandle** *condition)
Signals all Threads that is waiting on this condition to wake up.

6.144.1 Member Function Documentation

6.144.1.1 **static decaf::util::concurrent::ConditionHandle* decaf::internal::util::concurrent::ConditionImpl::create (decaf::util::concurrent::MutexHandle * mutex) [static]**

Creates the Condition object and attaches it to the given MutexHandle.

Parameters

<i>mutex</i>	the Mutex handle that this Condition is attached to.
--------------	--

Returns

a newly constructed Condition handle that is attached to the given handle.

6.144.1.2 **static void decaf::internal::util::concurrent::ConditionImpl::destroy (decaf::util::concurrent::ConditionHandle * handle) [static]**

Destroy a previously create Condition instance.

Parameters

<i>handle</i>	The Condition handle to be destroyed.
---------------	---------------------------------------

6.144.1.3 **static void decaf::internal::util::concurrent::ConditionImpl::notify (decaf::util::concurrent::ConditionHandle * condition) [static]**

Signals one Thread that is waiting on this condition to wake up.

Parameters

<i>condition</i>	the handle to the condition to wait on.
------------------	---

6.144.1.4 static void decaf::internal::util::concurrent::ConditionImpl::notifyAll (decaf::util::concurrent::ConditionHandle * *condition*) [static]

Signals all Threads that is waiting on this condition to wake up.

Parameters

<i>condition</i>	the handle to the condition to wait on.
------------------	---

6.144.1.5 static void decaf::internal::util::concurrent::ConditionImpl::wait (decaf::util::concurrent::ConditionHandle * *condition*) [static]

Waits for the condition to be signaled.

Parameters

<i>condition</i>	the handle to the condition to wait on.
------------------	---

6.144.1.6 static void decaf::internal::util::concurrent::ConditionImpl::wait (decaf::util::concurrent::ConditionHandle * *condition*, long long *mills*, long long *nanos*) [static]

Waits for the condition to be signaled or for the time specified to ellapse.

Parameters

<i>condition</i>	the handle to the condition to wait on.
<i>mills</i>	the time in milliseconds to wait for the condition to be signaled.
<i>nanos</i>	additional time in nanoseconds to wait for the thread to be signaled.

The documentation for this class was generated from the following file:

- src/main/decaf/internal/util/concurrent/ConditionImpl.h

6.145 decaf::net::ConnectException Class Reference

```
#include <src/main/decaf/net/ConnectException.h>
```

Inheritance diagram for decaf::net::ConnectException:

Public Member Functions

- **ConnectException** () throw ()
Default Constructor.
- **ConnectException** (const **Exception** &ex) throw ()
Conversion Constructor from some other Exception.
- **ConnectException** (const **ConnectException** &ex) throw ()
Copy Constructor.
- **ConnectException** (const char *file, const int lineNumber, const std::exception ***cause**, const char *msg,...) throw ()
Constructor - Initializes the file name and line number where this message occurred.
- **ConnectException** (const std::exception ***cause**) throw ()
Constructor.
- **ConnectException** (const char *file, const int lineNumber, const char *msg,...) throw ()
Constructor - Initializes the file name and line number where this message occurred.
- virtual **ConnectException** * **clone** () const
Clones this exception.
- virtual ~**ConnectException** () throw ()

6.145.1 Constructor & Destructor Documentation

6.145.1.1 **decaf::net::ConnectException::ConnectException ()** throw ()
[inline]

Default Constructor.

6.145.1.2 **decaf::net::ConnectException::ConnectException (const Exception & ex)** throw () [inline]

Conversion Constructor from some other Exception.

Parameters

ex	An exception that should become this type of Exception
-----------	--

6.145.1.3 **decaf::net::ConnectException::ConnectException (const ConnectException & ex)** throw () [inline]

Copy Constructor.

Parameters

ex	An exception that should become this type of Exception
-----------	--

6.145.1.4 `decaf::net::ConnectException::ConnectException (const char * file, const int lineNumber, const std::exception * cause, const char * msg, ...) throw ()`
`[inline]`

Constructor - Initializes the file name and line number where this message occurred.

Sets the message to report, using an optional list of arguments to parse into the message

Parameters

<i>file</i>	The file name where exception occurs
<i>lineNumber</i>	The line number where the exception occurred.
<i>cause</i>	The exception that was the cause for this one to be thrown.
<i>msg</i>	The message to report
...	list of primitives that are formatted into the message

6.145.1.5 `decaf::net::ConnectException::ConnectException (const std::exception * cause) throw ()` `[inline]`

Constructor.

Parameters

<i>cause</i>	Pointer to the exception that caused this one to be thrown, the object is cloned caller retains ownership.
--------------	--

6.145.1.6 `decaf::net::ConnectException::ConnectException (const char * file, const int lineNumber, const char * msg, ...) throw ()` `[inline]`

Constructor - Initializes the file name and line number where this message occurred.

Sets the message to report, using an optional list of arguments to parse into the message

Parameters

<i>file</i>	The file name where exception occurs
<i>lineNumber</i>	The line number where the exception occurred.
<i>msg</i>	The message to report
...	list of primitives that are formatted into the message

6.145.1.7 `virtual decaf::net::ConnectException::~~ConnectException () throw ()`
`[inline, virtual]`

6.145.2 Member Function Documentation

6.145.2.1 `virtual ConnectException* decaf::net::ConnectException::clone () const`
`[inline, virtual]`

Clones this exception.

This is useful for cases where you need to preserve the type of the original exception as well as the message. All subclasses should override.

Returns

a new Exception instance that is a copy of this Exception object.

Reimplemented from `decaf::net::SocketException` (p. 2473).

The documentation for this class was generated from the following file:

- `src/main/decaf/net/ConnectException.h`

6.146 cms::Connection Class Reference

The client's connection to its provider.

```
#include <src/main/cms/Connection.h>
```

Inheritance diagram for `cms::Connection`:

Public Member Functions

- virtual `~Connection ()` throw ()
- virtual void `close ()=0`
Closes this connection as well as any Sessions created from it (and those Sessions' consumers and producers).
- virtual const `ConnectionMetaData * getMetaData ()` const =0
Gets the metadata for this connection.
- virtual `Session * createSession ()=0`
Creates an AUTO_ACKNOWLEDGE Session (p. 2361).
- virtual `Session * createSession (Session::AcknowledgeMode ackMode)=0`
Creates a new Session (p. 2361) to work for this Connection (p. 933) using the specified acknowledgment mode.
- virtual `std::string getClientID ()` const =0
Get the Client Id for this session, the client Id is provider specific and is either assigned by the connection factory or set using the setClientID method.
- virtual void `setClientID (const std::string &clientID)=0`
Sets the client identifier for this connection.
- virtual `ExceptionListener * getExceptionListener ()` const =0

Gets the registered Exception Listener for this connection.

- virtual void **setExceptionListener** (**ExceptionListener** *listener)=0

Sets the registered Exception Listener for this connection.

6.146.1 Detailed Description

The client's connection to its provider.

Connections support concurrent use.

A connection serves several purposes:

- It encapsulates an open connection with a JMS provider. It typically represents an open TCP/IP socket between a client and the service provider software.
- Its creation is where client authentication takes place.
- It can specify a unique client identifier.
- It provides a **ConnectionMetaData** (p. 978) object.
- It supports an optional **ExceptionListener** (p. 1286) object.

Because the creation of a connection involves setting up authentication and communication, a connection is a relatively heavy-weight object. Most clients will do all their messaging with a single connection. Other more advanced applications may use several connections. The CMS API does not architect a reason for using multiple connections; however, there may be operational reasons for doing so.

A CMS client typically creates a connection, one or more sessions, and a number of message producers and consumers. When a connection is created, it is in stopped mode. That means that no messages are being delivered.

It is typical to leave the connection in stopped mode until setup is complete (that is, until all message consumers have been created). At that point, the client calls the connection's start method, and messages begin arriving at the connection's consumers. This setup convention minimizes any client confusion that may result from asynchronous message delivery while the client is still in the process of setting itself up.

A connection can be started immediately, and the setup can be done afterwards. Clients that do this must be prepared to handle asynchronous message delivery while they are still in the process of setting up.

A message producer can send messages while a connection is stopped.

Since

1.0

6.146.2 Constructor & Destructor Documentation

6.146.2.1 virtual cms::Connection::~Connection () throw () [virtual]

6.146.3 Member Function Documentation

6.146.3.1 `virtual void cms::Connection::close ()` [pure virtual]

Closes this connection as well as any Sessions created from it (and those Sessions' consumers and producers).

Exceptions

<i>CMSException</i> (p. 826)
--

Implements **cms::Closeable** (p. 816).

Implemented in **activemq::core::ActiveMQConnection** (p. 195).

6.146.3.2 `virtual Session* cms::Connection::createSession ()` [pure virtual]

Creates an AUTO_ACKNOWLEDGE **Session** (p. 2361).

Exceptions

<i>CMSException</i> (p. 826)
--

Implemented in **activemq::core::ActiveMQConnection** (p. 196).

6.146.3.3 `virtual Session* cms::Connection::createSession (Session::AcknowledgeMode ackMode)` [pure virtual]

Creates a new **Session** (p. 2361) to work for this **Connection** (p. 933) using the specified acknowledgment mode.

Parameters

<i>ackMode</i>	the Acknowledgment Mode to use.
----------------	---------------------------------

Exceptions

<i>CMSException</i> (p. 826)
--

Implemented in **activemq::core::ActiveMQConnection** (p. 196), and **activemq::core::ActiveMQXAConnection** (p. 433).

6.146.3.4 `virtual std::string cms::Connection::getClientID () const [pure virtual]`

Get the Client Id for this session, the client Id is provider specific and is either assigned by the connection factory or set using the `setClientID` method.

Returns

Client Id String for this **Connection** (p. 933).

Exceptions

<i>CMSEException</i> (p. 826)	if the provider fails to return the client id or an internal error occurs.
---	--

Implemented in **activemq::core::ActiveMQConnection** (p. 198).

6.146.3.5 `virtual ExceptionListener* cms::Connection::getExceptionListener () const [pure virtual]`

Gets the registered Exception Listener for this connection.

Returns

pointer to an exception listener or NULL

Implemented in **activemq::core::ActiveMQConnection** (p. 199).

6.146.3.6 `virtual const ConnectionMetaData* cms::Connection::getMetaData () const [pure virtual]`

Gets the metadata for this connection.

Returns

the connection MetaData pointer (caller does not own it).

Exceptions

<i>CMSEException</i> (p. 826)	if the provider fails to get the connection metadata for this connection.
---	---

See also

ConnectionMetaData (p. 978)

Since

2.0

Implemented in **activemq::core::ActiveMQConnection** (p. 200).

6.146.3.7 **virtual void cms::Connection::setClientID (const std::string & clientID)**
[pure virtual]

Sets the client identifier for this connection.

The preferred way to assign a CMS client's client identifier is for it to be configured in a client-specific **ConnectionFactory** (p. 955) object and transparently assigned to the **Connection** (p. 933) object it creates.

If a client sets the client identifier explicitly, it must do so immediately after it creates the connection and before any other action on the connection is taken. After this point, setting the client identifier is a programming error that should throw an **IllegalStateException** (p. 1419).

Parameters

<i>clientID</i>	The unique client identifier to assign to the Connection (p. 933).
-----------------	---

Exceptions

CMSException (p. 826)	if the provider fails to set the client id due to some internal error.
<i>InvalidClientID-Exception</i>	if the id given is somehow invalid or is a duplicate.
IllegalStateException (p. 1419)	if the client tries to set the id after a Connection (p. 933) method has been called.

Implemented in **activemq::core::ActiveMQConnection** (p. 207).

6.146.3.8 **virtual void cms::Connection::setExceptionListener (ExceptionListener * listener)** [pure virtual]

Sets the registered Exception Listener for this connection.

Parameters

<i>listener</i>	pointer to and ExceptionListener (p. 1286)
-----------------	---

Implemented in **activemq::core::ActiveMQConnection** (p. 209).

The documentation for this class was generated from the following file:

- src/main/cms/**Connection.h**

6.147 activemq::commands::ConnectionControl Class Reference

```
#include <src/main/activemq/commands/ConnectionControl.h>
```

Inheritance diagram for activemq::commands::ConnectionControl:

Public Member Functions

- **ConnectionControl** ()
- virtual **~ConnectionControl** ()
- virtual unsigned char **getDataStructureType** () const
*Get the **DataStructure** (p. 1133) Type as defined in CommandTypes.h.*
- virtual **ConnectionControl** * **cloneDataStructure** () const
Clone this object and return a new instance that the caller now owns, this will be an exact copy of this one.
- virtual void **copyDataStructure** (const **DataStructure** *src)
Copy the contents of the passed object into this objects members, overwriting any existing data.
- virtual std::string **toString** () const
*Returns a string containing the information for this **DataStructure** (p. 1133) such as its type and value of its elements.*
- virtual bool **equals** (const **DataStructure** *value) const
*Compares the **DataStructure** (p. 1133) passed in to this one, and returns if they are equivalent.*
- virtual bool **isClose** () const
- virtual void **setClose** (bool close)
- virtual bool **isExit** () const
- virtual void **setExit** (bool exit)
- virtual bool **isFaultTolerant** () const
- virtual void **setFaultTolerant** (bool faultTolerant)
- virtual bool **isResume** () const
- virtual void **setResume** (bool resume)
- virtual bool **isSuspend** () const
- virtual void **setSuspend** (bool suspend)
- virtual const std::string & **getConnectedBrokers** () const
- virtual std::string & **getConnectedBrokers** ()
- virtual void **setConnectedBrokers** (const std::string &connectedBrokers)
- virtual const std::string & **getReconnectTo** () const
- virtual std::string & **getReconnectTo** ()
- virtual void **setReconnectTo** (const std::string &reconnectTo)
- virtual bool **isRebalanceConnection** () const
- virtual void **setRebalanceConnection** (bool rebalanceConnection)
- virtual bool **isConnectionControl** () const

- virtual **Pointer**< **Command** > **visit** (**activemq::state::CommandVisitor** *visitor)

Allows a Visitor to visit this command and return a response to the command based on the command type being visited.

Static Public Attributes

- static const unsigned char **ID_CONNECTIONCONTROL** = 18

Protected Attributes

- bool **close**
- bool **exit**
- bool **faultTolerant**
- bool **resume**
- bool **suspend**
- std::string **connectedBrokers**
- std::string **reconnectTo**
- bool **rebalanceConnection**

6.147.1 Constructor & Destructor Documentation

6.147.1.1 **activemq::commands::ConnectionControl::ConnectionControl** ()

6.147.1.2 **virtual activemq::commands::ConnectionControl::~~ConnectionControl** () [virtual]

6.147.2 Member Function Documentation

6.147.2.1 **virtual ConnectionControl*** **activemq::commands::ConnectionControl::cloneDataStructure** () const [virtual]

Clone this obbject and return a new instance that the caller now owns, this will be an exact copy of this one.

Returns

new copy of this object.

Implements **activemq::commands::DataStructure** (p. 1133).

6.147.2.2 `virtual void activemq::commands::ConnectionControl::copyDataStructure (const DataStructure * src)`
[virtual]

Copy the contents of the passed object into this objects members, overwriting any existing data.

Returns

src - Source Object

Reimplemented from `activemq::commands::BaseCommand` (p. 493).

6.147.2.3 `virtual bool activemq::commands::ConnectionControl::equals (const DataStructure * value) const` [virtual]

Compares the `DataStructure` (p. 1133) passed in to this one, and returns if they are equivalent.

Equivalent here means that they are of the same type, and that each element of the objects are the same.

Returns

true if `DataStructure` (p. 1133)'s are Equal.

Reimplemented from `activemq::commands::BaseCommand` (p. 494).

6.147.2.4 `virtual const std::string& activemq::commands::ConnectionControl::getConnectedBrokers () const`
[virtual]

6.147.2.5 `virtual std::string& activemq::commands::ConnectionControl::getConnectedBrokers ()` [virtual]

6.147.2.6 `virtual unsigned char activemq::commands::ConnectionControl::getDataStructureType () const` [virtual]

Get the `DataStructure` (p. 1133) Type as defined in CommandTypes.h.

Returns

The type of the data structure

Implements `activemq::commands::DataStructure` (p. 1137).

6.147.2.7 `virtual const std::string& activemq::commands::ConnectionControl::getReconnectTo () const` [virtual]

- 6.147.2.8 `virtual std::string& activemq::commands::ConnectionControl::get-ReconnectTo () [virtual]`
- 6.147.2.9 `virtual bool activemq::commands::ConnectionControl::isClose () const [virtual]`
- 6.147.2.10 `virtual bool activemq::commands::ConnectionControl::isConnectionControl () const [inline, virtual]`

Returns

an answer of true to the **isConnectionControl()** (p. 941) query.

Reimplemented from **activemq::commands::BaseCommand** (p. 495).

- 6.147.2.11 `virtual bool activemq::commands::ConnectionControl::isExit () const [virtual]`
- 6.147.2.12 `virtual bool activemq::commands::ConnectionControl::isFaultTolerant () const [virtual]`
- 6.147.2.13 `virtual bool activemq::commands::ConnectionControl::isRebalance-Connection () const [virtual]`
- 6.147.2.14 `virtual bool activemq::commands::ConnectionControl::isResume () const [virtual]`
- 6.147.2.15 `virtual bool activemq::commands::ConnectionControl::isSuspend () const [virtual]`
- 6.147.2.16 `virtual void activemq::commands::ConnectionControl::setClose (bool close) [virtual]`
- 6.147.2.17 `virtual void activemq::commands::ConnectionControl::set-ConnectedBrokers (const std::string & connectedBrokers) [virtual]`
- 6.147.2.18 `virtual void activemq::commands::ConnectionControl::setExit (bool exit) [virtual]`
- 6.147.2.19 `virtual void activemq::commands::ConnectionControl::setFaultTolerant (bool faultTolerant) [virtual]`
- 6.147.2.20 `virtual void activemq::commands::ConnectionControl::setRebalanceConnection (bool rebalanceConnection) [virtual]`

6.147.2.21 `virtual void activemq::commands::ConnectionControl::setReconnectTo (const std::string & reconnectTo) [virtual]`

6.147.2.22 `virtual void activemq::commands::ConnectionControl::setResume (bool resume) [virtual]`

6.147.2.23 `virtual void activemq::commands::ConnectionControl::setSuspend (bool suspend) [virtual]`

6.147.2.24 `virtual std::string activemq::commands::ConnectionControl::toString () const [virtual]`

Returns a string containing the information for this **DataSet** (p. 1133) such as its type and value of its elements.

Returns

formatted string useful for debugging.

Reimplemented from **activemq::commands::BaseCommand** (p. 498).

6.147.2.25 `virtual Pointer<Command> activemq::commands::ConnectionControl::visit (activemq::state::CommandVisitor * visitor) [virtual]`

Allows a Visitor to visit this command and return a response to the command based on the command type being visited.

The command will call the proper processXXX method in the visitor.

Returns

a **Response** (p. 2298) to the visitor being called or NULL if no response.

Implements **activemq::commands::Command** (p. 871).

6.147.3 Field Documentation

6.147.3.1 `bool activemq::commands::ConnectionControl::close [protected]`

6.147.3.2 `std::string activemq::commands::ConnectionControl::connectedBrokers [protected]`

6.147.3.3 `bool activemq::commands::ConnectionControl::exit [protected]`

6.147.3.4 `bool activemq::commands::ConnectionControl::faultTolerant [protected]`

6.147.3.5 `const unsigned char activemq::commands::ConnectionControl::ID_CONNECTIONCONTROL = 18` `[static]`

6.147.3.6 `bool activemq::commands::ConnectionControl::rebalanceConnection` `[protected]`

6.147.3.7 `std::string activemq::commands::ConnectionControl::reconnectTo` `[protected]`

6.147.3.8 `bool activemq::commands::ConnectionControl::resume` `[protected]`

6.147.3.9 `bool activemq::commands::ConnectionControl::suspend` `[protected]`

The documentation for this class was generated from the following file:

- `src/main/activemq/commands/ConnectionControl.h`

6.148 `activemq::wireformat::openwire::marshal::generated::ConnectionControlMarshaller` Class Reference

Marshaling code for Open Wire Format for **ConnectionControlMarshaller** (p. 943).

```
#include <src/main/activemq/wireformat/openwire/marshal/generated/-
ConnectionControlMarshaller.h>
```

Inheritance diagram for `activemq::wireformat::openwire::marshal::generated::ConnectionControlMarshaller`:

Public Member Functions

- **ConnectionControlMarshaller** ()
- virtual `~ConnectionControlMarshaller` ()
- virtual `commands::DataStructure * createObject` () const
Creates a new instance of the class that this class is a marshaling director for.
- virtual unsigned char `getDataStructureType` () const
Gets the DataStructureType that this class marshals/unmarshals.
- virtual void `tightUnmarshal` (`OpenWireFormat *wireFormat`, `commands::DataStructure *dataStructure`, `decaf::io::DataInputStream *dataIn`, `utils::BooleanStream *bs`)
Tight Un-marhsal to the given stream.
- virtual int `tightMarshal1` (`OpenWireFormat *wireFormat`, `commands::DataStructure *dataStructure`, `utils::BooleanStream *bs`)

6.148 activemq::wireformat::openwire::marshal::generated::ConnectionControlMarshaller Class

Reference

945

Tight Marshal to the given stream.

- virtual void **tightMarshal2** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataOutputStream** *dataOut, **utils::BooleanStream** *bs)

Tight Marshal to the given stream.

- virtual void **looseUnmarshal** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataInputStream** *dataIn)

Loose Un-marshal to the given stream.

- virtual void **looseMarshal** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataOutputStream** *dataOut)

Tight Marshal to the given stream.

6.148.1 Detailed Description

Marshaling code for Open Wire Format for **ConnectionControlMarshaller** (p. 943).

NOTE!: This file is auto generated - do not modify! if you need to make a change, please see the Java Classes in the activemq-openwire-generator module

6.148.2 Constructor & Destructor Documentation

6.148.2.1 **activemq::wireformat::openwire::marshal::generated::ConnectionControlMarshaller::ConnectionControlMarshaller** ()
[inline]

6.148.2.2 **virtual activemq::wireformat::openwire::marshal::generated::ConnectionControlMarshaller::~~ConnectionControlMarshaller** ()
[inline, virtual]

6.148.3 Member Function Documentation

6.148.3.1 **virtual commands::DataStructure* activemq::wireformat::openwire::marshal::generated::ConnectionControlMarshaller::createObject** ()
const [virtual]

Creates a new instance of the class that this class is a marshaling director for.

Returns

newly allocated Command

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1120).

6.148.3.2 `virtual unsigned char activemq::wireformat::openwire::marshal::generated-
::ConnectionControlMarshaller::getDataStructureType () const
[virtual]`

Gets the DataStructureType that this class marshals/unmarshals.

Returns

byte Id of this classes DataStructureType

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller**
(p. 1122).

6.148.3.3 `virtual void activemq::wireformat::openwire::marshal::generated::-
ConnectionControlMarshaller::looseMarshal (OpenWireFormat * format,
commands::DataStructure * command, decaf::io::DataOutputStream * ds
) [virtual]`

Tight Marshal to the given stream.

Parameters

<i>format</i>	- The OpenWireFormat properties
<i>command</i>	- the object to Marshal
<i>ds</i>	- DataOutputStream to marshal to

Exceptions

<i>IOException</i>	if an error occurs.
--------------------	---------------------

Reimplemented from **activemq::wireformat::openwire::marshal::generated::Base-
CommandMarshaller** (p. 500).

6.148.3.4 `virtual void activemq::wireformat::openwire::marshal-
::generated::ConnectionControlMarshaller::looseUnmarshal (OpenWireFormat * format,
commands::DataStructure * command,
decaf::io::DataInputStream * dis) [virtual]`

Loose Un-marshal to the given stream.

Parameters

<i>format</i>	- The OpenWireFormat properties
<i>command</i>	- the object to Un-Marshal
<i>dis</i>	- the DataInputStream to Un-Marshal from

6.148 activemq::wireformat::openwire::marshal::generated::ConnectionControlMarshaller Class

Reference

947

Exceptions

<i>IOException</i>	if an error occurs.
--------------------	---------------------

Reimplemented from **activemq::wireformat::openwire::marshal::generated::BaseCommandMarshaller** (p. 501).

6.148.3.5 `virtual int activemq::wireformat::openwire::marshal::generated::ConnectionControlMarshaller::tightMarshal1 (OpenWireFormat * format, commands::DataStructure * command, utils::BooleanStream * bs) [virtual]`

Tight Marshal to the given stream.

Parameters

<i>format</i>	The OpenWireFormat properties
<i>command</i>	The object to Marshal
<i>bs</i>	The boolean stream to marshal to.

Exceptions

<i>IOException</i>	if an error occurs.
--------------------	---------------------

Reimplemented from **activemq::wireformat::openwire::marshal::generated::BaseCommandMarshaller** (p. 503).

6.148.3.6 `virtual void activemq::wireformat::openwire::marshal::generated::ConnectionControlMarshaller::tightMarshal2 (OpenWireFormat * format, commands::DataStructure * command, decaf::io::DataOutputStream * ds, utils::BooleanStream * bs) [virtual]`

Tight Marshal to the given stream.

Parameters

<i>format</i>	- The OpenWireFormat properties
<i>command</i>	- the object to Marshal
<i>ds</i>	- the DataOutputStream to Marshal to
<i>bs</i>	- boolean stream to marshal to.

Exceptions

<i>IOException</i>	if an error occurs.
--------------------	---------------------

Reimplemented from **activemq::wireformat::openwire::marshal::generated::BaseCommandMarshaller** (p. 504).

```
6.148.3.7 virtual void activemq::wireformat::openwire::marshal-
::generated::ConnectionControlMarshaller::tightUnmarshal (
OpenWireFormat * format, commands::DataStructure * command,
decaf::io::DataInputStream * dis, utils::BooleanStream * bs )
[virtual]
```

Tight Un-marhsal to the given stream.

Parameters

<i>format</i>	- The OpenwireFormat properties
<i>command</i>	- the object to Un-Marshal
<i>dis</i>	- the DataInputStream to Un-Marshal from
<i>bs</i>	- boolean stream to unmarshal from.

Exceptions

<i>IOException</i>	if an error occurs.
--------------------	---------------------

Reimplemented from **activemq::wireformat::openwire::marshal::generated::Base-CommandMarshaller** (p. 505).

The documentation for this class was generated from the following file:

- src/main/activemq/wireformat/openwire/marshal/generated/**ConnectionControl-Marshaller.h**

6.149 activemq::commands::ConnectionError Class Reference

```
#include <src/main/activemq/commands/ConnectionError.h>
```

Inheritance diagram for activemq::commands::ConnectionError:

Public Member Functions

- **ConnectionError** ()
- virtual **~ConnectionError** ()
- virtual unsigned char **getDataStructureType** () const
*Get the **DataStructure** (p. 1133) Type as defined in CommandTypes.h.*
- virtual **ConnectionError** * **cloneDataStructure** () const
Clone this obbect and return a new instance that the caller now owns, this will be an exact copy of this one.
- virtual void **copyDataStructure** (const **DataStructure** *src)
Copy the contents of the passed object into this objects members, overwriting any existing data.

- virtual std::string **toString** () const
*Returns a string containing the information for this **DataSetructure** (p. 1133) such as its type and value of its elements.*
- virtual bool **equals** (const **DataSetructure** *value) const
*Compares the **DataSetructure** (p. 1133) passed in to this one, and returns if they are equivalent.*
- virtual const **Pointer** < **BrokerError** > & **getException** () const
- virtual **Pointer**< **BrokerError** > & **getException** ()
- virtual void **setException** (const **Pointer**< **BrokerError** > &exception)
- virtual const **Pointer** < **ConnectionId** > & **getConnectionId** () const
- virtual **Pointer**< **ConnectionId** > & **getConnectionId** ()
- virtual void **setConnectionId** (const **Pointer**< **ConnectionId** > &connectionId)
- virtual **Pointer**< **Command** > **visit** (activemq::state::CommandVisitor *visitor)
Allows a Visitor to visit this command and return a response to the command based on the command type being visited.

Static Public Attributes

- static const unsigned char **ID_CONNECTIONERROR** = 16

Protected Attributes

- **Pointer**< **BrokerError** > **exception**
- **Pointer**< **ConnectionId** > **connectionId**

6.149.1 Constructor & Destructor Documentation

6.149.1.1 **activemq::commands::ConnectionError::ConnectionError** ()

6.149.1.2 **virtual activemq::commands::ConnectionError::~~ConnectionError** ()
[virtual]

6.149.2 Member Function Documentation

6.149.2.1 **virtual ConnectionError* activemq::commands::-ConnectionError::cloneDataSetructure** () const
[virtual]

Clone this obbject and return a new instance that the caller now owns, this will be an exact copy of this one.

Returns

new copy of this object.

Implements **activemq::commands::DataSetructure** (p. 1133).

6.149.2.2 `virtual void activemq::commands::ConnectionError::copyDataStructure (const DataStructure * src) [virtual]`

Copy the contents of the passed object into this objects members, overwriting any existing data.

Returns

src - Source Object

Reimplemented from `activemq::commands::BaseCommand` (p. 493).

6.149.2.3 `virtual bool activemq::commands::ConnectionError::equals (const DataStructure * value) const [virtual]`

Compares the **DataStructure** (p. 1133) passed in to this one, and returns if they are equivalent.

Equivalent here means that they are of the same type, and that each element of the objects are the same.

Returns

true if **DataStructure** (p. 1133)'s are Equal.

Reimplemented from `activemq::commands::BaseCommand` (p. 494).

6.149.2.4 `virtual const Pointer<ConnectionId>& activemq::commands::ConnectionError::getConnectionId () const [virtual]`

6.149.2.5 `virtual Pointer<ConnectionId>& activemq::commands::ConnectionError::getConnectionId () [virtual]`

6.149.2.6 `virtual unsigned char activemq::commands::ConnectionError::getDataStructureType () const [virtual]`

Get the **DataStructure** (p. 1133) Type as defined in CommandTypes.h.

Returns

The type of the data structure

Implements `activemq::commands::DataStructure` (p. 1137).

6.149.2.7 `virtual const Pointer<BrokerError>& activemq::commands::ConnectionError::getException () const [virtual]`

6.149.2.8 virtual **Pointer**<**BrokerError**>& **activemq::commands::ConnectionError::getException** () [virtual]

6.149.2.9 virtual void **activemq::commands::ConnectionError::setConnectionId** (const **Pointer**< **ConnectionId** > & *connectionId*) [virtual]

6.149.2.10 virtual void **activemq::commands::ConnectionError::setException** (const **Pointer**< **BrokerError** > & *exception*) [virtual]

6.149.2.11 virtual std::string **activemq::commands::ConnectionError::toString** () const [virtual]

Returns a string containing the information for this **DataSet** (p. 1133) such as its type and value of its elements.

Returns

formatted string useful for debugging.

Reimplemented from **activemq::commands::BaseCommand** (p. 498).

6.149.2.12 virtual **Pointer**<**Command**> **activemq::commands::ConnectionError::visit** (**activemq::state::CommandVisitor** * *visitor*) [virtual]

Allows a Visitor to visit this command and return a response to the command based on the command type being visited.

The command will call the proper processXXX method in the visitor.

Returns

a **Response** (p. 2298) to the visitor being called or NULL if no response.

Implements **activemq::commands::Command** (p. 871).

6.149.3 Field Documentation

6.149.3.1 **Pointer**<**ConnectionId**> **activemq::commands::ConnectionError::connectionId** [protected]

6.149.3.2 **Pointer**<**BrokerError**> **activemq::commands::ConnectionError::exception** [protected]

6.149.3.3 const unsigned char **activemq::commands::ConnectionError::ID_CONNECTIONERROR** = 16 [static]

The documentation for this class was generated from the following file:

- src/main/activemq/commands/**ConnectionError.h**

6.150 activemq::wireformat::openwire::marshal::generated:- ConnectionErrorMarshaller Class Reference

Marshaling code for Open Wire Format for **ConnectionErrorMarshaller** (p. 951).

```
#include <src/main/activemq/wireformat/openwire/marshal/generated/-  
ConnectionErrorMarshaller.h>
```

Inheritance diagram for activemq::wireformat::openwire::marshal::generated:-
ConnectionErrorMarshaller:

Public Member Functions

- **ConnectionErrorMarshaller** ()
- virtual **~ConnectionErrorMarshaller** ()
- virtual **commands::DataStructure * createObject** () const
Creates a new instance of the class that this class is a marshaling director for.
- virtual unsigned char **getDataStructureType** () const
Gets the DataStructureType that this class marshals/unmarshals.
- virtual void **tightUnmarshal** (**OpenWireFormat** *wireFormat, **commands:-DataStructure** *dataStructure, **decaf::io::DataInputStream** *dataIn, **utils:-BooleanStream** *bs)
Tight Un-marhsal to the given stream.
- virtual int **tightMarshal1** (**OpenWireFormat** *wireFormat, **commands::Data-Structure** *dataStructure, **utils::BooleanStream** *bs)
Tight Marhsal to the given stream.
- virtual void **tightMarshal2** (**OpenWireFormat** *wireFormat, **commands::Data-Structure** *dataStructure, **decaf::io::DataOutputStream** *dataOut, **utils:-BooleanStream** *bs)
Tight Marhsal to the given stream.
- virtual void **looseUnmarshal** (**OpenWireFormat** *wireFormat, **commands:-DataStructure** *dataStructure, **decaf::io::DataInputStream** *dataIn)
Loose Un-marhsal to the given stream.
- virtual void **looseMarshal** (**OpenWireFormat** *wireFormat, **commands::Data-Structure** *dataStructure, **decaf::io::DataOutputStream** *dataOut)
Tight Marhsal to the given stream.

6.150.1 Detailed Description

Marshaling code for Open Wire Format for **ConnectionErrorMarshaller** (p. 951).

NOTE!: This file is auto generated - do not modify! if you need to make a change, please see the Java Classes in the activemq-openwire-generator module

6.150 activemq::wireformat::openwire::marshal::generated::ConnectionErrorMarshaller Class

Reference

953

6.150.2 Constructor & Destructor Documentation

6.150.2.1 **activemq::wireformat::openwire::marshal::generated::ConnectionErrorMarshaller::ConnectionErrorMarshaller ()**
[inline]

6.150.2.2 **virtual activemq::wireformat::openwire::marshal::generated::ConnectionErrorMarshaller::~~ConnectionErrorMarshaller ()**
[inline, virtual]

6.150.3 Member Function Documentation

6.150.3.1 **virtual commands::DataStructure* activemq::wireformat::openwire::marshal::generated::ConnectionErrorMarshaller::createObject ()**
const [virtual]

Creates a new instance of the class that this class is a marshaling director for.

Returns

newly allocated Command

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1120).

6.150.3.2 **virtual unsigned char activemq::wireformat::openwire::marshal::generated::ConnectionErrorMarshaller::getDataStructureType () const**
[virtual]

Gets the DataStructureType that this class marshals/unmarshals.

Returns

byte Id of this classes DataStructureType

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1122).

6.150.3.3 **virtual void activemq::wireformat::openwire::marshal::generated::ConnectionErrorMarshaller::looseMarshal (OpenWireFormat * *format*, commands::DataStructure * *command*, decaf::io::DataOutputStream * *ds*)** [virtual]

Tight Marshal to the given stream.

Parameters

<i>format</i>	- The OpenWireFormat properties
<i>command</i>	- the object to Marshal
<i>ds</i>	- DataOutputStream to marshal to

Exceptions

<i>IOException</i>	if an error occurs.
--------------------	---------------------

Reimplemented from **activemq::wireformat::openwire::marshal::generated::Base-CommandMarshaller** (p. 500).

6.150.3.4 `virtual void activemq::wireformat::openwire::marshal::generated::ConnectionErrorMarshaller::looseUnmarshal (OpenWireFormat * format, commands::DataStructure * command, decaf::io::DataInputStream * dis) [virtual]`

Loose Un-marhsal to the given stream.

Parameters

<i>format</i>	- The OpenWireFormat properties
<i>command</i>	- the object to Un-Marshal
<i>dis</i>	- the DataInputStream to Un-Marshal from

Exceptions

<i>IOException</i>	if an error occurs.
--------------------	---------------------

Reimplemented from **activemq::wireformat::openwire::marshal::generated::Base-CommandMarshaller** (p. 501).

6.150.3.5 `virtual int activemq::wireformat::openwire::marshal::generated::ConnectionErrorMarshaller::tightMarshal1 (OpenWireFormat * format, commands::DataStructure * command, utils::BooleanStream * bs) [virtual]`

Tight Marhsal to the given stream.

Parameters

<i>format</i>	The OpenWireFormat properties
<i>command</i>	The object to Marshal
<i>bs</i>	The boolean stream to marshal to.

Exceptions

<i>IOException</i>	if an error occurs.
--------------------	---------------------

Reimplemented from **activemq::wireformat::openwire::marshal::generated::Base-CommandMarshaller** (p. 503).

6.150 activemq::wireformat::openwire::marshal::generated::ConnectionError-Marshaller Class

Reference

955

6.150.3.6 virtual void **activemq::wireformat::openwire::marshal::generated::ConnectionErrorMarshaller::tightMarshal2** (**OpenWireFormat** * *format*, **commands::DataStructure** * *command*, **decaf::io::DataOutputStream** * *ds*, **utils::BooleanStream** * *bs*) [virtual]

Tight Marhsal to the given stream.

Parameters

<i>format</i>	- The OpenWireFormat properties
<i>command</i>	- the object to Marshal
<i>ds</i>	- the DataOutputStream to Marshal to
<i>bs</i>	- boolean stream to marshal to.

Exceptions

<i>IOException</i>	if an error occurs.
--------------------	---------------------

Reimplemented from **activemq::wireformat::openwire::marshal::generated::Base-CommandMarshaller** (p. 504).

6.150.3.7 virtual void **activemq::wireformat::openwire::marshal::generated::ConnectionErrorMarshaller::tightUnmarshal** (**OpenWireFormat** * *format*, **commands::DataStructure** * *command*, **decaf::io::DataInputStream** * *dis*, **utils::BooleanStream** * *bs*) [virtual]

Tight Un-marhsal to the given stream.

Parameters

<i>format</i>	- The OpenWireFormat properties
<i>command</i>	- the object to Un-Marshal
<i>dis</i>	- the DataInputStream to Un-Marshal from
<i>bs</i>	- boolean stream to unmarshal from.

Exceptions

<i>IOException</i>	if an error occurs.
--------------------	---------------------

Reimplemented from **activemq::wireformat::openwire::marshal::generated::Base-CommandMarshaller** (p. 505).

The documentation for this class was generated from the following file:

- src/main/activemq/wireformat/openwire/marshal/generated/**ConnectionError-Marshaller.h**

6.151 cms::ConnectionFactory Class Reference

Defines the interface for a factory that creates connection objects, the **Connection** (p. 933) objects returned implement the CMS **Connection** (p. 933) interface and hide the CMS Provider specific implementation details behind that interface.

```
#include <src/main/cms/ConnectionFactory.h>
```

Inheritance diagram for cms::ConnectionFactory:

Public Member Functions

- virtual **~ConnectionFactory** () throw ()
- virtual **Connection** * **createConnection** ()=0
Creates a connection with the default user identity.
- virtual **cms::Connection** * **createConnection** (const std::string &username, const std::string &password)=0
Creates a connection with the default specified identity.
- virtual **cms::Connection** * **createConnection** (const std::string &username, const std::string &password, const std::string &clientId)=0
Creates a connection with the specified user identity.

Static Public Member Functions

- static **ConnectionFactory** * **createCMSConnectionFactory** (const std::string &brokerURI)
Static method that is used to create a provider specific connection factory.

6.151.1 Detailed Description

Defines the interface for a factory that creates connection objects, the **Connection** (p. 933) objects returned implement the CMS **Connection** (p. 933) interface and hide the CMS Provider specific implementation details behind that interface.

A Client creates a new **ConnectionFactory** (p. 955) either directly by instantiating the provider specific implementation of the factory or by using the static method `createCMSConnectionFactory` which all providers are required to implement.

Since

1.0

6.151.2 Constructor & Destructor Documentation

6.151.2.1 `virtual cms::ConnectionFactory::~~ConnectionFactory () throw ()`
[virtual]

6.151.3 Member Function Documentation

6.151.3.1 `static ConnectionFactory* cms::ConnectionFactory::create-
CMSConnectionFactory (const std::string & brokerURI)`
[static]

Static method that is used to create a provider specific connection factory.

The provider implements this method in their library and returns an instance of a - **ConnectionFactory** (p.955) derived object. Clients can use this method to remain abstracted from the specific CMS implementation being used.

Parameters

<i>brokerURI</i>	The remote address to use to connect to the Provider.
------------------	---

Returns

A pointer to a provider specific implementation of the **ConnectionFactory** (p.955) interface, the caller is responsible for deleting this resource.

Exceptions

CMSException (p. 826)	if an internal error occurs while creating the ConnectionFactory (p.955).
---------------------------------	--

6.151.3.2 `virtual Connection* cms::ConnectionFactory::createConnection ()`
[pure virtual]

Creates a connection with the default user identity.

The connection is created in stopped mode. No messages will be delivered until the **Connection.start** (p. 2534) method is explicitly called.

Returns

A pointer to a connection object, caller owns the pointer and is responsible for closing the connection and deleting the instance.

Exceptions

CMSException (p. 826)	if an internal error occurs while creating the Connection (p.933).
---------------------------------	---

Implemented in **activemq::core::ActiveMQConnectionFactory** (p. 217).

6.151.3.3 **virtual cms::Connection* cms::ConnectionFactory::createConnection**
 (const std::string & *username*, const std::string & *password*) [pure
 virtual]

Creates a connection with the default specified identity.

The connection is created in stopped mode. No messages will be delivered until the **Connection.start** (p. 2534) method is explicitly called. The username and password values passed here do not change the defaults, subsequent calls to the parameterless createConnection will continue to use the default values that were set in the Constructor.

Parameters

<i>username</i>	The user name used to authenticate with the Provider.
<i>password</i>	The password used to authenticate with the Provider.

Returns

A pointer to a connection object, caller owns the pointer and is responsible for closing the connection and deleting the instance.

Exceptions

CMSException (p. 826)	if an internal error occurs while creating the Connection (p. 933).
---------------------------------	--

Implemented in **activemq::core::ActiveMQConnectionFactory** (p. 217).

6.151.3.4 **virtual cms::Connection* cms::ConnectionFactory::createConnection** (
 const std::string & *username*, const std::string & *password*, const std::string &
clientId) [pure virtual]

Creates a connection with the specified user identity.

The connection is created in stopped mode. No messages will be delivered until the **Connection.start** (p. 2534) method is explicitly called. The user name and password values passed here do not change the defaults, subsequent calls to the parameterless createConnection will continue to use the default values that were set in the Constructor.

Parameters

<i>username</i>	The user name used to authenticate with the Provider.
<i>password</i>	The password used to authenticate with the Provider.
<i>clientId</i>	The Client Id assigned to connection. If the id is the empty string ("") then a random client Id is created for this connection.

Returns

A pointer to a connection object, caller owns the pointer and is responsible for closing the connection and deleting the instance.

Exceptions

CMSException (p. 826)	if an internal error occurs while creating the Connection (p. 933).
---------------------------------	--

Implemented in **activemq::core::ActiveMQConnectionFactory** (p. 218).

The documentation for this class was generated from the following file:

- src/main/cms/**ConnectionFactory.h**

6.152 activemq::exceptions::ConnectionFailedException Class - Reference

```
#include <src/main/activemq/exceptions/ConnectionFailed-Exception.h>
```

Inheritance diagram for activemq::exceptions::ConnectionFailedException:

Public Member Functions

- **ConnectionFailedException** ()
- **ConnectionFailedException** (const **exceptions::ActiveMQException** &ex)
- **ConnectionFailedException** (const **ConnectionFailedException** &ex)
- **ConnectionFailedException** (const char *file, const int lineNumber, const char *msg,...)
- virtual **ConnectionFailedException** * **clone** () const
Clones this exception.
- virtual ~**ConnectionFailedException** () throw ()

6.152.1 Constructor & Destructor Documentation

6.152.1.1 **activemq::exceptions::ConnectionFailedException::ConnectionFailedException ()** `[inline]`

6.152.1.2 **activemq::exceptions::ConnectionFailedException::ConnectionFailedException (const exceptions::ActiveMQException & ex)** `[inline]`

6.152.1.3 **activemq::exceptions::ConnectionFailedException::ConnectionFailedException** (const ConnectionFailedException & ex)
[inline]

6.152.1.4 **activemq::exceptions::ConnectionFailedException::ConnectionFailedException** (const char * file, const int lineNumber, const char * msg, ...)
[inline]

6.152.1.5 **virtual activemq::exceptions::ConnectionFailedException::~~ConnectionFailedException** () throw () [inline, virtual]

6.152.2 Member Function Documentation

6.152.2.1 **virtual ConnectionFailedException* activemq::exceptions::ConnectionFailedException::clone** () const [inline, virtual]

Clones this exception.

This is useful for cases where you need to preserve the type of the original exception as well as the message. All subclasses should override.

Returns

Copy of this Exception object

Reimplemented from **activemq::exceptions::ActiveMQException** (p. 263).

The documentation for this class was generated from the following file:

- src/main/activemq/exceptions/**ConnectionFailedException.h**

6.153 activemq::commands::ConnectionId Class Reference

```
#include <src/main/activemq/commands/ConnectionId.h>
```

Inheritance diagram for activemq::commands::ConnectionId:

Public Types

- typedef **decaf::lang::PointerComparator** < ConnectionId > **COMPARATOR**

Public Member Functions

- **ConnectionId** ()

- **ConnectionId** (const **ConnectionId** &other)
- **ConnectionId** (const **SessionId** *sessionId)
- **ConnectionId** (const **ProducerId** *producerId)
- **ConnectionId** (const **ConsumerId** *consumerId)
- virtual ~**ConnectionId** ()
- virtual unsigned char **getDataStructureType** () const
*Get the **DataStructure** (p. 1133) Type as defined in CommandTypes.h.*
- virtual **ConnectionId** * **cloneDataStructure** () const
Clone this object and return a new instance that the caller now owns, this will be an exact copy of this one.
- virtual void **copyDataStructure** (const **DataStructure** *src)
Copy the contents of the passed object into this objects members, overwriting any existing data.
- virtual std::string **toString** () const
*Returns a string containing the information for this **DataStructure** (p. 1133) such as its type and value of its elements.*
- virtual bool **equals** (const **DataStructure** *value) const
*Compares the **DataStructure** (p. 1133) passed in to this one, and returns if they are equivalent.*
- virtual const std::string & **getValue** () const
- virtual std::string & **getValue** ()
- virtual void **setValue** (const std::string &value)
- virtual int **compareTo** (const **ConnectionId** &value) const
- virtual bool **equals** (const **ConnectionId** &value) const
- virtual bool **operator==** (const **ConnectionId** &value) const
- virtual bool **operator<** (const **ConnectionId** &value) const
- **ConnectionId** & **operator=** (const **ConnectionId** &other)

Static Public Attributes

- static const unsigned char **ID_CONNECTIONID** = 120

Protected Attributes

- std::string **value**

6.153.1 Member Typedef Documentation

- 6.153.1.1 typedef decaf::lang::PointerComparator<ConnectionId>
 activemq::commands::ConnectionId::COMPARATOR

6.153.2 Constructor & Destructor Documentation

- 6.153.2.1 activemq::commands::ConnectionId::ConnectionId ()

6.153.2.2 **activemq::commands::ConnectionId::ConnectionId** (**const** **ConnectionId** & *other*)

6.153.2.3 **activemq::commands::ConnectionId::ConnectionId** (**const** **SessionId** * *sessionId*)

6.153.2.4 **activemq::commands::ConnectionId::ConnectionId** (**const** **ProducerId** * *producerId*)

6.153.2.5 **activemq::commands::ConnectionId::ConnectionId** (**const** **ConsumerId** * *consumerId*)

6.153.2.6 **virtual** **activemq::commands::ConnectionId::~~ConnectionId** ()
[virtual]

6.153.3 Member Function Documentation

6.153.3.1 **virtual** **ConnectionId*** **activemq::commands::ConnectionId::cloneData-**
Structure () **const** [virtual]

Clone this object and return a new instance that the caller now owns, this will be an exact copy of this one.

Returns

new copy of this object.

Implements **activemq::commands::DataStructure** (p. 1133).

6.153.3.2 **virtual** **int** **activemq::commands::ConnectionId::compareTo** (**const** **ConnectionId** & *value*) **const** [virtual]

6.153.3.3 **virtual** **void** **activemq::commands::ConnectionId::copyDataStructure** (**const** **DataStructure** * *src*) [virtual]

Copy the contents of the passed object into this objects members, overwriting any existing data.

Returns

src - Source Object

Implements **activemq::commands::DataStructure** (p. 1134).

6.153.3.4 **virtual** **bool** **activemq::commands::ConnectionId::equals** (**const** **DataStructure** * *value*) **const** [virtual]

Compares the **DataStructure** (p. 1133) passed in to this one, and returns if they are equivalent.

Equivalent here means that they are of the same type, and that each element of the objects are the same.

Returns

true if **DataSet** (p. 1133)'s are Equal.

Implements **activemq::commands::DataSet** (p. 1135).

6.153.3.5 virtual bool **activemq::commands::ConnectionId::equals** (const **ConnectionId** & *value*) const [virtual]

6.153.3.6 virtual unsigned char **activemq::commands::ConnectionId::getData-StructureType** () const [virtual]

Get the **DataSet** (p. 1133) Type as defined in CommandTypes.h.

Returns

The type of the data structure

Implements **activemq::commands::DataSet** (p. 1137).

6.153.3.7 virtual const std::string& **activemq::commands::ConnectionId::getValue** () const [virtual]

6.153.3.8 virtual std::string& **activemq::commands::ConnectionId::getValue** () [virtual]

6.153.3.9 virtual bool **activemq::commands::ConnectionId::operator<** (const **ConnectionId** & *value*) const [virtual]

6.153.3.10 **ConnectionId**& **activemq::commands::ConnectionId::operator=** (const **ConnectionId** & *other*)

6.153.3.11 virtual bool **activemq::commands::ConnectionId::operator==** (const **ConnectionId** & *value*) const [virtual]

6.153.3.12 virtual void **activemq::commands::ConnectionId::setValue** (const std::string & *value*) [virtual]

6.153.3.13 virtual std::string **activemq::commands::ConnectionId::toString** () const [virtual]

Returns a string containing the information for this **DataSet** (p. 1133) such as its type and value of its elements.

Returns

formatted string useful for debugging.

Reimplemented from **activemq::commands::BaseDataStructure** (p. 531).

6.153.4 Field Documentation

6.153.4.1 `const unsigned char activemq::commands::ConnectionId::ID_CONNECTI-
ONID = 120` `[static]`

Referenced by `activemq::state::CommandVisitorAdapter::processRemoveInfo()`.

6.153.4.2 `std::string activemq::commands::ConnectionId::value` `[protected]`

The documentation for this class was generated from the following file:

- `src/main/activemq/commands/ConnectionId.h`

**6.154 activemq::wireformat::openwire::marshal::generated:-
ConnectionIdMarshaller Class Reference**

Marshaling code for Open Wire Format for **ConnectionIdMarshaller** (p. 963).

```
#include <src/main/activemq/wireformat/openwire/marshal/generated/-  
ConnectionIdMarshaller.h>
```

Inheritance diagram for `activemq::wireformat::openwire::marshal::generated:-
ConnectionIdMarshaller`:

Public Member Functions

- **ConnectionIdMarshaller** ()
- virtual **~ConnectionIdMarshaller** ()
- virtual **commands::DataStructure * createObject** () const
Creates a new instance of the class that this class is a marshaling director for.
- virtual unsigned char **getDataStructureType** () const
Gets the DataStructureType that this class marshals/unmarshals.
- virtual void **tightUnmarshal** (**OpenWireFormat *wireFormat**, **commands:-
DataStructure *dataStructure**, **decaf::io::DataInputStream *dataIn**, **utils:-
BooleanStream *bs**)
Tight Un-marhsal to the given stream.

6.154

activemq::wireformat::openwire::marshal::generated::ConnectionIdMarshaller

Class Reference

965

- virtual int **tightMarshal1** (**OpenWireFormat** *wireFormat, **commands::Data-Structure** *dataStructure, **utils::BooleanStream** *bs)

Tight Marshal to the given stream.

- virtual void **tightMarshal2** (**OpenWireFormat** *wireFormat, **commands::Data-Structure** *dataStructure, **decaf::io::DataOutputStream** *dataOut, **utils::BooleanStream** *bs)

Tight Marshal to the given stream.

- virtual void **looseUnmarshal** (**OpenWireFormat** *wireFormat, **commands::Data-Structure** *dataStructure, **decaf::io::DataInputStream** *dataIn)

Loose Un-marshal to the given stream.

- virtual void **looseMarshal** (**OpenWireFormat** *wireFormat, **commands::Data-Structure** *dataStructure, **decaf::io::DataOutputStream** *dataOut)

Tight Marshal to the given stream.

6.154.1 Detailed Description

Marshaling code for Open Wire Format for **ConnectionIdMarshaller** (p. 963).

NOTE!: This file is auto generated - do not modify! if you need to make a change, please see the Java Classes in the activemq-openwire-generator module

6.154.2 Constructor & Destructor Documentation

6.154.2.1 **activemq::wireformat::openwire::marshal::generated-
::ConnectionIdMarshaller::ConnectionIdMarshaller ()**
[inline]

6.154.2.2 **virtual activemq::wireformat::openwire::marshal::generated-
::ConnectionIdMarshaller::~~ConnectionIdMarshaller ()** [inline,
virtual]

6.154.3 Member Function Documentation

6.154.3.1 **virtual commands::DataStructure* activemq::wireformat::openwire-
::marshal::generated::ConnectionIdMarshaller::createObject ()** const
[virtual]

Creates a new instance of the class that this class is a marshaling director for.

Returns

newly allocated Command

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller**
(p. 1120).

6.154.3.2 `virtual unsigned char activemq::wireformat::openwire::marshal-
::generated::ConnectionIdMarshaller::getDataStructureType () const
[virtual]`

Gets the DataStructureType that this class marshals/unmarshals.

Returns

byte Id of this classes DataStructureType

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller**
(p. 1122).

6.154.3.3 `virtual void activemq::wireformat::openwire::marshal::generated::-
ConnectionIdMarshaller::looseMarshal (OpenWireFormat * format,
commands::DataStructure * command, decaf::io::DataOutputStream * ds
) [virtual]`

Tight Marshal to the given stream.

Parameters

<i>format</i>	- The OpenWireFormat properties
<i>command</i>	- the object to Marshal
<i>ds</i>	- DataOutputStream to marshal to

Exceptions

<i>IOException</i>	if an error occurs.
--------------------	---------------------

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller**
(p. 1123).

6.154.3.4 `virtual void activemq::wireformat::openwire::marshal::generated::-
ConnectionIdMarshaller::looseUnmarshal (OpenWireFormat * format,
commands::DataStructure * command, decaf::io::DataInputStream * dis)
[virtual]`

Loose Un-marshal to the given stream.

Parameters

<i>format</i>	- The OpenWireFormat properties
<i>command</i>	- the object to Un-Marshal
<i>dis</i>	- the DataInputStream to Un-Marshal from

6.154

activemq::wireformat::openwire::marshal::generated::ConnectionIdMarshaller**Class Reference****967****Exceptions**

<i>IOException</i>	if an error occurs.
--------------------	---------------------

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller**
(p. 1125).

6.154.3.5 `virtual int activemq::wireformat::openwire::marshal::generated::-
ConnectionIdMarshaller::tightMarshal1 (OpenWireFormat * format,
commands::DataStructure * command, utils::BooleanStream * bs)
[virtual]`

Tight Marshal to the given stream.

Parameters

<i>format</i>	The OpenWireFormat properties
<i>command</i>	The object to Marshal
<i>bs</i>	The boolean stream to marshal to.

Exceptions

<i>IOException</i>	if an error occurs.
--------------------	---------------------

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller**
(p. 1127).

6.154.3.6 `virtual void activemq::wireformat::openwire::marshal::generated::-
ConnectionIdMarshaller::tightMarshal2 (OpenWireFormat * format,
commands::DataStructure * command, decaf::io::DataOutputStream *
ds, utils::BooleanStream * bs) [virtual]`

Tight Marshal to the given stream.

Parameters

<i>format</i>	- The OpenWireFormat properties
<i>command</i>	- the object to Marshal
<i>ds</i>	- the DataOutputStream to Marshal to
<i>bs</i>	- boolean stream to marshal to.

Exceptions

<i>IOException</i>	if an error occurs.
--------------------	---------------------

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller**
(p. 1129).

6.154.3.7 virtual void **activemq::wireformat::openwire::marshal::generated::ConnectionIdMarshaller::tightUnmarshal** (**OpenWireFormat** * *format*, **commands::DataStructure** * *command*, **decaf::io::DataInputStream** * *dis*, **utils::BooleanStream** * *bs*) [virtual]

Tight Un-marhsal to the given stream.

Parameters

<i>format</i>	- The OpenWireFormat properties
<i>command</i>	- the object to Un-Marshal
<i>dis</i>	- the DataInputStream to Un-Marshal from
<i>bs</i>	- boolean stream to unmarshal from.

Exceptions

<i>IOException</i>	if an error occurs.
--------------------	---------------------

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1131).

The documentation for this class was generated from the following file:

- src/main/activemq/wireformat/openwire/marshal/generated/**ConnectionIdMarshaller.h**

6.155 **activemq::commands::ConnectionInfo** Class Reference

```
#include <src/main/activemq/commands/ConnectionInfo.h>
```

Inheritance diagram for **activemq::commands::ConnectionInfo**:

Public Member Functions

- **ConnectionInfo** ()
- virtual **~ConnectionInfo** ()
- virtual unsigned char **getDataStructureType** () const
*Get the **DataStructure** (p. 1133) Type as defined in CommandTypes.h.*
- virtual **ConnectionInfo** * **cloneDataStructure** () const
Clone this obbect and return a new instance that the caller now owns, this will be an exact copy of this one.
- virtual void **copyDataStructure** (const **DataStructure** *src)
Copy the contents of the passed object into this objects members, overwriting any existing data.
- virtual std::string **toString** () const

Returns a string containing the information for this **DataSet** (p. 1133) such as its type and value of its elements.

- virtual bool **equals** (const **DataSet** *value) const
Compares the DataSet (p. 1133) passed in to this one, and returns if they are equivalent.
- **Pointer**< **RemoveInfo** > **createRemoveCommand** () const
- virtual const **Pointer** < **ConnectionId** > & **getConnectionId** () const
- virtual **Pointer**< **ConnectionId** > & **getConnectionId** ()
- virtual void **setConnectionId** (const **Pointer**< **ConnectionId** > &connectionId)
- virtual const std::string & **getClientId** () const
- virtual std::string & **getClientId** ()
- virtual void **setClientId** (const std::string &clientId)
- virtual const std::string & **getPassword** () const
- virtual std::string & **getPassword** ()
- virtual void **setPassword** (const std::string &password)
- virtual const std::string & **getUserName** () const
- virtual std::string & **getUserName** ()
- virtual void **setUserName** (const std::string &userName)
- virtual const std::vector < **decaf::lang::Pointer** < **BrokerId** > > & **getBrokerPath** () const
- virtual std::vector < **decaf::lang::Pointer** < **BrokerId** > > & **getBrokerPath** ()
- virtual void **setBrokerPath** (const std::vector< **decaf::lang::Pointer**< **BrokerId** > > &brokerPath)
- virtual bool **isBrokerMasterConnector** () const
- virtual void **setBrokerMasterConnector** (bool brokerMasterConnector)
- virtual bool **isManageable** () const
- virtual void **setManageable** (bool manageable)
- virtual bool **isClientMaster** () const
- virtual void **setClientMaster** (bool clientMaster)
- virtual bool **isFaultTolerant** () const
- virtual void **setFaultTolerant** (bool faultTolerant)
- virtual bool **isFailoverReconnect** () const
- virtual void **setFailoverReconnect** (bool failoverReconnect)
- virtual bool **isConnectionInfo** () const
- virtual **Pointer**< **Command** > **visit** (activemq::state::CommandVisitor *visitor)

Allows a Visitor to visit this command and return a response to the command based on the command type being visited.

Static Public Attributes

- static const unsigned char **ID_CONNECTIONINFO** = 3

Protected Attributes

- **Pointer**< **ConnectionId** > **connectionId**
- std::string **clientId**
- std::string **password**
- std::string **userName**
- std::vector < **decaf::lang::Pointer** < **BrokerId** > > **brokerPath**
- bool **brokerMasterConnector**
- bool **manageable**
- bool **clientMaster**
- bool **faultTolerant**
- bool **failoverReconnect**

6.155.1 Constructor & Destructor Documentation

6.155.1.1 **activemq::commands::ConnectionInfo::ConnectionInfo** ()

6.155.1.2 **virtual activemq::commands::ConnectionInfo::~~ConnectionInfo** ()
[virtual]

6.155.2 Member Function Documentation

6.155.2.1 **virtual ConnectionInfo* activemq::commands::ConnectionInfo::clone-DataStructure** () const [virtual]

Clone this object and return a new instance that the caller now owns, this will be an exact copy of this one.

Returns

new copy of this object.

Implements **activemq::commands::DataStructure** (p. 1133).

6.155.2.2 **virtual void activemq::commands::ConnectionInfo::copyDataStructure** (**const DataStructure * src**) [virtual]

Copy the contents of the passed object into this objects members, overwriting any existing data.

Returns

src - Source Object

Reimplemented from **activemq::commands::BaseCommand** (p. 493).

6.155.2.3 **Pointer<RemoveInfo>** **activemq::commands::ConnectionInfo::createRemoveCommand** () const

6.155.2.4 **virtual bool** **activemq::commands::ConnectionInfo::equals** (const **DataSet** * *value*) const [virtual]

Compares the **DataSet** (p. 1133) passed in to this one, and returns if they are equivalent.

Equivalent here means that they are of the same type, and that each element of the objects are the same.

Returns

true if **DataSet** (p. 1133)'s are Equal.

Reimplemented from **activemq::commands::BaseCommand** (p. 494).

6.155.2.5 **virtual const std::vector< decaf::lang::Pointer<BrokerId> >&** **activemq::commands::ConnectionInfo::getBrokerPath** () const [virtual]

6.155.2.6 **virtual std::vector< decaf::lang::Pointer<BrokerId> >&** **activemq::commands::ConnectionInfo::getBrokerPath** () [virtual]

6.155.2.7 **virtual const std::string&** **activemq::commands::ConnectionInfo::getClientId** () const [virtual]

6.155.2.8 **virtual std::string&** **activemq::commands::ConnectionInfo::getClientId** () [virtual]

6.155.2.9 **virtual const Pointer<ConnectionId>&** **activemq::commands::ConnectionInfo::getConnectionId** () const [virtual]

6.155.2.10 **virtual Pointer<ConnectionId>&** **activemq::commands::ConnectionInfo::getConnectionId** () [virtual]

6.155.2.11 **virtual unsigned char** **activemq::commands::ConnectionInfo::getDataSetType** () const [virtual]

Get the **DataSet** (p. 1133) Type as defined in CommandTypes.h.

Returns

The type of the data structure

Implements **activemq::commands::DataSet** (p. 1137).

- 6.155.2.12 `virtual const std::string& activemq::commands::ConnectionInfo::getPassword () const [virtual]`
- 6.155.2.13 `virtual std::string& activemq::commands::ConnectionInfo::getPassword () [virtual]`
- 6.155.2.14 `virtual const std::string& activemq::commands::ConnectionInfo::getUserName () const [virtual]`
- 6.155.2.15 `virtual std::string& activemq::commands::ConnectionInfo::getUserName () [virtual]`
- 6.155.2.16 `virtual bool activemq::commands::ConnectionInfo::isBrokerMasterConnector () const [virtual]`
- 6.155.2.17 `virtual bool activemq::commands::ConnectionInfo::isClientMaster () const [virtual]`
- 6.155.2.18 `virtual bool activemq::commands::ConnectionInfo::isConnectionInfo () const [inline, virtual]`

Returns

an answer of true to the **isConnectionInfo()** (p. 971) query.

Reimplemented from **activemq::commands::BaseCommand** (p. 495).

- 6.155.2.19 `virtual bool activemq::commands::ConnectionInfo::isFailoverReconnect () const [virtual]`
- 6.155.2.20 `virtual bool activemq::commands::ConnectionInfo::isFaultTolerant () const [virtual]`
- 6.155.2.21 `virtual bool activemq::commands::ConnectionInfo::isManageable () const [virtual]`
- 6.155.2.22 `virtual void activemq::commands::ConnectionInfo::setBrokerMasterConnector (bool brokerMasterConnector) [virtual]`
- 6.155.2.23 `virtual void activemq::commands::ConnectionInfo::setBrokerPath (const std::vector< decaf::lang::Pointer< BrokerId > > & brokerPath) [virtual]`
- 6.155.2.24 `virtual void activemq::commands::ConnectionInfo::setClientId (const std::string & clientId) [virtual]`

- 6.155.2.25 `virtual void activemq::commands::ConnectionInfo::setClientMaster (bool clientMaster) [virtual]`
- 6.155.2.26 `virtual void activemq::commands::ConnectionInfo::setConnectionId (const Pointer< ConnectionId > & connectionId) [virtual]`
- 6.155.2.27 `virtual void activemq::commands::ConnectionInfo::setFailoverReconnect (bool failoverReconnect) [virtual]`
- 6.155.2.28 `virtual void activemq::commands::ConnectionInfo::setFaultTolerant (bool faultTolerant) [virtual]`
- 6.155.2.29 `virtual void activemq::commands::ConnectionInfo::setManageable (bool manageable) [virtual]`
- 6.155.2.30 `virtual void activemq::commands::ConnectionInfo::setPassword (const std::string & password) [virtual]`
- 6.155.2.31 `virtual void activemq::commands::ConnectionInfo::setUserName (const std::string & userName) [virtual]`
- 6.155.2.32 `virtual std::string activemq::commands::ConnectionInfo::toString () const [virtual]`

Returns a string containing the information for this **DataStructure** (p. 1133) such as its type and value of its elements.

Returns

formatted string useful for debugging.

Reimplemented from **activemq::commands::BaseCommand** (p. 498).

- 6.155.2.33 `virtual Pointer<Command> activemq::commands::ConnectionInfo::visit (activemq::state::CommandVisitor * visitor) [virtual]`

Allows a Visitor to visit this command and return a response to the command based on the command type being visited.

The command will call the proper processXXX method in the visitor.

Returns

a **Response** (p. 2298) to the visitor being called or NULL if no response.

Implements **activemq::commands::Command** (p. 871).

6.155.3 Field Documentation

- 6.155.3.1 `bool activemq::commands::ConnectionInfo::brokerMasterConnector`
[protected]
- 6.155.3.2 `std::vector< decaf::lang::Pointer<BrokerId> >`
`activemq::commands::ConnectionInfo::brokerPath` [protected]
- 6.155.3.3 `std::string activemq::commands::ConnectionInfo::clientId`
[protected]
- 6.155.3.4 `bool activemq::commands::ConnectionInfo::clientMaster`
[protected]
- 6.155.3.5 `Pointer<ConnectionId> activemq::commands::ConnectionInfo::connectionId` [protected]
- 6.155.3.6 `bool activemq::commands::ConnectionInfo::failoverReconnect`
[protected]
- 6.155.3.7 `bool activemq::commands::ConnectionInfo::faultTolerant`
[protected]
- 6.155.3.8 `const unsigned char activemq::commands::ConnectionInfo::ID_CONNECTIONINFO = 3` [static]
- 6.155.3.9 `bool activemq::commands::ConnectionInfo::manageable`
[protected]
- 6.155.3.10 `std::string activemq::commands::ConnectionInfo::password`
[protected]
- 6.155.3.11 `std::string activemq::commands::ConnectionInfo::userName`
[protected]

The documentation for this class was generated from the following file:

- `src/main/activemq/commands/ConnectionInfo.h`

6.156 `activemq::wireformat::openwire::marshal::generated::-` `ConnectionInfoMarshaller` Class Reference

Marshaling code for Open Wire Format for `ConnectionInfoMarshaller` (p. 974).

```
#include <src/main/activemq/wireformat/openwire/marshal/generated/-
ConnectionInfoMarshaller.h>
```

6.156

activemq::wireformat::openwire::marshal::generated::ConnectionInfoMarshaller
Class Reference 975

Inheritance diagram for **activemq::wireformat::openwire::marshal::generated::ConnectionInfoMarshaller**:
ConnectionInfoMarshaller:

Public Member Functions

- **ConnectionInfoMarshaller** ()
- virtual **~ConnectionInfoMarshaller** ()
- virtual **commands::DataStructure * createObject** () const
Creates a new instance of the class that this class is a marshaling director for.
- virtual unsigned char **getDataStructureType** () const
Gets the DataStructureType that this class marshals/unmarshals.
- virtual void **tightUnmarshal** (**OpenWireFormat *wireFormat**, **commands::DataStructure *dataStructure**, **decaf::io::DataInputStream *dataIn**, **utils::BooleanStream *bs**)
Tight Un-marhsal to the given stream.
- virtual int **tightMarshal1** (**OpenWireFormat *wireFormat**, **commands::DataStructure *dataStructure**, **utils::BooleanStream *bs**)
Tight Marhsal to the given stream.
- virtual void **tightMarshal2** (**OpenWireFormat *wireFormat**, **commands::DataStructure *dataStructure**, **decaf::io::DataOutputStream *dataOut**, **utils::BooleanStream *bs**)
Tight Marhsal to the given stream.
- virtual void **looseUnmarshal** (**OpenWireFormat *wireFormat**, **commands::DataStructure *dataStructure**, **decaf::io::DataInputStream *dataIn**)
Loose Un-marhsal to the given stream.
- virtual void **looseMarshal** (**OpenWireFormat *wireFormat**, **commands::DataStructure *dataStructure**, **decaf::io::DataOutputStream *dataOut**)
Tight Marhsal to the given stream.

6.156.1 Detailed Description

Marshaling code for Open Wire Format for **ConnectionInfoMarshaller** (p. 974).

NOTE!: This file is auto generated - do not modify! if you need to make a change, please see the Java Classes in the activemq-openwire-generator module

6.156.2 Constructor & Destructor Documentation

6.156.2.1 **activemq::wireformat::openwire::marshal::generated::ConnectionInfoMarshaller::ConnectionInfoMarshaller** ()
[inline]

6.156.2.2 `virtual activemq::wireformat::openwire::marshal::generated::-
ConnectionInfoMarshaller::~~ConnectionInfoMarshaller ()
[inline, virtual]`

6.156.3 Member Function Documentation

6.156.3.1 `virtual commands::DataStructure* activemq::wireformat::openwire-
::marshal::generated::ConnectionInfoMarshaller::createObject () const
[virtual]`

Creates a new instance of the class that this class is a marshaling director for.

Returns

newly allocated Command

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller**
(p. 1120).

6.156.3.2 `virtual unsigned char activemq::wireformat::openwire::marshal-
::generated::ConnectionInfoMarshaller::getDataStructureType () const
[virtual]`

Gets the DataStructureType that this class marshals/unmarshals.

Returns

byte Id of this classes DataStructureType

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller**
(p. 1122).

6.156.3.3 `virtual void activemq::wireformat::openwire::marshal::generated::-
ConnectionInfoMarshaller::looseMarshal (OpenWireFormat * format,
commands::DataStructure * command, decaf::io::DataOutputStream * ds
) [virtual]`

Tight Marshal to the given stream.

Parameters

<i>format</i>	- The OpenWireFormat properties
<i>command</i>	- the object to Marshal
<i>ds</i>	- DataOutputStream to marshal to

Exceptions

<i>IOException</i>	if an error occurs.
--------------------	---------------------

6.156

activemq::wireformat::openwire::marshal::generated::ConnectionInfoMarshaller
Class Reference 977

Reimplemented from **activemq::wireformat::openwire::marshal::generated::BaseCommandMarshaller** (p. 500).

6.156.3.4 virtual void **activemq::wireformat::openwire::marshal::generated::ConnectionInfoMarshaller::looseUnmarshal** (**OpenWireFormat** * *format*,
commands::DataStructure * *command*, **decaf::io::DataInputStream** * *dis*)
[virtual]

Loose Un-marhsal to the given stream.

Parameters

<i>format</i>	- The OpenWireFormat properties
<i>command</i>	- the object to Un-Marshal
<i>dis</i>	- the DataInputStream to Un-Marshal from

Exceptions

<i>IOException</i>	if an error occurs.
--------------------	---------------------

Reimplemented from **activemq::wireformat::openwire::marshal::generated::BaseCommandMarshaller** (p. 501).

6.156.3.5 virtual int **activemq::wireformat::openwire::marshal::generated::ConnectionInfoMarshaller::tightMarshal1** (**OpenWireFormat** * *format*,
commands::DataStructure * *command*, **utils::BooleanStream** * *bs*)
[virtual]

Tight Marhsal to the given stream.

Parameters

<i>format</i>	The OpenWireFormat properties
<i>command</i>	The object to Marshal
<i>bs</i>	The boolean stream to marshal to.

Exceptions

<i>IOException</i>	if an error occurs.
--------------------	---------------------

Reimplemented from **activemq::wireformat::openwire::marshal::generated::BaseCommandMarshaller** (p. 503).

6.156.3.6 `virtual void activemq::wireformat::openwire::marshal::generated::ConnectionInfoMarshaller::tightMarshal2 (OpenWireFormat * format, commands::DataStructure * command, decaf::io::DataOutputStream * ds, utils::BooleanStream * bs) [virtual]`

Tight Marshal to the given stream.

Parameters

<i>format</i>	- The OpenWireFormat properties
<i>command</i>	- the object to Marshal
<i>ds</i>	- the DataOutputStream to Marshal to
<i>bs</i>	- boolean stream to marshal to.

Exceptions

<i>IOException</i>	if an error occurs.
--------------------	---------------------

Reimplemented from **activemq::wireformat::openwire::marshal::generated::Base-CommandMarshaller** (p. 504).

6.156.3.7 `virtual void activemq::wireformat::openwire::marshal::generated::ConnectionInfoMarshaller::tightUnmarshal (OpenWireFormat * format, commands::DataStructure * command, decaf::io::DataInputStream * dis, utils::BooleanStream * bs) [virtual]`

Tight Un-marshal to the given stream.

Parameters

<i>format</i>	- The OpenWireFormat properties
<i>command</i>	- the object to Un-Marshal
<i>dis</i>	- the DataInputStream to Un-Marshal from
<i>bs</i>	- boolean stream to unmarshal from.

Exceptions

<i>IOException</i>	if an error occurs.
--------------------	---------------------

Reimplemented from **activemq::wireformat::openwire::marshal::generated::Base-CommandMarshaller** (p. 505).

The documentation for this class was generated from the following file:

- `src/main/activemq/wireformat/openwire/marshal/generated/ConnectionInfo-Marshaller.h`

6.157 cms::ConnectionMetaData Class Reference

A **ConnectionMetaData** (p.978) object provides information describing the - **Connection** (p.933) object.

```
#include <src/main/cms/ConnectionMetaData.h>
```

Inheritance diagram for cms::ConnectionMetaData:

Public Member Functions

- virtual **~ConnectionMetaData** () throw ()
- virtual std::string **getCMSVersion** () const =0
Gets the CMS API version.
- virtual int **getCMSMajorVersion** () const =0
Gets the CMS major version number.
- virtual int **getCMSMinorVersion** () const =0
Gets the CMS minor version number.
- virtual std::string **getCMSProviderName** () const =0
Gets the CMS provider name.
- virtual std::string **getProviderVersion** () const =0
Gets the CMS provider version.
- virtual int **getProviderMajorVersion** () const =0
Gets the CMS provider major version number.
- virtual int **getProviderMinorVersion** () const =0
Gets the CMS provider minor version number.
- virtual std::vector< std::string > **getCMSXPropertyNames** () const =0
Gets an Vector of the CMSX property names.

6.157.1 Detailed Description

A **ConnectionMetaData** (p.978) object provides information describing the - **Connection** (p.933) object.

Since

1.3

6.157.2 Constructor & Destructor Documentation

6.157.2.1 virtual cms::ConnectionMetaData::~~ConnectionMetaData () throw ()
[virtual]

6.157.3 Member Function Documentation

6.157.3.1 `virtual int cms::ConnectionMetaData::getCMSMajorVersion () const`
`[pure virtual]`

Gets the CMS major version number.

Returns

the CMS API major version number

Exceptions

<i>CMSException</i> (p. 826)	If the CMS Provider fails to retrieve the metadata due to some internal error.
--	--

Implemented in `activemq::core::ActiveMQConnectionMetaData` (p. 228).

6.157.3.2 `virtual int cms::ConnectionMetaData::getCMSMinorVersion () const`
`[pure virtual]`

Gets the CMS minor version number.

Returns

the CMS API minor version number

Exceptions

<i>CMSException</i> (p. 826)	If the CMS Provider fails to retrieve the metadata due to some internal error.
--	--

Implemented in `activemq::core::ActiveMQConnectionMetaData` (p. 228).

6.157.3.3 `virtual std::string cms::ConnectionMetaData::getCMSProviderName ()`
`const [pure virtual]`

Gets the CMS provider name.

Returns

the CMS provider name

Exceptions

<i>CMSException</i> (p. 826)	If the CMS Provider fails to retrieve the metadata due to some internal error.
--	--

Implemented in **activemq::core::ActiveMQConnectionMetaData** (p. 228).

6.157.3.4 `virtual std::string cms::ConnectionMetaData::getCMSVersion () const`
`[pure virtual]`

Gets the CMS API version.

Returns

the CMS API Version in String form.

Exceptions

<i>CMSEException</i> (p. 826)	If the CMS Provider fails to retrieve the metadata due to some internal error.
---	--

Implemented in **activemq::core::ActiveMQConnectionMetaData** (p. 229).

6.157.3.5 `virtual std::vector<std::string> cms::ConnectionMetaData::getCMSXPropertyNames () const`
`[pure virtual]`

Gets an Vector of the CMSX property names.

Returns

an Vector of CMSX property names

Exceptions

<i>CMSEException</i> (p. 826)	If the CMS Provider fails to retrieve the metadata due to some internal error.
---	--

Implemented in **activemq::core::ActiveMQConnectionMetaData** (p. 229).

6.157.3.6 `virtual int cms::ConnectionMetaData::getProviderMajorVersion () const`
`[pure virtual]`

Gets the CMS provider major version number.

Returns

the CMS provider major version number

Exceptions

<i>CMSEException</i> (p. 826)	If the CMS Provider fails to retrieve the metadata due to some internal error.
---	--

Implemented in **activemq::core::ActiveMQConnectionMetaData** (p. 230).

6.157.3.7 `virtual int cms::ConnectionMetaData::getProviderMinorVersion () const`
`[pure virtual]`

Gets the CMS provider minor version number.

Returns

the CMS provider minor version number

Exceptions

CMSException (p. 826)	If the CMS Provider fails to retrieve the metadata due to some internal error.
---------------------------------	--

Implemented in **activemq::core::ActiveMQConnectionMetaData** (p. 230).

6.157.3.8 `virtual std::string cms::ConnectionMetaData::getProviderVersion () const`
`[pure virtual]`

Gets the CMS provider version.

Returns

the CMS provider version

Exceptions

CMSException (p. 826)	If the CMS Provider fails to retrieve the metadata due to some internal error.
---------------------------------	--

Implemented in **activemq::core::ActiveMQConnectionMetaData** (p. 230).

The documentation for this class was generated from the following file:

- `src/main/cms/ConnectionMetaData.h`

6.158 activemq::state::ConnectionState Class Reference

```
#include <src/main/activemq/state/ConnectionState.h>
```

Public Member Functions

- **ConnectionState** (const **Pointer**< **ConnectionInfo** > &info)

- virtual `~ConnectionState ()`
- `std::string toString () const`
- `const Pointer < commands::ConnectionInfo > & getInfo () const`
- `void checkShutdown () const`
- `void shutdown ()`
- `void reset (const Pointer< ConnectionInfo > &info)`
- `void addTempDestination (const Pointer< DestinationInfo > &info)`
- `void removeTempDestination (const Pointer< ActiveMQDestination > &destination)`
- `void addTransactionState (const Pointer< TransactionId > &id)`
- `const Pointer< TransactionState > & getTransactionState (const Pointer< TransactionId > &id) const`
- `std::vector< Pointer < TransactionState > > & getTransactionStates () const`
- `Pointer< TransactionState > removeTransactionState (const Pointer< TransactionId > &id)`
- `void addSession (const Pointer< SessionInfo > &info)`
- `Pointer< SessionState > removeSession (const Pointer< SessionId > &id)`
- `const Pointer< SessionState > & getSessionState (const Pointer< SessionId > &id) const`
- `const LinkedList< Pointer < DestinationInfo > > & getTempDesinations () const`
- `std::vector< Pointer < SessionState > > & getSessionStates () const`
- `StlMap< Pointer< ConsumerId > , Pointer< ConsumerInfo > , ConsumerId::COMPARATOR > & getRecoveringPullConsumers ()`
- `void setConnectionInterruptProcessingComplete (bool connectionInterruptProcessingComplete)`
- `bool isConnectionInterruptProcessingComplete ()`

6.158.1 Constructor & Destructor Documentation

6.158.1.1 `activemq::state::ConnectionState::ConnectionState (const Pointer< ConnectionInfo > & info)`

6.158.1.2 `virtual activemq::state::ConnectionState::~~ConnectionState ()`
[virtual]

6.158.2 Member Function Documentation

6.158.2.1 `void activemq::state::ConnectionState::addSession (const Pointer< SessionInfo > & info)` [inline]

References `activemq::commands::SessionInfo::getSessionId()`.

- 6.158.2.2 `void activemq::state::ConnectionState::addTempDestination (const
Pointer< DestinationInfo > & info) [inline]`
- 6.158.2.3 `void activemq::state::ConnectionState::addTransactionState (const
Pointer< TransactionId > & id) [inline]`
- 6.158.2.4 `void activemq::state::ConnectionState::checkShutdown () const`
- 6.158.2.5 `const Pointer<commands::ConnectionInfo>&
activemq::state::ConnectionState::getInfo () const [inline]`
- 6.158.2.6 `StlMap< Pointer<ConsumerId>, Pointer<ConsumerInfo>,
ConsumerId::COMPARATOR >& activemq::state::-
ConnectionState::getRecoveringPullConsumers ()
[inline]`
- 6.158.2.7 `const Pointer<SessionState>& activemq::state::Connection-
State::getSessionState (const Pointer< SessionId > & id) const
[inline]`
- 6.158.2.8 `std::vector< Pointer<SessionState> > activemq-
::state::ConnectionState::getSessionStates () const
[inline]`
- 6.158.2.9 `const LinkedList< Pointer<DestinationInfo> >&
activemq::state::ConnectionState::getTempDesinations () const
[inline]`
- 6.158.2.10 `const Pointer<TransactionState>& activemq::state::ConnectionState-
::getTransactionState (const Pointer< TransactionId > & id) const
[inline]`

References `decaf::lang::Pointer< T, REFCOUNTER >::get()`.

- 6.158.2.11 `std::vector< Pointer<TransactionState> > activemq-
::state::ConnectionState::getTransactionStates () const
[inline]`
- 6.158.2.12 `bool activemq::state::ConnectionState::isConnectionInterrupt-
ProcessingComplete () [inline]`
- 6.158.2.13 `Pointer<SessionState> activemq::state::Connection-
State::removeSession (const Pointer< SessionId > & id)
[inline]`

6.158.2.14 `void activemq::state::ConnectionState::removeTempDestination (const Pointer< ActiveMQDestination > & destination) [inline]`

References `decaf::lang::Pointer< T, REFCOUNTER >::get()`, and `activemq::commands::DestinationInfo::getDestination()`.

6.158.2.15 `Pointer<TransactionState> activemq::state::ConnectionState::removeTransactionState (const Pointer< TransactionId > & id) [inline]`

6.158.2.16 `void activemq::state::ConnectionState::reset (const Pointer< ConnectionInfo > & info)`

6.158.2.17 `void activemq::state::ConnectionState::setConnectionInterruptProcessingComplete (bool connectionInterruptProcessingComplete) [inline]`

6.158.2.18 `void activemq::state::ConnectionState::shutdown ()`

6.158.2.19 `std::string activemq::state::ConnectionState::toString () const`

The documentation for this class was generated from the following file:

- `src/main/activemq/state/ConnectionState.h`

6.159 activemq::state::ConnectionStateTracker Class Reference

```
#include <src/main/activemq/state/ConnectionStateTracker.h>
```

Inheritance diagram for `activemq::state::ConnectionStateTracker`:

Public Member Functions

- `ConnectionStateTracker ()`
- `virtual ~ConnectionStateTracker ()`
- `Pointer< Tracked > track (const Pointer< Command > &command)`
- `void trackBack (const Pointer< Command > &command)`
- `void restore (const Pointer< transport::Transport > &transport)`
- `void connectionInterruptProcessingComplete (transport::Transport *transport, const Pointer< ConnectionId > &connectionId)`
- `void transportInterrupted ()`
- `virtual Pointer< Command > processDestinationInfo (DestinationInfo *info)`

- virtual **Pointer< Command > processRemoveDestination** (**DestinationInfo** *info)
- virtual **Pointer< Command > processProducerInfo** (**ProducerInfo** *info)
- virtual **Pointer< Command > processRemoveProducer** (**ProducerId** *id)
- virtual **Pointer< Command > processConsumerInfo** (**ConsumerInfo** *info)
- virtual **Pointer< Command > processRemoveConsumer** (**ConsumerId** *id)
- virtual **Pointer< Command > processSessionInfo** (**SessionInfo** *info)
- virtual **Pointer< Command > processRemoveSession** (**SessionId** *id)
- virtual **Pointer< Command > processConnectionInfo** (**ConnectionInfo** *info)
- virtual **Pointer< Command > processRemoveConnection** (**ConnectionId** *id)
- virtual **Pointer< Command > processMessage** (**Message** *message)
- virtual **Pointer< Command > processMessageAck** (**MessageAck** *ack)
- virtual **Pointer< Command > processBeginTransaction** (**TransactionInfo** *info)
- virtual **Pointer< Command > processPrepareTransaction** (**TransactionInfo** *info)
- virtual **Pointer< Command > processCommitTransactionOnePhase** (- **TransactionInfo** *info)
- virtual **Pointer< Command > processCommitTransactionTwoPhase** (- **TransactionInfo** *info)
- virtual **Pointer< Command > processRollbackTransaction** (**TransactionInfo** *info)
- virtual **Pointer< Command > processEndTransaction** (**TransactionInfo** *info)
- bool **isRestoreConsumers** () const
- void **setRestoreConsumers** (bool restoreConsumers)
- bool **isRestoreProducers** () const
- void **setRestoreProducers** (bool restoreProducers)
- bool **isRestoreSessions** () const
- void **setRestoreSessions** (bool restoreSessions)
- bool **isTrackTransactions** () const
- void **setTrackTransactions** (bool trackTransactions)
- bool **isRestoreTransaction** () const
- void **setRestoreTransaction** (bool restoreTransaction)
- bool **isTrackMessages** () const
- void **setTrackMessages** (bool trackMessages)
- int **getMaxCacheSize** () const
- void **setMaxCacheSize** (int maxCacheSize)
- bool **isTrackTransactionProducers** () const
- void **setTrackTransactionProducers** (bool trackTransactionProducers)

Friends

- class **RemoveTransactionAction**

6.159.1 Constructor & Destructor Documentation

6.159.1.1 **activemq::state::ConnectionStateTracker::ConnectionStateTracker** ()

6.159.1.2 **virtual activemq::state::ConnectionStateTracker::~~ConnectionStateTracker** () [virtual]

6.159.2 Member Function Documentation

6.159.2.1 **void activemq::state::ConnectionStateTracker::connectionInterruptProcessingComplete** (transport::Transport * *transport*, const Pointer< ConnectionId > & *connectionId*)

6.159.2.2 **int activemq::state::ConnectionStateTracker::getMaxCacheSize** () const [inline]

6.159.2.3 **bool activemq::state::ConnectionStateTracker::isRestoreConsumers** () const [inline]

6.159.2.4 **bool activemq::state::ConnectionStateTracker::isRestoreProducers** () const [inline]

6.159.2.5 **bool activemq::state::ConnectionStateTracker::isRestoreSessions** () const [inline]

6.159.2.6 **bool activemq::state::ConnectionStateTracker::isRestoreTransaction** () const [inline]

6.159.2.7 **bool activemq::state::ConnectionStateTracker::isTrackMessages** () const [inline]

6.159.2.8 **bool activemq::state::ConnectionStateTracker::isTrackTransactionProducers** () const [inline]

6.159.2.9 **bool activemq::state::ConnectionStateTracker::isTrackTransactions** () const [inline]

6.159.2.10 **virtual Pointer<Command> activemq::state::ConnectionStateTracker::processBeginTransaction** (TransactionInfo * *info*) [virtual]

Implements **activemq::state::CommandVisitor** (p. 874).

6.159.2.11 **virtual Pointer<Command> activemq::state::ConnectionStateTracker::processCommitTransactionOnePhase** (TransactionInfo * *info*) [virtual]

Implements **activemq::state::CommandVisitor** (p. 874).

6.159.2.12 **virtual Pointer<Command> activemq::state::ConnectionStateTracker::processCommitTransactionTwoPhase (TransactionInfo * info)**
[virtual]

Implements **activemq::state::CommandVisitor** (p. 874).

6.159.2.13 **virtual Pointer<Command> activemq::state::ConnectionStateTracker::processConnectionInfo (ConnectionInfo * info)**
[virtual]

Implements **activemq::state::CommandVisitor** (p. 875).

6.159.2.14 **virtual Pointer<Command> activemq::state::ConnectionStateTracker::processConsumerInfo (ConsumerInfo * info)**
[virtual]

Implements **activemq::state::CommandVisitor** (p. 875).

6.159.2.15 **virtual Pointer<Command> activemq::state::ConnectionStateTracker::processDestinationInfo (DestinationInfo * info)**
[virtual]

Implements **activemq::state::CommandVisitor** (p. 875).

6.159.2.16 **virtual Pointer<Command> activemq::state::ConnectionStateTracker::processEndTransaction (TransactionInfo * info)**
[virtual]

Implements **activemq::state::CommandVisitor** (p. 875).

6.159.2.17 **virtual Pointer<Command> activemq::state::ConnectionStateTracker::processMessage (Message * message)**
[virtual]

Implements **activemq::state::CommandVisitor** (p. 876).

6.159.2.18 **virtual Pointer<Command> activemq::state::ConnectionStateTracker::processMessageAck (MessageAck * ack)**
[virtual]

Implements **activemq::state::CommandVisitor** (p. 876).

6.159.2.19 `virtual Pointer<Command> activemq::state::ConnectionStateTracker::processPrepareTransaction (TransactionInfo * info)`
[virtual]

Implements `activemq::state::CommandVisitor` (p. 876).

6.159.2.20 `virtual Pointer<Command> activemq::state::ConnectionStateTracker::processProducerInfo (ProducerInfo * info)`
[virtual]

Implements `activemq::state::CommandVisitor` (p. 877).

6.159.2.21 `virtual Pointer<Command> activemq::state::ConnectionStateTracker::processRemoveConnection (ConnectionId * id)`
[virtual]

Implements `activemq::state::CommandVisitor` (p. 877).

6.159.2.22 `virtual Pointer<Command> activemq::state::ConnectionStateTracker::processRemoveConsumer (ConsumerId * id)`
[virtual]

Implements `activemq::state::CommandVisitor` (p. 877).

6.159.2.23 `virtual Pointer<Command> activemq::state::ConnectionStateTracker::processRemoveDestination (DestinationInfo * info)`
[virtual]

Implements `activemq::state::CommandVisitor` (p. 877).

6.159.2.24 `virtual Pointer<Command> activemq::state::ConnectionStateTracker::processRemoveProducer (ProducerId * id)`
[virtual]

Implements `activemq::state::CommandVisitor` (p. 877).

6.159.2.25 `virtual Pointer<Command> activemq::state::ConnectionStateTracker::processRemoveSession (SessionId * id)`
[virtual]

Implements `activemq::state::CommandVisitor` (p. 877).

6.159.2.26 `virtual Pointer<Command> activemq::state::ConnectionStateTracker::processRollbackTransaction (TransactionInfo * info)`
[virtual]

Implements **activemq::state::CommandVisitor** (p. 878).

6.159.2.27 `virtual Pointer<Command> activemq::state::ConnectionStateTracker::processSessionInfo (SessionInfo * info)`
[virtual]

Implements **activemq::state::CommandVisitor** (p. 878).

6.159.2.28 `void activemq::state::ConnectionStateTracker::restore (const Pointer<transport::Transport> & transport)`

6.159.2.29 `void activemq::state::ConnectionStateTracker::setMaxCacheSize (int maxCacheSize)` [inline]

6.159.2.30 `void activemq::state::ConnectionStateTracker::setRestoreConsumers (bool restoreConsumers)` [inline]

6.159.2.31 `void activemq::state::ConnectionStateTracker::setRestoreProducers (bool restoreProducers)` [inline]

6.159.2.32 `void activemq::state::ConnectionStateTracker::setRestoreSessions (bool restoreSessions)` [inline]

6.159.2.33 `void activemq::state::ConnectionStateTracker::setRestoreTransaction (bool restoreTransaction)` [inline]

6.159.2.34 `void activemq::state::ConnectionStateTracker::setTrackMessages (bool trackMessages)` [inline]

6.159.2.35 `void activemq::state::ConnectionStateTracker::setTrackTransactionProducers (bool trackTransactionProducers)`
[inline]

6.159.2.36 `void activemq::state::ConnectionStateTracker::setTrackTransactions (bool trackTransactions)` [inline]

6.159.2.37 `Pointer<Tracked> activemq::state::ConnectionStateTracker::track (const Pointer<Command> & command)`

6.159.2.38 `void activemq::state::ConnectionStateTracker::trackBack (const Pointer<Command> & command)`

6.159.2.39 `void activemq::state::ConnectionStateTracker::transportInterrupted ()`

6.159.3 Friends And Related Function Documentation

6.159.3.1 friend class **RemoveTransactionAction** [friend]

The documentation for this class was generated from the following file:

- src/main/activemq/state/**ConnectionStateTracker.h**

6.160 decaf::util::logging::ConsoleHandler Class Reference

This **Handler** (p. 1401) publishes log records to System.err.

```
#include <src/main/decaf/util/logging/ConsoleHandler.h>
```

Inheritance diagram for decaf::util::logging::ConsoleHandler:

Public Member Functions

- **ConsoleHandler** ()
- virtual \sim **ConsoleHandler** ()
- virtual void **close** ()
Close the current output stream.
- virtual void **publish** (const **LogRecord** &record)
*Publish the Log Record to this **Handler** (p. 1401).*

6.160.1 Detailed Description

This **Handler** (p. 1401) publishes log records to System.err.

By default the **SimpleFormatter** (p. 2441) is used to generate brief summaries.

Configuration: By default each **ConsoleHandler** (p. 990) is initialized using the following **LogManager** (p. 1712) configuration properties. If properties are not defined (or have invalid values) then the specified default values are used.

ConsoleHandler.level specifies the default level for the **Handler** (p. 1401) (defaults to **Level.INFO** (p. 1620)). ConsoleHandler.filter specifies the name of a **Filter** (p. 1333) class to use (defaults to no **Filter** (p. 1333)). ConsoleHandler.formatter specifies the name of a **Formatter** (p. 1387) class to use (defaults to **SimpleFormatter** (p. 2441)).

Since

1.0

6.160.2 Constructor & Destructor Documentation

6.160.2.1 `decaf::util::logging::ConsoleHandler::ConsoleHandler ()`

6.160.2.2 `virtual decaf::util::logging::ConsoleHandler::~~ConsoleHandler ()`
[inline, virtual]

6.160.3 Member Function Documentation

6.160.3.1 `virtual void decaf::util::logging::ConsoleHandler::close ()` [virtual]

Close the current output stream.

Override the **StreamHandler** (p.2603) close to flush the Std Err stream but doesn't close.

Exceptions

<i>IOException</i>

Reimplemented from **decaf::util::logging::StreamHandler** (p. 2604).

6.160.3.2 `virtual void decaf::util::logging::ConsoleHandler::publish (const LogRecord & record)` [virtual]

Publish the Log Record to this **Handler** (p. 1401).

Parameters

<i>record</i>	The LogRecord (p.1719) to Publish
---------------	--

Reimplemented from **decaf::util::logging::StreamHandler** (p. 2605).

The documentation for this class was generated from the following file:

- `src/main/decaf/util/logging/ConsoleHandler.h`

6.161 activemq::commands::ConsumerControl Class Reference

```
#include <src/main/activemq/commands/ConsumerControl.h>
```

Inheritance diagram for `activemq::commands::ConsumerControl`:

Public Member Functions

- **ConsumerControl ()**

- virtual **~ConsumerControl** ()
- virtual unsigned char **getDataStructureType** () const
*Get the **DataSetructure** (p. 1133) Type as defined in CommandTypes.h.*
- virtual **ConsumerControl** * **cloneDataSetructure** () const
Clone this object and return a new instance that the caller now owns, this will be an exact copy of this one.
- virtual void **copyDataSetructure** (const **DataSetructure** *src)
Copy the contents of the passed object into this objects members, overwriting any existing data.
- virtual std::string **toString** () const
*Returns a string containing the information for this **DataSetructure** (p. 1133) such as its type and value of its elements.*
- virtual bool **equals** (const **DataSetructure** *value) const
*Compares the **DataSetructure** (p. 1133) passed in to this one, and returns if they are equivalent.*
- virtual const **Pointer** < **ActiveMQDestination** > & **getDestination** () const
- virtual **Pointer** < **ActiveMQDestination** > & **getDestination** ()
- virtual void **setDestination** (const **Pointer**< **ActiveMQDestination** > &destination)
- virtual bool **isClose** () const
- virtual void **setClose** (bool close)
- virtual const **Pointer** < **ConsumerId** > & **getConsumerId** () const
- virtual **Pointer**< **ConsumerId** > & **getConsumerId** ()
- virtual void **setConsumerId** (const **Pointer**< **ConsumerId** > &consumerId)
- virtual int **getPrefetch** () const
- virtual void **setPrefetch** (int prefetch)
- virtual bool **isFlush** () const
- virtual void **setFlush** (bool flush)
- virtual bool **isStart** () const
- virtual void **setStart** (bool start)
- virtual bool **isStop** () const
- virtual void **setStop** (bool stop)
- virtual **Pointer**< **Command** > **visit** (activemq::state::CommandVisitor *visitor)
Allows a Visitor to visit this command and return a response to the command based on the command type being visited.

Static Public Attributes

- static const unsigned char **ID_CONSUMERCONTROL** = 17

Protected Attributes

- **Pointer**< **ActiveMQDestination** > **destination**
- **bool** **close**
- **Pointer**< **ConsumerId** > **consumerId**
- **int** **prefetch**
- **bool** **flush**
- **bool** **start**
- **bool** **stop**

6.161.1 Constructor & Destructor Documentation

6.161.1.1 **activemq::commands::ConsumerControl::ConsumerControl** ()

6.161.1.2 **virtual activemq::commands::ConsumerControl::~~ConsumerControl** ()
[virtual]

6.161.2 Member Function Documentation

6.161.2.1 **virtual ConsumerControl* activemq::commands::ConsumerControl::cloneDataStructure** () const
[virtual]

Clone this obbject and return a new instance that the caller now owns, this will be an exact copy of this one.

Returns

new copy of this object.

Implements **activemq::commands::DataStructure** (p. 1133).

6.161.2.2 **virtual void activemq::commands::ConsumerControl::copyDataStructure**
(**const DataStructure * src**) [virtual]

Copy the contents of the passed object into this objects members, overwriting any existing data.

Returns

src - Source Object

Reimplemented from **activemq::commands::BaseCommand** (p. 493).

6.161.2.3 `virtual bool activemq::commands::ConsumerControl::equals (const DataStructure * value) const [virtual]`

Compares the **DataStructure** (p. 1133) passed in to this one, and returns if they are equivalent.

Equivalent here means that they are of the same type, and that each element of the objects are the same.

Returns

true if **DataStructure** (p. 1133)'s are Equal.

Reimplemented from **activemq::commands::BaseCommand** (p. 494).

6.161.2.4 `virtual const Pointer<ConsumerId>& activemq::commands::ConsumerControl::getConsumerId () const [virtual]`

6.161.2.5 `virtual Pointer<ConsumerId>& activemq::commands::ConsumerControl::getConsumerId () [virtual]`

6.161.2.6 `virtual unsigned char activemq::commands::ConsumerControl::getDataStructureType () const [virtual]`

Get the **DataStructure** (p. 1133) Type as defined in CommandTypes.h.

Returns

The type of the data structure

Implements **activemq::commands::DataStructure** (p. 1137).

6.161.2.7 `virtual const Pointer<ActiveMQDestination>& activemq::commands::ConsumerControl::getDestination () const [virtual]`

6.161.2.8 `virtual Pointer<ActiveMQDestination>& activemq::commands::ConsumerControl::getDestination () [virtual]`

6.161.2.9 `virtual int activemq::commands::ConsumerControl::getPrefetch () const [virtual]`

6.161.2.10 `virtual bool activemq::commands::ConsumerControl::isClose () const [virtual]`

6.161.2.11 `virtual bool activemq::commands::ConsumerControl::isFlush () const [virtual]`

- 6.161.2.12 `virtual bool activemq::commands::ConsumerControl::isStart () const`
[virtual]
- 6.161.2.13 `virtual bool activemq::commands::ConsumerControl::isStop () const`
[virtual]
- 6.161.2.14 `virtual void activemq::commands::ConsumerControl::setClose (bool`
`close) [virtual]`
- 6.161.2.15 `virtual void activemq::commands::ConsumerControl::setConsumerId (`
`const Pointer< ConsumerId > & consumerId) [virtual]`
- 6.161.2.16 `virtual void activemq::commands::ConsumerControl::setDestination (`
`const Pointer< ActiveMQDestination > & destination) [virtual]`
- 6.161.2.17 `virtual void activemq::commands::ConsumerControl::setFlush (bool`
`flush) [virtual]`
- 6.161.2.18 `virtual void activemq::commands::ConsumerControl::setPrefetch (int`
`prefetch) [virtual]`
- 6.161.2.19 `virtual void activemq::commands::ConsumerControl::setStart (bool start`
`) [virtual]`
- 6.161.2.20 `virtual void activemq::commands::ConsumerControl::setStop (bool stop`
`) [virtual]`
- 6.161.2.21 `virtual std::string activemq::commands::ConsumerControl::toString ()`
`const [virtual]`

Returns a string containing the information for this **DataStructure** (p. 1133) such as its type and value of its elements.

Returns

formatted string useful for debugging.

Reimplemented from **activemq::commands::BaseCommand** (p. 498).

- 6.161.2.22 `virtual Pointer<Command> activemq::commands::Consumer-`
`Control::visit (activemq::state::CommandVisitor * visitor)`
[virtual]

Allows a Visitor to visit this command and return a response to the command based on the command type being visited.

The command will call the proper processXXX method in the visitor.

6.162 activemq::wireformat::openwire::marshal::generated::ConsumerControlMarshaller Class

Reference

997

Returns

a **Response** (p. 2298) to the visitor being called or NULL if no response.

Implements **activemq::commands::Command** (p. 871).

6.161.3 Field Documentation

6.161.3.1 **bool** **activemq::commands::ConsumerControl::close** [protected]

6.161.3.2 **Pointer<ConsumerId>** **activemq::commands::ConsumerControl::consumerId** [protected]

6.161.3.3 **Pointer<ActiveMQDestination>** **activemq::commands::ConsumerControl::destination** [protected]

6.161.3.4 **bool** **activemq::commands::ConsumerControl::flush** [protected]

6.161.3.5 **const unsigned char** **activemq::commands::ConsumerControl::ID_CONSUMERCONTROL = 17** [static]

6.161.3.6 **int** **activemq::commands::ConsumerControl::prefetch** [protected]

6.161.3.7 **bool** **activemq::commands::ConsumerControl::start** [protected]

6.161.3.8 **bool** **activemq::commands::ConsumerControl::stop** [protected]

The documentation for this class was generated from the following file:

- src/main/activemq/commands/**ConsumerControl.h**

6.162 activemq::wireformat::openwire::marshal::generated::ConsumerControlMarshaller Class Reference

Marshaling code for Open Wire Format for **ConsumerControlMarshaller** (p. 996).

```
#include <src/main/activemq/wireformat/openwire/marshal/generated/-  
ConsumerControlMarshaller.h>
```

Inheritance diagram for **activemq::wireformat::openwire::marshal::generated::ConsumerControlMarshaller**:

Public Member Functions

- **ConsumerControlMarshaller** ()

- virtual **~ConsumerControlMarshaller** ()
- virtual **commands::DataStructure * createObject** () const
Creates a new instance of the class that this class is a marshaling director for.
- virtual unsigned char **getDataStructureType** () const
Gets the DataStructureType that this class marshals/unmarshals.
- virtual void **tightUnmarshal** (**OpenWireFormat *wireFormat**, **commands::DataStructure *dataStructure**, **decaf::io::DataInputStream *dataIn**, **utils::BooleanStream *bs**)
Tight Un-marhsal to the given stream.
- virtual int **tightMarshal1** (**OpenWireFormat *wireFormat**, **commands::DataStructure *dataStructure**, **utils::BooleanStream *bs**)
Tight Marhsal to the given stream.
- virtual void **tightMarshal2** (**OpenWireFormat *wireFormat**, **commands::DataStructure *dataStructure**, **decaf::io::DataOutputStream *dataOut**, **utils::BooleanStream *bs**)
Tight Marhsal to the given stream.
- virtual void **looseUnmarshal** (**OpenWireFormat *wireFormat**, **commands::DataStructure *dataStructure**, **decaf::io::DataInputStream *dataIn**)
Loose Un-marhsal to the given stream.
- virtual void **looseMarshal** (**OpenWireFormat *wireFormat**, **commands::DataStructure *dataStructure**, **decaf::io::DataOutputStream *dataOut**)
Tight Marhsal to the given stream.

6.162.1 Detailed Description

Marshaling code for Open Wire Format for **ConsumerControlMarshaller** (p. 996).

NOTE!: This file is auto generated - do not modify! if you need to make a change, please see the Java Classes in the activemq-openwire-generator module

6.162.2 Constructor & Destructor Documentation

- 6.162.2.1 **activemq::wireformat::openwire::marshal::generated::ConsumerControlMarshaller::ConsumerControlMarshaller** ()
[inline]
- 6.162.2.2 **virtual activemq::wireformat::openwire::marshal::generated::ConsumerControlMarshaller::~~ConsumerControlMarshaller** () [inline, virtual]

6.162.3 Member Function Documentation

- 6.162.3.1 **virtual commands::DataStructure* activemq::wireformat::openwire::marshal::generated::ConsumerControlMarshaller::createObject** ()
const [virtual]

Creates a new instance of the class that this class is a marshaling director for.

6.162 activemq::wireformat::openwire::marshal::generated::ConsumerControl-Marshaller Class

Reference

999

Returns

newly allocated Command

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1120).

6.162.3.2 `virtual unsigned char activemq::wireformat::openwire::marshal::generated::ConsumerControlMarshaller::getDataStructureType () const`
[virtual]

Gets the DataStructureType that this class marshals/unmarshals.

Returns

byte Id of this classes DataStructureType

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1122).

6.162.3.3 `virtual void activemq::wireformat::openwire::marshal::generated::ConsumerControlMarshaller::looseMarshal (OpenWireFormat * format, commands::DataStructure * command, decaf::io::DataOutputStream * ds)` [virtual]

Tight Marshal to the given stream.

Parameters

<i>format</i>	- The OpenWireFormat properties
<i>command</i>	- the object to Marshal
<i>ds</i>	- DataOutputStream to marshal to

Exceptions

<i>IOException</i>	if an error occurs.
--------------------	---------------------

Reimplemented from **activemq::wireformat::openwire::marshal::generated::Base-CommandMarshaller** (p. 500).

6.162.3.4 `virtual void activemq::wireformat::openwire::marshal::generated::ConsumerControlMarshaller::looseUnmarshal (OpenWireFormat * format, commands::DataStructure * command, decaf::io::DataInputStream * dis)` [virtual]

Loose Un-marshal to the given stream.

Parameters

<i>format</i>	- The OpenwireFormat properties
<i>command</i>	- the object to Un-Marshal
<i>dis</i>	- the DataInputStream to Un-Marshal from

Exceptions

<i>IOException</i>	if an error occurs.
--------------------	---------------------

Reimplemented from **activemq::wireformat::openwire::marshal::generated::Base-CommandMarshaller** (p. 501).

```
6.162.3.5 virtual int activemq::wireformat::openwire::marshal::generated::-
ConsumerControlMarshaller::tightMarshal1 ( OpenWireFormat * format,
commands::DataStructure * command, utils::BooleanStream * bs )
[virtual]
```

Tight Marshal to the given stream.

Parameters

<i>format</i>	The OpenwireFormat properties
<i>command</i>	The object to Marshal
<i>bs</i>	The boolean stream to marshal to.

Exceptions

<i>IOException</i>	if an error occurs.
--------------------	---------------------

Reimplemented from **activemq::wireformat::openwire::marshal::generated::Base-CommandMarshaller** (p. 503).

```
6.162.3.6 virtual void activemq::wireformat::openwire::marshal::generated::-
ConsumerControlMarshaller::tightMarshal2 ( OpenWireFormat * format,
commands::DataStructure * command, decaf::io::DataOutputStream *
ds, utils::BooleanStream * bs ) [virtual]
```

Tight Marshal to the given stream.

Parameters

<i>format</i>	- The OpenwireFormat properties
<i>command</i>	- the object to Marshal
<i>ds</i>	- the DataOutputStream to Marshal to
<i>bs</i>	- boolean stream to marshal to.

Exceptions

<i>IOException</i>	if an error occurs.
--------------------	---------------------

Reimplemented from **activemq::wireformat::openwire::marshal::generated::Base-CommandMarshaller** (p. 504).

6.162.3.7 virtual void **activemq::wireformat::openwire::marshal::generated::ConsumerControlMarshaller::tightUnmarshal** (
OpenWireFormat * *format*, **commands::DataStructure** * *command*,
decaf::io::DataInputStream * *dis*, **utils::BooleanStream** * *bs*)
 [virtual]

Tight Un-marhsal to the given stream.

Parameters

<i>format</i>	- The OpenWireFormat properties
<i>command</i>	- the object to Un-Marshal
<i>dis</i>	- the DataInputStream to Un-Marshal from
<i>bs</i>	- boolean stream to unmarshal from.

Exceptions

<i>IOException</i>	if an error occurs.
--------------------	---------------------

Reimplemented from **activemq::wireformat::openwire::marshal::generated::Base-CommandMarshaller** (p. 505).

The documentation for this class was generated from the following file:

- src/main/activemq/wireformat/openwire/marshal/generated/**ConsumerControl-Marshaller.h**

6.163 activemq::commands::ConsumerId Class Reference

```
#include <src/main/activemq/commands/ConsumerId.h>
```

Inheritance diagram for **activemq::commands::ConsumerId**:

Public Types

- typedef **decaf::lang::PointerComparator** < **ConsumerId** > **COMPARATOR**

Public Member Functions

- **ConsumerId** ()
- **ConsumerId** (const **ConsumerId** &other)
- **ConsumerId** (const **SessionId** &sessionId, long long consumerId)
- virtual ~**ConsumerId** ()
- virtual unsigned char **getDataStructureType** () const
*Get the **DataStructure** (p. 1133) Type as defined in CommandTypes.h.*
- virtual **ConsumerId** * **cloneDataStructure** () const
Clone this object and return a new instance that the caller now owns, this will be an exact copy of this one.
- virtual void **copyDataStructure** (const **DataStructure** *src)
Copy the contents of the passed object into this objects members, overwriting any existing data.
- virtual std::string **toString** () const
*Returns a string containing the information for this **DataStructure** (p. 1133) such as its type and value of its elements.*
- virtual bool **equals** (const **DataStructure** *value) const
*Compares the **DataStructure** (p. 1133) passed in to this one, and returns if they are equivalent.*
- const **Pointer**< **SessionId** > & **getParentId** () const
- virtual const std::string & **getConnectionId** () const
- virtual std::string & **getConnectionId** ()
- virtual void **setConnectionId** (const std::string &connectionId)
- virtual long long **getSessionId** () const
- virtual void **setSessionId** (long long sessionId)
- virtual long long **getValue** () const
- virtual void **setValue** (long long value)
- virtual int **compareTo** (const **ConsumerId** &value) const
- virtual bool **equals** (const **ConsumerId** &value) const
- virtual bool **operator==** (const **ConsumerId** &value) const
- virtual bool **operator<** (const **ConsumerId** &value) const
- **ConsumerId** & **operator=** (const **ConsumerId** &other)

Static Public Attributes

- static const unsigned char **ID_CONSUMERID** = 122

Protected Attributes

- std::string **connectionId**
- long long **sessionId**
- long long **value**

6.163.1 Member Typedef Documentation

6.163.1.1 `typedef decaf::lang::PointerComparator<ConsumerId>
activemq::commands::ConsumerId::COMPARATOR`

6.163.2 Constructor & Destructor Documentation

6.163.2.1 `activemq::commands::ConsumerId::ConsumerId ()`

6.163.2.2 `activemq::commands::ConsumerId::ConsumerId (const ConsumerId &
other)`

6.163.2.3 `activemq::commands::ConsumerId::ConsumerId (const SessionId &
sessionId, long long consumerId)`

6.163.2.4 `virtual activemq::commands::ConsumerId::~~ConsumerId ()
[virtual]`

6.163.3 Member Function Documentation

6.163.3.1 `virtual ConsumerId* activemq::commands::ConsumerId::cloneData-
Structure () const [virtual]`

Clone this object and return a new instance that the caller now owns, this will be an exact copy of this one.

Returns

new copy of this object.

Implements **activemq::commands::DataStructure** (p. 1133).

6.163.3.2 `virtual int activemq::commands::ConsumerId::compareTo (const
ConsumerId & value) const [virtual]`

6.163.3.3 `virtual void activemq::commands::ConsumerId::copyDataStructure (const
DataStructure * src) [virtual]`

Copy the contents of the passed object into this objects members, overwriting any existing data.

Returns

src - Source Object

Implements **activemq::commands::DataStructure** (p. 1134).

6.163.3.4 `virtual bool activemq::commands::ConsumerId::equals (const
DataStructure * value) const [virtual]`

Compares the **DataStructure** (p. 1133) passed in to this one, and returns if they are equivalent.

Equivalent here means that they are of the same type, and that each element of the objects are the same.

Returns

true if **DataStructure** (p. 1133)'s are Equal.

Implements **activemq::commands::DataStructure** (p. 1135).

6.163.3.5 `virtual bool activemq::commands::ConsumerId::equals (const
ConsumerId & value) const [virtual]`

6.163.3.6 `virtual const std::string& activemq::commands::ConsumerId::get-
ConnectionId () const [virtual]`

6.163.3.7 `virtual std::string& activemq::commands::ConsumerId::getConnectionId ()
[virtual]`

6.163.3.8 `virtual unsigned char activemq::commands::ConsumerId::getData-
StructureType () const [virtual]`

Get the **DataStructure** (p. 1133) Type as defined in CommandTypes.h.

Returns

The type of the data structure

Implements **activemq::commands::DataStructure** (p. 1137).

6.163.3.9 `const Pointer<SessionId>& activemq::commands::ConsumerId::get-
ParentId () const`

6.163.3.10 `virtual long long activemq::commands::ConsumerId::getSessionId ()
const [virtual]`

6.163.3.11 `virtual long long activemq::commands::ConsumerId::getValue () const
[virtual]`

6.163.3.12 `virtual bool activemq::commands::ConsumerId::operator< (const ConsumerId &
value) const [virtual]`

6.163.3.13 `ConsumerId& activemq::commands::ConsumerId::operator= (const
ConsumerId & other)`

6.163.3.14 `virtual bool activemq::commands::ConsumerId::operator== (const ConsumerId & value) const` [virtual]

6.163.3.15 `virtual void activemq::commands::ConsumerId::setConnectionId (const std::string & connectionId)` [virtual]

6.163.3.16 `virtual void activemq::commands::ConsumerId::setSessionId (long long sessionId)` [virtual]

6.163.3.17 `virtual void activemq::commands::ConsumerId::setValue (long long value)` [virtual]

6.163.3.18 `virtual std::string activemq::commands::ConsumerId::toString () const` [virtual]

Returns a string containing the information for this **DataSet** (p. 1133) such as its type and value of its elements.

Returns

formatted string useful for debugging.

Reimplemented from **activemq::commands::BaseDataSet** (p. 531).

6.163.4 Field Documentation

6.163.4.1 `std::string activemq::commands::ConsumerId::connectionId` [protected]

6.163.4.2 `const unsigned char activemq::commands::ConsumerId::ID_CONSUMERID = 122` [static]

Referenced by `activemq::state::CommandVisitorAdapter::processRemoveInfo()`.

6.163.4.3 `long long activemq::commands::ConsumerId::sessionId` [protected]

6.163.4.4 `long long activemq::commands::ConsumerId::value` [protected]

The documentation for this class was generated from the following file:

- `src/main/activemq/commands/ConsumerId.h`

6.164 activemq::wireformat::openwire::marshal::generated:- ConsumerIdMarshaller Class Reference

Marshaling code for Open Wire Format for **ConsumerIdMarshaller** (p. 1005).

```
#include <src/main/activemq/wireformat/openwire/marshal/generated/-  
ConsumerIdMarshaller.h>
```

Inheritance diagram for activemq::wireformat::openwire::marshal::generated:-
ConsumerIdMarshaller:

Public Member Functions

- **ConsumerIdMarshaller** ()
- virtual **~ConsumerIdMarshaller** ()
- virtual **commands::DataStructure * createObject** () const
Creates a new instance of the class that this class is a marshaling director for.
- virtual unsigned char **getDataStructureType** () const
Gets the DataStructureType that this class marshals/unmarshals.
- virtual void **tightUnmarshal** (**OpenWireFormat** *wireFormat, **commands:-DataStructure** *dataStructure, **decaf::io::DataInputStream** *dataIn, **utils:-BooleanStream** *bs)
Tight Un-marhsal to the given stream.
- virtual int **tightMarshal1** (**OpenWireFormat** *wireFormat, **commands::Data-Structure** *dataStructure, **utils::BooleanStream** *bs)
Tight Marshal to the given stream.
- virtual void **tightMarshal2** (**OpenWireFormat** *wireFormat, **commands::Data-Structure** *dataStructure, **decaf::io::DataOutputStream** *dataOut, **utils:-BooleanStream** *bs)
Tight Marshal to the given stream.
- virtual void **looseUnmarshal** (**OpenWireFormat** *wireFormat, **commands:-DataStructure** *dataStructure, **decaf::io::DataInputStream** *dataIn)
Loose Un-marhsal to the given stream.
- virtual void **looseMarshal** (**OpenWireFormat** *wireFormat, **commands::Data-Structure** *dataStructure, **decaf::io::DataOutputStream** *dataOut)
Tight Marshal to the given stream.

6.164.1 Detailed Description

Marshaling code for Open Wire Format for **ConsumerIdMarshaller** (p. 1005).

NOTE!: This file is auto generated - do not modify! if you need to make a change, please see the Java Classes in the activemq-openwire-generator module

6.164

activemq::wireformat::openwire::marshal::generated::ConsumerIdMarshaller

Class Reference

1007

6.164.2 Constructor & Destructor Documentation

6.164.2.1 **activemq::wireformat::openwire::marshal::generated-
::ConsumerIdMarshaller::ConsumerIdMarshaller ()**
[inline]

6.164.2.2 **virtual activemq::wireformat::openwire::marshal::generated:-
ConsumerIdMarshaller::~~ConsumerIdMarshaller ()** [inline,
virtual]

6.164.3 Member Function Documentation

6.164.3.1 **virtual commands::DataStructure* activemq::wireformat::openwire-
::marshal::generated::ConsumerIdMarshaller::createObject ()** const
[virtual]

Creates a new instance of the class that this class is a marshaling director for.

Returns

newly allocated Command

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller**
(p. 1120).

6.164.3.2 **virtual unsigned char activemq::wireformat::openwire::marshal-
::generated::ConsumerIdMarshaller::getDataStructureType ()** const
[virtual]

Gets the DataStructureType that this class marshals/unmarshals.

Returns

byte Id of this classes DataStructureType

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller**
(p. 1122).

6.164.3.3 **virtual void activemq::wireformat::openwire::marshal::generated-
::ConsumerIdMarshaller::looseMarshal (OpenWireFormat * *format*,
commands::DataStructure * *command*, decaf::io::DataOutputStream * *ds*
)** [virtual]

Tight Marshal to the given stream.

Parameters

<i>format</i>	- The OpenWireFormat properties
<i>command</i>	- the object to Marshal
<i>ds</i>	- DataOutputStream to marshal to

Generated on Tue Feb 28 2012 17:47:25 for activemq-cpp-3.4.1 by Doxygen

Exceptions

<i>IOException</i>	if an error occurs.
--------------------	---------------------

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1123).

6.164.3.4 **virtual void activemq::wireformat::openwire::marshal::generated::-ConsumerIdMarshaller::looseUnmarshal (OpenWireFormat * *format*, commands::DataStructure * *command*, decaf::io::DataInputStream * *dis*)**
[virtual]

Loose Un-marhsal to the given stream.

Parameters

<i>format</i>	- The OpenWireFormat properties
<i>command</i>	- the object to Un-Marshal
<i>dis</i>	- the DataInputStream to Un-Marshal from

Exceptions

<i>IOException</i>	if an error occurs.
--------------------	---------------------

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1125).

6.164.3.5 **virtual int activemq::wireformat::openwire::marshal::generated::-ConsumerIdMarshaller::tightMarshal1 (OpenWireFormat * *format*, commands::DataStructure * *command*, utils::BooleanStream * *bs*)**
[virtual]

Tight Marhsal to the given stream.

Parameters

<i>format</i>	The OpenWireFormat properties
<i>command</i>	The object to Marshal
<i>bs</i>	The boolean stream to marshal to.

Exceptions

<i>IOException</i>	if an error occurs.
--------------------	---------------------

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1127).

6.164

activemq::wireformat::openwire::marshal::generated::ConsumerIdMarshaller

Class Reference

1009

6.164.3.6 virtual void activemq::wireformat::openwire::marshal::generated-
 ::ConsumerIdMarshaller::tightMarshal2 (OpenWireFormat * *format*,
 commands::DataStructure * *command*, decaf::io::DataOutputStream *
ds, utils::BooleanStream * *bs*) [virtual]

Tight Marhsal to the given stream.

Parameters

<i>format</i>	- The OpenWireFormat properties
<i>command</i>	- the object to Marshal
<i>ds</i>	- the DataOutputStream to Marshal to
<i>bs</i>	- boolean stream to marshal to.

Exceptions

<i>IOException</i>	if an error occurs.
--------------------	---------------------

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller**
 (p. 1129).

6.164.3.7 virtual void activemq::wireformat::openwire::marshal::generated-
 ConsumerIdMarshaller::tightUnmarshal (OpenWireFormat * *format*,
 commands::DataStructure * *command*, decaf::io::DataInputStream * *dis*,
 utils::BooleanStream * *bs*) [virtual]

Tight Un-marhsal to the given stream.

Parameters

<i>format</i>	- The OpenWireFormat properties
<i>command</i>	- the object to Un-Marshal
<i>dis</i>	- the DataInputStream to Un-Marshal from
<i>bs</i>	- boolean stream to unmarshal from.

Exceptions

<i>IOException</i>	if an error occurs.
--------------------	---------------------

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller**
 (p. 1131).

The documentation for this class was generated from the following file:

- src/main/activemq/wireformat/openwire/marshal/generated/**ConsumerId-Marshaller.h**

6.165 activemq::commands::ConsumerInfo Class Reference

```
#include <src/main/activemq/commands/ConsumerInfo.h>
```

Inheritance diagram for activemq::commands::ConsumerInfo:

Public Member Functions

- **ConsumerInfo** ()
- virtual **~ConsumerInfo** ()
- virtual unsigned char **getDataStructureType** () const
*Get the **DataStructure** (p. 1133) Type as defined in CommandTypes.h.*
- virtual **ConsumerInfo * cloneDataStructure** () const
Clone this object and return a new instance that the caller now owns, this will be an exact copy of this one.
- virtual void **copyDataStructure** (const **DataStructure** *src)
Copy the contents of the passed object into this objects members, overwriting any existing data.
- virtual std::string **toString** () const
*Returns a string containing the information for this **DataStructure** (p. 1133) such as its type and value of its elements.*
- virtual bool **equals** (const **DataStructure** *value) const
*Compares the **DataStructure** (p. 1133) passed in to this one, and returns if they are equivalent.*
- **Pointer< RemoveInfo > createRemoveCommand** () const
- virtual const **Pointer < ConsumerId > & getConsumerId** () const
- virtual **Pointer< ConsumerId > & getConsumerId** ()
- virtual void **setConsumerId** (const **Pointer< ConsumerId > &consumerId**)
- virtual bool **isBrowser** () const
- virtual void **setBrowser** (bool browser)
- virtual const **Pointer < ActiveMQDestination > & getDestination** () const
- virtual **Pointer < ActiveMQDestination > & getDestination** ()
- virtual void **setDestination** (const **Pointer< ActiveMQDestination > &destination**)
- virtual int **getPrefetchSize** () const
- virtual void **setPrefetchSize** (int prefetchSize)
- virtual int **getMaximumPendingMessageLimit** () const
- virtual void **setMaximumPendingMessageLimit** (int maximumPendingMessageLimit)
- virtual bool **isDispatchAsync** () const
- virtual void **setDispatchAsync** (bool dispatchAsync)
- virtual const std::string & **getSelector** () const
- virtual std::string & **getSelector** ()
- virtual void **setSelector** (const std::string &selector)

- virtual const std::string & **getSubscriptionName** () const
- virtual std::string & **getSubscriptionName** ()
- virtual void **setSubscriptionName** (const std::string &subscriptionName)
- virtual bool **isNoLocal** () const
- virtual void **setNoLocal** (bool noLocal)
- virtual bool **isExclusive** () const
- virtual void **setExclusive** (bool exclusive)
- virtual bool **isRetroactive** () const
- virtual void **setRetroactive** (bool retroactive)
- virtual unsigned char **getPriority** () const
- virtual void **setPriority** (unsigned char priority)
- virtual const std::vector < **decaf::lang::Pointer** < **BrokerId** > > & **getBrokerPath** () const
- virtual std::vector < **decaf::lang::Pointer** < **BrokerId** > > & **getBrokerPath** ()
- virtual void **setBrokerPath** (const std::vector< **decaf::lang::Pointer**< **BrokerId** > > &brokerPath)
- virtual const **Pointer** < **BooleanExpression** > & **getAdditionalPredicate** () const
- virtual **Pointer** < **BooleanExpression** > & **getAdditionalPredicate** ()
- virtual void **setAdditionalPredicate** (const **Pointer**< **BooleanExpression** > &additionalPredicate)
- virtual bool **isNetworkSubscription** () const
- virtual void **setNetworkSubscription** (bool networkSubscription)
- virtual bool **isOptimizedAcknowledge** () const
- virtual void **setOptimizedAcknowledge** (bool optimizedAcknowledge)
- virtual bool **isNoRangeAcks** () const
- virtual void **setNoRangeAcks** (bool noRangeAcks)
- virtual const std::vector < **decaf::lang::Pointer** < **ConsumerId** > > & **getNetworkConsumerPath** () const
- virtual std::vector < **decaf::lang::Pointer** < **ConsumerId** > > & **getNetworkConsumerPath** ()
- virtual void **setNetworkConsumerPath** (const std::vector< **decaf::lang::Pointer**< **ConsumerId** > > &networkConsumerPath)
- virtual bool **isConsumerInfo** () const
- virtual **Pointer**< **Command** > **visit** (activemq::state::CommandVisitor *visitor)

Allows a Visitor to visit this command and return a response to the command based on the command type being visited.

Static Public Attributes

- static const unsigned char **ID_CONSUMERINFO** = 5

Protected Attributes

- **Pointer**< **ConsumerId** > **consumerId**
- bool **browser**
- **Pointer**< **ActiveMQDestination** > **destination**
- int **prefetchSize**
- int **maximumPendingMessageLimit**
- bool **dispatchAsync**
- std::string **selector**
- std::string **subscriptionName**
- bool **noLocal**
- bool **exclusive**
- bool **retroactive**
- unsigned char **priority**
- std::vector < **decaf::lang::Pointer** < **BrokerId** > > **brokerPath**
- **Pointer**< **BooleanExpression** > **additionalPredicate**
- bool **networkSubscription**
- bool **optimizedAcknowledge**
- bool **noRangeAcks**
- std::vector < **decaf::lang::Pointer** < **ConsumerId** > > **networkConsumerPath**

6.165.1 Constructor & Destructor Documentation

6.165.1.1 **activemq::commands::ConsumerInfo::ConsumerInfo** ()

6.165.1.2 **virtual activemq::commands::ConsumerInfo::~~ConsumerInfo** ()
[virtual]

6.165.2 Member Function Documentation

6.165.2.1 **virtual ConsumerInfo* activemq::commands::ConsumerInfo::cloneDataStructure** () const [virtual]

Clone this obbject and return a new instance that the caller now owns, this will be an exact copy of this one.

Returns

new copy of this object.

Implements **activemq::commands::DataStructure** (p. 1133).

6.165.2.2 virtual void **activemq::commands::ConsumerInfo::copyDataStructure** (
const **DataStructure** * *src*) [virtual]

Copy the contents of the passed object into this objects members, overwriting any existing data.

Returns

src - Source Object

Reimplemented from **activemq::commands::BaseCommand** (p. 493).

6.165.2.3 **Pointer<RemoveInfo> activemq::commands::ConsumerInfo::create-
RemoveCommand** () const

6.165.2.4 virtual bool **activemq::commands::ConsumerInfo::equals** (const
DataStructure * *value*) const [virtual]

Compares the **DataStructure** (p. 1133) passed in to this one, and returns if they are equivalent.

Equivalent here means that they are of the same type, and that each element of the objects are the same.

Returns

true if **DataStructure** (p. 1133)'s are Equal.

Reimplemented from **activemq::commands::BaseCommand** (p. 494).

6.165.2.5 virtual const **Pointer<BooleanExpression>& activemq-
::commands::ConsumerInfo::getAdditionalPredicate** () const
[virtual]

6.165.2.6 virtual **Pointer<BooleanExpression>& activemq-
::commands::ConsumerInfo::getAdditionalPredicate** ()
[virtual]

6.165.2.7 virtual const std::vector< decaf::lang::Pointer<BrokerId> >&
activemq::commands::ConsumerInfo::getBrokerPath () const
[virtual]

6.165.2.8 virtual std::vector< decaf::lang::Pointer<BrokerId> >&
activemq::commands::ConsumerInfo::getBrokerPath () [virtual]

6.165.2.9 virtual const **Pointer<ConsumerId>& activemq-
::commands::ConsumerInfo::getConsumerId** () const
[virtual]

Referenced by **activemq::state::SessionState::addConsumer**().

6.165.2.10 virtual **Pointer**<**ConsumerId**>& **activemq::commands::ConsumerInfo::getConsumerId** () [virtual]

6.165.2.11 virtual unsigned char **activemq::commands::ConsumerInfo::getDataStructureType** () const [virtual]

Get the **DataStructure** (p. 1133) Type as defined in **CommandTypes.h**.

Returns

The type of the data structure

Implements **activemq::commands::DataStructure** (p. 1137).

6.165.2.12 virtual const **Pointer**<**ActiveMQDestination**>& **activemq::commands::ConsumerInfo::getDestination** () const [virtual]

6.165.2.13 virtual **Pointer**<**ActiveMQDestination**>& **activemq::commands::ConsumerInfo::getDestination** () [virtual]

6.165.2.14 virtual int **activemq::commands::ConsumerInfo::getMaximumPendingMessageLimit** () const [virtual]

6.165.2.15 virtual const std::vector< **decaf::lang::Pointer**<**ConsumerId**> >& **activemq::commands::ConsumerInfo::getNetworkConsumerPath** () const [virtual]

6.165.2.16 virtual std::vector< **decaf::lang::Pointer**<**ConsumerId**> >& **activemq::commands::ConsumerInfo::getNetworkConsumerPath** () [virtual]

6.165.2.17 virtual int **activemq::commands::ConsumerInfo::getPrefetchSize** () const [virtual]

6.165.2.18 virtual unsigned char **activemq::commands::ConsumerInfo::getPriority** () const [virtual]

6.165.2.19 virtual const std::string& **activemq::commands::ConsumerInfo::getSelector** () const [virtual]

6.165.2.20 virtual std::string& **activemq::commands::ConsumerInfo::getSelector** () [virtual]

- 6.165.2.21 `virtual const std::string& activemq::commands::ConsumerInfo::getSubscriptionName () const`
[virtual]
- 6.165.2.22 `virtual std::string& activemq::commands::ConsumerInfo::getSubscriptionName ()` [virtual]
- 6.165.2.23 `virtual bool activemq::commands::ConsumerInfo::isBrowser () const`
[virtual]
- 6.165.2.24 `virtual bool activemq::commands::ConsumerInfo::isConsumerInfo () const` [inline, virtual]

Returns

an answer of true to the **isConsumerInfo()** (p. 1014) query.

Reimplemented from **activemq::commands::BaseCommand** (p. 495).

- 6.165.2.25 `virtual bool activemq::commands::ConsumerInfo::isDispatchAsync () const` [virtual]
- 6.165.2.26 `virtual bool activemq::commands::ConsumerInfo::isExclusive () const`
[virtual]
- 6.165.2.27 `virtual bool activemq::commands::ConsumerInfo::isNetworkSubscription () const` [virtual]
- 6.165.2.28 `virtual bool activemq::commands::ConsumerInfo::isNoLocal () const`
[virtual]
- 6.165.2.29 `virtual bool activemq::commands::ConsumerInfo::isNoRangeAcks () const` [virtual]
- 6.165.2.30 `virtual bool activemq::commands::ConsumerInfo::isOptimizedAcknowledge () const` [virtual]
- 6.165.2.31 `virtual bool activemq::commands::ConsumerInfo::isRetroactive () const`
[virtual]
- 6.165.2.32 `virtual void activemq::commands::ConsumerInfo::setAdditionalPredicate (const Pointer< BooleanExpression > & additionalPredicate)`
[virtual]
- 6.165.2.33 `virtual void activemq::commands::ConsumerInfo::setBrokerPath (const std::vector< decaf::lang::Pointer< BrokerId > > & brokerPath)`
[virtual]

- 6.165.2.34 `virtual void activemq::commands::ConsumerInfo::setBrowser (bool browser) [virtual]`
- 6.165.2.35 `virtual void activemq::commands::ConsumerInfo::setConsumerId (const Pointer< ConsumerId > & consumerId) [virtual]`
- 6.165.2.36 `virtual void activemq::commands::ConsumerInfo::setDestination (const Pointer< ActiveMQDestination > & destination) [virtual]`
- 6.165.2.37 `virtual void activemq::commands::ConsumerInfo::setDispatchAsync (bool dispatchAsync) [virtual]`
- 6.165.2.38 `virtual void activemq::commands::ConsumerInfo::setExclusive (bool exclusive) [virtual]`
- 6.165.2.39 `virtual void activemq::commands::ConsumerInfo::setMaximum-PendingMessageLimit (int maximumPendingMessageLimit) [virtual]`
- 6.165.2.40 `virtual void activemq::commands::ConsumerInfo::setNetwork-ConsumerPath (const std::vector< decaf::lang::Pointer< ConsumerId > > & networkConsumerPath) [virtual]`
- 6.165.2.41 `virtual void activemq::commands::ConsumerInfo-::setNetworkSubscription (bool networkSubscription) [virtual]`
- 6.165.2.42 `virtual void activemq::commands::ConsumerInfo::setNoLocal (bool noLocal) [virtual]`
- 6.165.2.43 `virtual void activemq::commands::ConsumerInfo::setNoRangeAcks (bool noRangeAcks) [virtual]`
- 6.165.2.44 `virtual void activemq::commands::ConsumerInfo::set-OptimizedAcknowledge (bool optimizedAcknowledge) [virtual]`
- 6.165.2.45 `virtual void activemq::commands::ConsumerInfo::setPrefetchSize (int prefetchSize) [virtual]`
- 6.165.2.46 `virtual void activemq::commands::ConsumerInfo::setPriority (unsigned char priority) [virtual]`
- 6.165.2.47 `virtual void activemq::commands::ConsumerInfo::setRetroactive (bool retroactive) [virtual]`
- 6.165.2.48 `virtual void activemq::commands::ConsumerInfo::setSelector (const std::string & selector) [virtual]`

6.165.2.49 `virtual void activemq::commands::ConsumerInfo::setSubscriptionName (const std::string & subscriptionName) [virtual]`

6.165.2.50 `virtual std::string activemq::commands::ConsumerInfo::toString () const [virtual]`

Returns a string containing the information for this **DataStructure** (p. 1133) such as its type and value of its elements.

Returns

formatted string useful for debugging.

Reimplemented from **activemq::commands::BaseCommand** (p. 498).

6.165.2.51 `virtual Pointer<Command> activemq::commands::ConsumerInfo::visit (activemq::state::CommandVisitor * visitor) [virtual]`

Allows a Visitor to visit this command and return a response to the command based on the command type being visited.

The command will call the proper processXXX method in the visitor.

Returns

a **Response** (p. 2298) to the visitor being called or NULL if no response.

Implements **activemq::commands::Command** (p. 871).

6.165.3 Field Documentation

6.165.3.1 `Pointer<BooleanExpression> activemq::commands::ConsumerInfo::additionalPredicate [protected]`

6.165.3.2 `std::vector< decaf::lang::Pointer<BrokerId> > activemq::commands::ConsumerInfo::brokerPath [protected]`

6.165.3.3 `bool activemq::commands::ConsumerInfo::browser [protected]`

6.165.3.4 `Pointer<ConsumerId> activemq::commands::ConsumerInfo::consumerId [protected]`

6.165.3.5 `Pointer<ActiveMQDestination> activemq::commands::ConsumerInfo::destination [protected]`

6.165.3.6 `bool activemq::commands::ConsumerInfo::dispatchAsync [protected]`

- 6.165.3.7 `bool activemq::commands::ConsumerInfo::exclusive` [protected]
- 6.165.3.8 `const unsigned char activemq::commands::ConsumerInfo::ID_CONSUMERINFO = 5` [static]
- 6.165.3.9 `int activemq::commands::ConsumerInfo::maximumPendingMessageLimit` [protected]
- 6.165.3.10 `std::vector< decaf::lang::Pointer<ConsumerId> > activemq::commands::ConsumerInfo::networkConsumerPath` [protected]
- 6.165.3.11 `bool activemq::commands::ConsumerInfo::networkSubscription` [protected]
- 6.165.3.12 `bool activemq::commands::ConsumerInfo::noLocal` [protected]
- 6.165.3.13 `bool activemq::commands::ConsumerInfo::noRangeAcks` [protected]
- 6.165.3.14 `bool activemq::commands::ConsumerInfo::optimizedAcknowledge` [protected]
- 6.165.3.15 `int activemq::commands::ConsumerInfo::prefetchSize` [protected]
- 6.165.3.16 `unsigned char activemq::commands::ConsumerInfo::priority` [protected]
- 6.165.3.17 `bool activemq::commands::ConsumerInfo::retroactive` [protected]
- 6.165.3.18 `std::string activemq::commands::ConsumerInfo::selector` [protected]
- 6.165.3.19 `std::string activemq::commands::ConsumerInfo::subscriptionName` [protected]

The documentation for this class was generated from the following file:

- `src/main/activemq/commands/ConsumerInfo.h`

6.166 `activemq::wireformat::openwire::marshal::generated::ConsumerInfoMarshaller` Class Reference

Marshaling code for Open Wire Format for `ConsumerInfoMarshaller` (p. 1017).

6.166

activemq::wireformat::openwire::marshal::generated::ConsumerInfoMarshaller

Class Reference

1019

```
#include <src/main/activemq/wireformat/openwire/marshal/generated/-  
ConsumerInfoMarshaller.h>
```

Inheritance diagram for activemq::wireformat::openwire::marshal::generated::ConsumerInfoMarshaller:

Public Member Functions

- **ConsumerInfoMarshaller** ()
- virtual **~ConsumerInfoMarshaller** ()
- virtual **commands::DataStructure * createObject** () const
Creates a new instance of the class that this class is a marshaling director for.
- virtual unsigned char **getDataStructureType** () const
Gets the DataStructureType that this class marshals/unmarshals.
- virtual void **tightUnmarshal** (**OpenWireFormat *wireFormat**, **commands::DataStructure *dataStructure**, **decaf::io::DataInputStream *dataIn**, **utils::BooleanStream *bs**)
Tight Un-marhsal to the given stream.
- virtual int **tightMarshal1** (**OpenWireFormat *wireFormat**, **commands::DataStructure *dataStructure**, **utils::BooleanStream *bs**)
Tight Marhsal to the given stream.
- virtual void **tightMarshal2** (**OpenWireFormat *wireFormat**, **commands::DataStructure *dataStructure**, **decaf::io::DataOutputStream *dataOut**, **utils::BooleanStream *bs**)
Tight Marhsal to the given stream.
- virtual void **looseUnmarshal** (**OpenWireFormat *wireFormat**, **commands::DataStructure *dataStructure**, **decaf::io::DataInputStream *dataIn**)
Loose Un-marhsal to the given stream.
- virtual void **looseMarshal** (**OpenWireFormat *wireFormat**, **commands::DataStructure *dataStructure**, **decaf::io::DataOutputStream *dataOut**)
Tight Marhsal to the given stream.

6.166.1 Detailed Description

Marshaling code for Open Wire Format for **ConsumerInfoMarshaller** (p. 1017).

NOTE!: This file is auto generated - do not modify! if you need to make a change, please see the Java Classes in the activemq-openwire-generator module

6.166.2 Constructor & Destructor Documentation

6.166.2.1 **activemq::wireformat::openwire::marshal::generated::ConsumerInfoMarshaller::ConsumerInfoMarshaller** ()
[inline]

6.166.2.2 `virtual activemq::wireformat::openwire::marshal::generated::ConsumerInfoMarshaller::~~ConsumerInfoMarshaller () [inline, virtual]`

6.166.3 Member Function Documentation

6.166.3.1 `virtual commands::DataStructure* activemq::wireformat::openwire::marshal::generated::ConsumerInfoMarshaller::createObject () const [virtual]`

Creates a new instance of the class that this class is a marshaling director for.

Returns

newly allocated Command

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1120).

6.166.3.2 `virtual unsigned char activemq::wireformat::openwire::marshal::generated::ConsumerInfoMarshaller::getDataStructureType () const [virtual]`

Gets the DataStructureType that this class marshals/unmarshals.

Returns

byte Id of this classes DataStructureType

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1122).

6.166.3.3 `virtual void activemq::wireformat::openwire::marshal::generated::ConsumerInfoMarshaller::looseMarshal (OpenWireFormat * format, commands::DataStructure * command, decaf::io::DataOutputStream * ds) [virtual]`

Tight Marshal to the given stream.

Parameters

<i>format</i>	- The OpenWireFormat properties
<i>command</i>	- the object to Marshal
<i>ds</i>	- DataOutputStream to marshal to

Exceptions

<i>IOException</i>	if an error occurs.
--------------------	---------------------

6.166

activemq::wireformat::openwire::marshal::generated::ConsumerInfoMarshaller
Class Reference 1021
Reimplemented from **activemq::wireformat::openwire::marshal::generated::BaseCommandMarshaller** (p. 500).

6.166.3.4 virtual void **activemq::wireformat::openwire::marshal::generated::ConsumerInfoMarshaller::looseUnmarshal** (**OpenWireFormat** * *format*,
commands::DataStructure * *command*, **decaf::io::DataInputStream** * *dis*)
[virtual]

Loose Un-marhsal to the given stream.

Parameters

<i>format</i>	- The OpenWireFormat properties
<i>command</i>	- the object to Un-Marshal
<i>dis</i>	- the DataInputStream to Un-Marshal from

Exceptions

<i>IOException</i>	if an error occurs.
--------------------	---------------------

Reimplemented from **activemq::wireformat::openwire::marshal::generated::BaseCommandMarshaller** (p. 501).

6.166.3.5 virtual int **activemq::wireformat::openwire::marshal::generated::ConsumerInfoMarshaller::tightMarshal1** (**OpenWireFormat** * *format*,
commands::DataStructure * *command*, **utils::BooleanStream** * *bs*)
[virtual]

Tight Marhsal to the given stream.

Parameters

<i>format</i>	The OpenWireFormat properties
<i>command</i>	The object to Marshal
<i>bs</i>	The boolean stream to marshal to.

Exceptions

<i>IOException</i>	if an error occurs.
--------------------	---------------------

Reimplemented from **activemq::wireformat::openwire::marshal::generated::BaseCommandMarshaller** (p. 503).

6.166.3.6 `virtual void activemq::wireformat::openwire::marshal::generated::ConsumerInfoMarshaller::tightMarshal2 (OpenWireFormat * format, commands::DataStructure * command, decaf::io::DataOutputStream * ds, utils::BooleanStream * bs) [virtual]`

Tight Marhsal to the given stream.

Parameters

<i>format</i>	- The OpenWireFormat properties
<i>command</i>	- the object to Marshal
<i>ds</i>	- the DataOutputStream to Marshal to
<i>bs</i>	- boolean stream to marshal to.

Exceptions

<i>IOException</i>	if an error occurs.
--------------------	---------------------

Reimplemented from **activemq::wireformat::openwire::marshal::generated::Base-CommandMarshaller** (p. 504).

6.166.3.7 `virtual void activemq::wireformat::openwire::marshal::generated::ConsumerInfoMarshaller::tightUnmarshal (OpenWireFormat * format, commands::DataStructure * command, decaf::io::DataInputStream * dis, utils::BooleanStream * bs) [virtual]`

Tight Un-marhsal to the given stream.

Parameters

<i>format</i>	- The OpenWireFormat properties
<i>command</i>	- the object to Un-Marshal
<i>dis</i>	- the DataInputStream to Un-Marshal from
<i>bs</i>	- boolean stream to unmarshal from.

Exceptions

<i>IOException</i>	if an error occurs.
--------------------	---------------------

Reimplemented from **activemq::wireformat::openwire::marshal::generated::Base-CommandMarshaller** (p. 505).

The documentation for this class was generated from the following file:

- `src/main/activemq/wireformat/openwire/marshal/generated/ConsumerInfo-Marshaller.h`

6.167 activemq::state::ConsumerState Class Reference

```
#include <src/main/activemq/state/ConsumerState.h>
```

Public Member Functions

- **ConsumerState** (const **Pointer**< **ConsumerInfo** > &info)
- virtual **~ConsumerState** ()
- std::string **toString** () const
- const **Pointer**< **ConsumerInfo** > & **getInfo** () const

6.167.1 Constructor & Destructor Documentation

6.167.1.1 **activemq::state::ConsumerState::ConsumerState** (const **Pointer**< **ConsumerInfo** > & *info*)

6.167.1.2 virtual **activemq::state::ConsumerState::~~ConsumerState** ()
[virtual]

6.167.2 Member Function Documentation

6.167.2.1 const **Pointer**<**ConsumerInfo**>& **activemq::state::ConsumerState::getInfo** () const [inline]

6.167.2.2 std::string **activemq::state::ConsumerState::toString** () const

The documentation for this class was generated from the following file:

- src/main/activemq/state/**ConsumerState.h**

6.168 activemq::commands::ControlCommand Class Reference

```
#include <src/main/activemq/commands/ControlCommand.h>
```

Inheritance diagram for activemq::commands::ControlCommand:

Public Member Functions

- **ControlCommand** ()
- virtual **~ControlCommand** ()
- virtual unsigned char **getDataStructureType** () const
Get the **DataStructure** (p. 1133) Type as defined in *CommandTypes.h*.

- virtual **ControlCommand** * **cloneDataStructure** () const
Clone this object and return a new instance that the caller now owns, this will be an exact copy of this one.
- virtual void **copyDataStructure** (const **DataStructure** *src)
Copy the contents of the passed object into this objects members, overwriting any existing data.
- virtual std::string **toString** () const
*Returns a string containing the information for this **DataStructure** (p. 1133) such as its type and value of its elements.*
- virtual bool **equals** (const **DataStructure** *value) const
*Compares the **DataStructure** (p. 1133) passed in to this one, and returns if they are equivalent.*
- virtual const std::string & **getCommand** () const
- virtual std::string & **getCommand** ()
- virtual void **setCommand** (const std::string &command)
- virtual **Pointer**< **Command** > **visit** (activemq::state::CommandVisitor *visitor)
Allows a Visitor to visit this command and return a response to the command based on the command type being visited.

Static Public Attributes

- static const unsigned char **ID_CONTROLCOMMAND** = 14

Protected Attributes

- std::string **command**

6.168.1 Constructor & Destructor Documentation

6.168.1.1 **activemq::commands::ControlCommand::ControlCommand** ()

6.168.1.2 **virtual activemq::commands::ControlCommand::~~ControlCommand** ()
 [virtual]

6.168.2 Member Function Documentation

6.168.2.1 **virtual ControlCommand*** **activemq::commands-
::ControlCommand::cloneDataStructure** () const
 [virtual]

Clone this object and return a new instance that the caller now owns, this will be an exact copy of this one.

Returns

new copy of this object.

Implements **activemq::commands::DataStructure** (p. 1133).

6.168.2.2 `virtual void activemq::commands::ControlCommand::copyDataStructure
(const DataStructure * src) [virtual]`

Copy the contents of the passed object into this objects members, overwriting any existing data.

Returns

src - Source Object

Reimplemented from **activemq::commands::BaseCommand** (p. 493).

6.168.2.3 `virtual bool activemq::commands::ControlCommand::equals (const
DataStructure * value) const [virtual]`

Compares the **DataStructure** (p. 1133) passed in to this one, and returns if they are equivalent.

Equivalent here means that they are of the same type, and that each element of the objects are the same.

Returns

true if **DataStructure** (p. 1133)'s are Equal.

Reimplemented from **activemq::commands::BaseCommand** (p. 494).

6.168.2.4 `virtual const std::string& activemq::commands::ControlCommand::get-
Command () const [virtual]`

6.168.2.5 `virtual std::string& activemq::commands::ControlCommand::get-
Command () [virtual]`

6.168.2.6 `virtual unsigned char activemq::commands::ControlCommand::getData-
StructureType () const [virtual]`

Get the **DataStructure** (p. 1133) Type as defined in CommandTypes.h.

Returns

The type of the data structure

Implements **activemq::commands::DataStructure** (p. 1137).

6.168.2.7 `virtual void activemq::commands::ControlCommand::setCommand (const std::string & command) [virtual]`

6.168.2.8 `virtual std::string activemq::commands::ControlCommand::toString () const [virtual]`

Returns a string containing the information for this **DataStructure** (p. 1133) such as its type and value of its elements.

Returns

formatted string useful for debugging.

Reimplemented from **activemq::commands::BaseCommand** (p. 498).

6.168.2.9 `virtual Pointer<Command> activemq::commands::ControlCommand::visit (activemq::state::CommandVisitor * visitor) [virtual]`

Allows a Visitor to visit this command and return a response to the command based on the command type being visited.

The command will call the proper processXXX method in the visitor.

Returns

a **Response** (p. 2298) to the visitor being called or NULL if no response.

Implements **activemq::commands::Command** (p. 871).

6.168.3 Field Documentation

6.168.3.1 `std::string activemq::commands::ControlCommand::command [protected]`

6.168.3.2 `const unsigned char activemq::commands::ControlCommand::ID_CONTROLCOMMAND = 14 [static]`

The documentation for this class was generated from the following file:

- `src/main/activemq/commands/ControlCommand.h`

6.169 activemq::wireformat::openwire::marshal::generated::ControlCommandMarshaller Class Reference

Marshaling code for Open Wire Format for **ControlCommandMarshaller** (p. 1025).

6.169 activemq::wireformat::openwire::marshal::generated::ControlCommandMarshaller Class

Reference

1027

```
#include <src/main/activemq/wireformat/openwire/marshal/generated/-  
ControlCommandMarshaller.h>
```

Inheritance diagram for activemq::wireformat::openwire::marshal::generated::ControlCommandMarshaller:

Public Member Functions

- **ControlCommandMarshaller** ()
- virtual **~ControlCommandMarshaller** ()
- virtual **commands::DataStructure * createObject** () const
Creates a new instance of the class that this class is a marshaling director for.
- virtual unsigned char **getDataStructureType** () const
Gets the DataStructureType that this class marshals/unmarshals.
- virtual void **tightUnmarshal** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataInputStream** *dataIn, **utils::BooleanStream** *bs)
Tight Un-marhsal to the given stream.
- virtual int **tightMarshal1** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **utils::BooleanStream** *bs)
Tight Marhsal to the given stream.
- virtual void **tightMarshal2** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataOutputStream** *dataOut, **utils::BooleanStream** *bs)
Tight Marhsal to the given stream.
- virtual void **looseUnmarshal** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataInputStream** *dataIn)
Loose Un-marhsal to the given stream.
- virtual void **looseMarshal** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataOutputStream** *dataOut)
Tight Marhsal to the given stream.

6.169.1 Detailed Description

Marshaling code for Open Wire Format for **ControlCommandMarshaller** (p. 1025).

NOTE!: This file is auto generated - do not modify! if you need to make a change, please see the Java Classes in the activemq-openwire-generator module

6.169.2 Constructor & Destructor Documentation

6.169.2.1 **activemq::wireformat::openwire::marshal::generated::ControlCommandMarshaller::ControlCommandMarshaller** ()
[inline]

6.169.2.2 `virtual activemq::wireformat::openwire::marshal::generated::ControlCommandMarshaller::~~ControlCommandMarshaller () [inline, virtual]`

6.169.3 Member Function Documentation

6.169.3.1 `virtual commands::DataStructure* activemq::wireformat::openwire::marshal::generated::ControlCommandMarshaller::createObject () const [virtual]`

Creates a new instance of the class that this class is a marshaling director for.

Returns

newly allocated Command

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1120).

6.169.3.2 `virtual unsigned char activemq::wireformat::openwire::marshal::generated::ControlCommandMarshaller::getDataStructureType () const [virtual]`

Gets the DataStructureType that this class marshals/unmarshals.

Returns

byte Id of this classes DataStructureType

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1122).

6.169.3.3 `virtual void activemq::wireformat::openwire::marshal::generated::ControlCommandMarshaller::looseMarshal (OpenWireFormat * format, commands::DataStructure * command, decaf::io::DataOutputStream * ds) [virtual]`

Tight Marshal to the given stream.

Parameters

<i>format</i>	- The OpenWireFormat properties
<i>command</i>	- the object to Marshal
<i>ds</i>	- DataOutputStream to marshal to

Exceptions

<i>IOException</i>	if an error occurs.
--------------------	---------------------

6.169 activemq::wireformat::openwire::marshal::generated::ControlCommand-Marshaller Class

Reference

1029

Reimplemented from **activemq::wireformat::openwire::marshal::generated::Base-CommandMarshaller** (p. 500).

6.169.3.4 virtual void **activemq::wireformat::openwire::marshal::generated::ControlCommandMarshaller::looseUnmarshal** (**OpenWireFormat** * *format*, **commands::DataStructure** * *command*, **decaf::io::DataInputStream** * *dis*) [virtual]

Loose Un-marhsal to the given stream.

Parameters

<i>format</i>	- The OpenWireFormat properties
<i>command</i>	- the object to Un-Marshal
<i>dis</i>	- the DataInputStream to Un-Marshal from

Exceptions

<i>IOException</i>	if an error occurs.
--------------------	---------------------

Reimplemented from **activemq::wireformat::openwire::marshal::generated::Base-CommandMarshaller** (p. 501).

6.169.3.5 virtual int **activemq::wireformat::openwire::marshal::generated::ControlCommandMarshaller::tightMarshal1** (**OpenWireFormat** * *format*, **commands::DataStructure** * *command*, **utils::BooleanStream** * *bs*) [virtual]

Tight Marhsal to the given stream.

Parameters

<i>format</i>	The OpenWireFormat properties
<i>command</i>	The object to Marshal
<i>bs</i>	The boolean stream to marshal to.

Exceptions

<i>IOException</i>	if an error occurs.
--------------------	---------------------

Reimplemented from **activemq::wireformat::openwire::marshal::generated::Base-CommandMarshaller** (p. 503).

6.169.3.6 `virtual void activemq::wireformat::openwire::marshal::generated::ControlCommandMarshaller::tightMarshal2 (OpenWireFormat * format, commands::DataStructure * command, decaf::io::DataOutputStream * ds, utils::BooleanStream * bs) [virtual]`

Tight Marshal to the given stream.

Parameters

<i>format</i>	- The OpenWireFormat properties
<i>command</i>	- the object to Marshal
<i>ds</i>	- the DataOutputStream to Marshal to
<i>bs</i>	- boolean stream to marshal to.

Exceptions

<i>IOException</i>	if an error occurs.
--------------------	---------------------

Reimplemented from **activemq::wireformat::openwire::marshal::generated::Base-CommandMarshaller** (p. 504).

6.169.3.7 `virtual void activemq::wireformat::openwire::marshal::generated::ControlCommandMarshaller::tightUnmarshal (OpenWireFormat * format, commands::DataStructure * command, decaf::io::DataInputStream * dis, utils::BooleanStream * bs) [virtual]`

Tight Un-marshal to the given stream.

Parameters

<i>format</i>	- The OpenWireFormat properties
<i>command</i>	- the object to Un-Marshal
<i>dis</i>	- the DataInputStream to Un-Marshal from
<i>bs</i>	- boolean stream to unmarshal from.

Exceptions

<i>IOException</i>	if an error occurs.
--------------------	---------------------

Reimplemented from **activemq::wireformat::openwire::marshal::generated::Base-CommandMarshaller** (p. 505).

The documentation for this class was generated from the following file:

- `src/main/activemq/wireformat/openwire/marshal/generated/ControlCommandMarshaller.h`

6.170 decaf::util::concurrent::CopyOnWriteArrayList< E > Class Template Reference

```
#include <src/main/decaf/util/concurrent/CopyOnWrite-  
ArrayList.h>
```

Inheritance diagram for decaf::util::concurrent::CopyOnWriteArrayList< E >:

Data Structures

- class **ArrayListIterator**

Public Member Functions

- **CopyOnWriteArrayList** ()
- **CopyOnWriteArrayList** (const **Collection**< E > &collection)
- **CopyOnWriteArrayList** (const **CopyOnWriteArrayList**< E > &collection)
- **CopyOnWriteArrayList** (const E *array, int **size**)
- virtual ~**CopyOnWriteArrayList** ()
- **CopyOnWriteArrayList**< E > & **operator=** (const **CopyOnWriteArrayList**< E > &list)
- **CopyOnWriteArrayList**< E > & **operator=** (const **Collection**< E > &list)
- virtual void **copy** (const **Collection**< E > &collection)
*Renders this **Collection** (p. 851) as a Copy of the given **Collection** (p. 851).*
- virtual bool **add** (const E &value)
Returns true if this collection changed as a result of the call.
- virtual bool **addAll** (const **Collection**< E > &collection)
Adds all of the elements in the specified collection to this collection.
- virtual void **clear** ()
Removes all of the elements from this collection (optional operation).
- virtual bool **contains** (const E &value) const
Returns true if this collection contains the specified element.
- virtual bool **containsAll** (const **Collection**< E > &collection) const
Returns true if this collection contains all of the elements in the specified collection.
- virtual bool **equals** (const **Collection**< E > &collection) const
Compares the passed collection to this one, if they contain the same elements, i.e.
- virtual bool **isEmpty** () const
- virtual bool **remove** (const E &value)
Removes a single instance of the specified element from the collection.
- virtual bool **removeAll** (const **Collection**< E > &collection)
Removes all this collection's elements that are also contained in the specified collection (optional operation).

- virtual bool **retainAll** (const **Collection**< E > &collection)

Retains only the elements in this collection that are contained in the specified collection (optional operation).
- virtual int **size** () const

Returns the number of elements in this collection.
- virtual std::vector< E > **toArray** () const

Returns an array containing all of the elements in this collection.
- virtual **decaf::util::Iterator** < E > * **iterator** ()
- virtual **decaf::util::Iterator** < E > * **iterator** () const
- virtual **ListIterator**< E > * **listIterator** ()
- virtual **ListIterator**< E > * **listIterator** () const
- virtual **ListIterator**< E > * **listIterator** (int index)
- virtual **ListIterator**< E > * **listIterator** (int index) const
- virtual int **indexOf** (const E &value) const

Returns the index of the first occurrence of the specified element in this list, or -1 if this list does not contain the element.
- virtual int **lastIndexOf** (const E &value) const

Returns the index of the last occurrence of the specified element in this list, or -1 if this list does not contain the element.
- virtual E **get** (int index) const

Gets the element contained at position passed.
- virtual E **set** (int index, const E &element)

Replaces the element at the specified position in this list with the specified element.
- virtual void **add** (int index, const E &element)

Inserts the specified element at the specified position in this list.
- virtual bool **addAll** (int index, const **Collection**< E > &collection)

Inserts all of the elements in the specified collection into this list at the specified position (optional operation).
- virtual E **removeAt** (int index)

Removes the element at the specified position in this list.
- virtual std::string **toString** () const
- bool **addIfAbsent** (const E &value)

*Adds the given value to the end of this **List** (p. 1658) if it is not already contained in this **List** (p. 1658).*
- int **addAllAbsent** (const **Collection**< E > &collection)

*Every element in the given collection that is not already contained in this **Collection** (p. 851) is added to the end of this collection.*
- int **lastIndexOf** (const E &value, int index)

*Searches backwards through the **List** (p. 1658) for the given element starting at the index specified.*
- int **indexOf** (const E &value, int index) const

*Searches the **List** (p. 1658) starting from the specified index and returns the index of the first item in the list that is equal to the given value.*
- virtual void **lock** ()

Locks the object.

- virtual bool **tryLock** ()
*Attempts to **Lock** (p. 1682) the object, if the lock is already held by another thread than this method returns false.*
- virtual void **unlock** ()
Unlocks the object.
- virtual void **wait** ()
Waits on a signal from this object, which is generated by a call to Notify.
- virtual void **wait** (long long millisecs)
Waits on a signal from this object, which is generated by a call to Notify.
- virtual void **wait** (long long millisecs, int nanos)
Waits on a signal from this object, which is generated by a call to Notify.
- virtual void **notify** ()
Signals a waiter on this object that it can now wake up and continue.
- virtual void **notifyAll** ()
Signals the waiters on this object that it can now wake up and continue.

Friends

- class **CopyOnWriteArraySet**

```
template<typename E> class decaf::util::concurrent::CopyOnWriteArrayList< E >
```

6.170.1 Constructor & Destructor Documentation

6.170.1.1 `template<typename E> decaf::util::concurrent::CopyOnWriteArrayList< E >::CopyOnWriteArrayList ()` [inline]

6.170.1.2 `template<typename E> decaf::util::concurrent::CopyOnWriteArrayList< E >::CopyOnWriteArrayList (const Collection< E > & collection)` [inline]

6.170.1.3 `template<typename E> decaf::util::concurrent::CopyOnWriteArrayList< E >::CopyOnWriteArrayList (const CopyOnWriteArrayList< E > & collection)` [inline]

6.170.1.4 `template<typename E> decaf::util::concurrent::CopyOnWriteArrayList< E >::CopyOnWriteArrayList (const E * array, int size)` [inline]

6.170.1.5 `template<typename E> virtual decaf::util::concurrent::CopyOnWriteArrayList< E >::~~CopyOnWriteArrayList ()` [inline, virtual]

6.170.2 Member Function Documentation

```
6.170.2.1  template<typename E> virtual bool decaf::util::concurrent::Copy-
           OnWriteArrayList< E >::add ( const E & value ) [inline,
           virtual]
```

Returns true if this collection changed as a result of the call.

(Returns false if this collection does not permit duplicates and already contains the specified element.)

Collections that support this operation may place limitations on what elements may be added to this collection. In particular, some collections will refuse to add null elements, and others will impose restrictions on the type of elements that may be added. **Collection** (p. 851) classes should clearly specify in their documentation any restrictions on what elements may be added.

If a collection refuses to add a particular element for any reason other than that it already contains the element, it must throw an exception (rather than returning false). This preserves the invariant that a collection always contains the specified element after this call returns.

For non-pointer values, i.e. class instances or string's the object will be copied into the collection, thus the object must support being copied, must not hide the copy constructor and assignment operator.

Parameters

<i>value</i>	The reference to the element to add to this Collection (p. 851).
--------------	---

Returns

true if the element was added to this **Collection** (p. 851).

Exceptions

<i>Unsupported-OperationException</i>	if this is an unmodifiable collection.
<i>NullPointerException</i>	if the Collection (p. 851) is a container of pointers and does not allow NULL values.
<i>IllegalArgumentException</i>	if some property of the element prevents it from being added to this collection
<i>IllegalStateException</i>	if the element cannot be added at this time due to insertion restrictions.

Implements **decaf::util::Collection< E >** (p. 853).

Referenced by decaf::util::concurrent::CopyOnWriteArrayList< ServiceListener * >::addAllAbsent().


```
6.170.2.2  template<typename E> virtual void decaf::util::concurrent::CopyOn-
WriteArrayList< E >::add ( int index, const E & element ) [inline,
virtual]
```

Inserts the specified element at the specified position in this list.

Shifts the element currently at that position (if any) and any subsequent elements to the right (adds one to their indices).

Parameters

<i>index</i>	The index at which the specified element is to be inserted.
<i>element</i>	The element to be inserted in this List (p. 1658).

Exceptions

<i>IndexOutOfBoundsException</i>	if the index is greater than size of the List (p. 1658).
<i>UnsupportedOperationException</i>	if this is an unmodifiable collection.
<i>NullPointerException</i>	if the Collection (p. 851) is a container of pointers and does not allow NULL values.
<i>IllegalArgumentException</i>	if some property of the element prevents it from being added to this collection
<i>IllegalStateException</i>	if the element cannot be added at this time due to insertion restrictions.

Implements **decaf::util::List< E >** (p. 1660).

```
6.170.2.3  template<typename E> virtual bool decaf::util::concurrent::CopyOnWrite-
ArrayList< E >::addAll ( const Collection< E > & collection ) [inline,
virtual]
```

Adds all of the elements in the specified collection to this collection.

The behavior of this operation is undefined if the specified collection is modified while the operation is in progress. (This implies that the behavior of this call is undefined if the specified collection is this collection, and this collection is nonempty.)

Parameters

<i>collection</i>	The Collection (p. 851) whose elements are added to this one.
-------------------	--

Returns

true if this collection changed as a result of the call

Exceptions

<i>UnsupportedOperationException</i>	if this is an unmodifiable collection.
<i>NullPointerException</i>	if the Collection (p. 851) is a container of pointers and does not allow NULL values.
<i>IllegalArgumentException</i>	if some property of an element prevents it from being added to this collection
<i>IllegalStateException</i>	if an element cannot be added at this time due to insertion restrictions.

Implements **decaf::util::Collection**< **E** > (p. 855).

6.170.2.4 `template<typename E> virtual bool decaf::util::concurrent::CopyOnWrite-
ArrayList< E >::addAll (int index, const Collection< E > & source)
[inline, virtual]`

Inserts all of the elements in the specified collection into this list at the specified position (optional operation).

Shifts the element currently at that position (if any) and any subsequent elements to the right (increases their indices). The new elements will appear in this list in the order that they are returned by the specified collection's iterator. The behavior of this operation is undefined if the specified collection is modified while the operation is in progress. (Note that this will occur if the specified collection is this list, and it's nonempty.)

Parameters

<i>index</i>	The index at which to insert the first element from the specified collection
<i>source</i>	The Collection (p. 851) containing elements to be added to this list

Returns

true if this list changed as a result of the call

Exceptions

<i>IndexOutOfBoundsException</i>	if the index given is less than zero or greater than the List (p. 1658) size.
<i>UnsupportedOperationException</i>	if this is an unmodifiable collection.
<i>NullPointerException</i>	if the Collection (p. 851) is a container of pointers and does not allow NULL values.
<i>IllegalArgumentException</i>	if some property of the element prevents it from being added to this collection
<i>IllegalStateException</i>	if the element cannot be added at this time due to insertion restrictions.

Implements **decaf::util::List< E >** (p. 1661).

6.170.2.5 `template<typename E> int decaf::util::concurrent::CopyOnWriteArrayList< E >::addAllAbsent (const Collection< E > & collection)`
`[inline]`

Every element in the given collection that is not already contained in this **Collection** (p. 851) is added to the end of this collection.

The order that the elements are added is dictated by the order that the collection's iterator returns them.

Parameters

<i>collection</i>	The collection whose elements are to be added if not already in this List (p. 1658).
-------------------	---

Returns

the number of elements that are added to this **List** (p. 1658).

6.170.2.6 `template<typename E> bool decaf::util::concurrent::CopyOnWriteArrayList< E >::addIfAbsent (const E & value)`
`[inline]`

Adds the given value to the end of this **List** (p. 1658) if it is not already contained in this **List** (p. 1658).

Parameters

<i>value</i>	The element to be added if not already contained in this List (p. 1658).
--------------	---

Returns

true if the element is added to this **List** (p. 1658).

6.170.2.7 `template<typename E> virtual void decaf::util::concurrent::CopyOnWriteArrayList< E >::clear ()` `[inline, virtual]`

Removes all of the elements from this collection (optional operation).

This collection will be empty after this method returns unless it throws an exception.

Exceptions

<i>UnsupportedOperationException</i>	if this is an unmodifiable collection.
--------------------------------------	--

Implements **decaf::util::Collection**< **E** > (p. 856).

Referenced by `decaf::util::concurrent::CopyOnWriteArrayList< ServiceListener * >::operator=()`.

6.170.2.8 `template<typename E> virtual bool decaf::util::concurrent::CopyOnWriteArrayList< E >::contains (const E & value) const [inline, virtual]`

Returns true if this collection contains the specified element.

More formally, returns true if and only if this collection contains at least one element *e* such that (*value* == NULL ? *e* == NULL : *value* == *e*).

Parameters

<i>value</i>	The value to check for presence in the collection.
--------------	--

Returns

true if there is at least one of the elements in the collection

Exceptions

<i>NullPointerException</i>	if the Collection (p.851) contains pointers and the Collection (p.851) does not allow for NULL elements (optional check).
-----------------------------	---

Implements **decaf::util::Collection**< **E** > (p. 857).

Referenced by `decaf::util::concurrent::CopyOnWriteArrayList< ServiceListener * >::containsAll()`.

6.170.2.9 `template<typename E> virtual bool decaf::util::concurrent::CopyOnWriteArrayList< E >::containsAll (const Collection< E > & collection) const [inline, virtual]`

Returns true if this collection contains all of the elements in the specified collection.

Parameters

<i>collection</i>	The Collection (p.851) to compare to this one.
-------------------	---

Exceptions

<i>NullPointerException</i>	if the Collection (p.851) contains pointers and the Collection (p.851) does not allow for NULL elements (optional check).
-----------------------------	---

Implements **decaf::util::Collection**< **E** > (p. 858).

6.170 decaf::util::concurrent::CopyOnWriteArrayList< E > Class Template Reference

1039

6.170.2.10 `template<typename E> virtual void decaf::util::concurrent::CopyOnWriteArrayList< E >::copy (const Collection< E > & collection) [inline, virtual]`

Renders this **Collection** (p. 851) as a Copy of the given **Collection** (p. 851).

Parameters

<i>collection</i>	The collection to mirror.
-------------------	---------------------------

Exceptions

<i>UnsupportedOperationException</i>	if this is an unmodifiable collection.
<i>IllegalStateException</i>	if the elements cannot be added at this time due to insertion restrictions.

Implements **decaf::util::Collection< E >** (p. 859).

6.170.2.11 `template<typename E> virtual bool decaf::util::concurrent::CopyOnWriteArrayList< E >::equals (const Collection< E > & value) const [inline, virtual]`

Compares the passed collection to this one, if they contain the same elements, i.e. all their elements are equivalent, then it returns true.

Returns

true if the Collections contain the same elements.

Implements **decaf::util::Collection< E >** (p. 860).

6.170.2.12 `template<typename E> virtual E decaf::util::concurrent::CopyOnWriteArrayList< E >::get (int index) const [inline, virtual]`

Gets the element contained at position passed.

Parameters

<i>index</i>	The position to get.
--------------	----------------------

Returns

value at index specified.

Exceptions

<i>IndexOutOfBoundsException</i>	if the index given is less than zero or greater than the List (p. 1658) size.
----------------------------------	--

Implements **decaf::util::List< E >** (p. 1662).

6.170.2.13 **template<typename E> virtual int decaf::util::concurrent::CopyOnWriteArrayList< E >::indexOf (const E & value) const** [inline, virtual]

Returns the index of the first occurrence of the specified element in this list, or -1 if this list does not contain the element.

More formally, returns the lowest index *i* such that `get(i) == value`, or -1 if there is no such index.

Parameters

<i>value</i>	The element to search for in this List (p. 1658).
--------------	--

Returns

the index of the first occurrence of the specified element in this list,

Exceptions

<i>NullPointerException</i>	if the Collection (p. 851) is a container of pointers and does not allow NULL values.
-----------------------------	--

Implements **decaf::util::List< E >** (p. 1663).

Referenced by `decaf::util::concurrent::CopyOnWriteArrayList< ServiceListener * >::addAllAbsent()`, `decaf::util::concurrent::CopyOnWriteArrayList< ServiceListener * >::addIfAbsent()`, and `decaf::util::concurrent::CopyOnWriteArrayList< ServiceListener * >::remove()`.

6.170.2.14 **template<typename E> int decaf::util::concurrent::CopyOnWriteArrayList< E >::indexOf (const E & value, int index) const** [inline]

Searches the **List** (p. 1658) starting from the specified index and returns the index of the first item in the list that is equal to the given value.

Parameters

<i>value</i>	The value to search for in the List (p. 1658).
<i>index</i>	The index in the List (p. 1658) to begin the search from.

Returns

the index in the **List** (p. 1658) that matches the given element or -1 if not found.

Exceptions

<i>IndexOutOfBoundsException</i>	if the given index is negative.
----------------------------------	---------------------------------

6.170.2.15 `template<typename E> virtual bool decaf::util::concurrent::CopyOnWriteArrayList< E >::isEmpty () const [inline, virtual]`

Returns

true if this collection contains no elements.

Implements **decaf::util::Collection< E >** (p. 860).

6.170.2.16 `template<typename E> virtual decaf::util::Iterator<E>* decaf::util::concurrent::CopyOnWriteArrayList< E >::iterator () [inline, virtual]`

Returns

an iterator over a set of elements of type T.

Implements **decaf::lang::Iterable< E >** (p. 1557).

Referenced by `decaf::util::concurrent::CopyOnWriteArrayList< ServiceListener * >::equals()`.

6.170.2.17 `template<typename E> virtual decaf::util::Iterator<E>* decaf::util::concurrent::CopyOnWriteArrayList< E >::iterator () const [inline, virtual]`

Implements **decaf::lang::Iterable< E >** (p. 1558).

6.170.2.18 `template<typename E> virtual int decaf::util::concurrent::CopyOnWriteArrayList< E >::lastIndexOf (const E & value) const [inline, virtual]`

Returns the index of the last occurrence of the specified element in this list, or -1 if this list does not contain the element.

More formally, returns the highest index i such that `get(i) == value` or -1 if there is no such index.

Parameters

<i>value</i>	The element to search for in this List (p. 1658).
--------------	--

Returns

the index of the last occurrence of the specified element in this list.

Exceptions

<i>NullPointerException</i>	if the Collection (p. 851) is a container of pointers and does not allow NULL values.
-----------------------------	--

Implements **decaf::util::List< E >** (p. 1664).

```
6.170.2.19  template<typename E> int decaf::util::concurrent::CopyOn-
WriteArrayList< E >::lastIndexOf ( const E & value, int index )
[inline]
```

Searches backwards through the **List** (p. 1658) for the given element starting at the index specified.

Parameters

<i>value</i>	The value to search for in the List (p. 1658).
<i>index</i>	The index in the list to begin the search from.

Returns

the index in the list that matches the value given, or -1 if not found.

Exceptions

<i>IndexOutOfBoundsException</i>	if the given index is greater than or equal to the List (p. 1658) size.
----------------------------------	--

```
6.170.2.20  template<typename E> virtual ListIterator<E>* decaf::util::concurrent-
::CopyOnWriteArrayList< E >::listIterator ( ) [inline,
virtual]
```

Returns

a list iterator over the elements in this list (in proper sequence).

Implements **decaf::util::List< E >** (p. 1665).

6.170.2.21 `template<typename E> virtual ListIterator<E>* decaf::util::concurrent::CopyOnWriteArrayList< E >::listIterator () const [inline, virtual]`

Implements **decaf::util::List< E >** (p. 1666).

6.170.2.22 `template<typename E> virtual ListIterator<E>* decaf::util::concurrent::CopyOnWriteArrayList< E >::listIterator (int index) [inline, virtual]`

Parameters

<i>index</i>	index of first element to be returned from the list iterator (by a call to the next method).
--------------	--

Returns

a list iterator of the elements in this list (in proper sequence), starting at the specified position in this list. The specified index indicates the first element that would be returned by an initial call to next. An initial call to previous would return the element with the specified index minus one.

Exceptions

<i>IndexOutOfBoundsException</i>	if the index is out of range (<code>index < 0 index > size()</code>) (p. 1046))
----------------------------------	--

Implements **decaf::util::List< E >** (p. 1666).

6.170.2.23 `template<typename E> virtual ListIterator<E>* decaf::util::concurrent::CopyOnWriteArrayList< E >::listIterator (int index) const [inline, virtual]`

Implements **decaf::util::List< E >** (p. 1668).

6.170.2.24 `template<typename E> virtual void decaf::util::concurrent::CopyOnWriteArrayList< E >::lock () [inline, virtual]`

Locks the object.

Exceptions

<i>RuntimeException</i>	if an error occurs while locking the object.
-------------------------	--

Implements **decaf::util::concurrent::Synchronizable** (p. 2640).

6.170.2.25 `template<typename E> virtual void decaf::util::concurrent-
::CopyOnWriteArrayList< E >::notify () [inline,
virtual]`

Signals a waiter on this object that it can now wake up and continue.

Must have this object locked before calling.

Exceptions

<i>IllegalMonitorState- Exception</i>	- if the current thread is not the owner of the the Synchronizable (p. 2639) Object.
<i>RuntimeException</i>	if an error occurs while notifying one of the waiting threads.

Implements **decaf::util::concurrent::Synchronizable** (p. 2641).

6.170.2.26 `template<typename E> virtual void decaf::util::concurrent::-
CopyOnWriteArrayList< E >::notifyAll () [inline,
virtual]`

Signals the waiters on this object that it can now wake up and continue.

Must have this object locked before calling.

Exceptions

<i>IllegalMonitorState- Exception</i>	- if the current thread is not the owner of the the Synchronizable (p. 2639) Object.
<i>RuntimeException</i>	if an error occurs while notifying the waiting threads.

Implements **decaf::util::concurrent::Synchronizable** (p. 2642).

6.170.2.27 `template<typename E> CopyOnWriteArrayList<E>&
decaf::util::concurrent::CopyOnWriteArrayList< E >::operator= (const
CopyOnWriteArrayList< E > & list) [inline]`

6.170.2.28 `template<typename E> CopyOnWriteArrayList<E>&
decaf::util::concurrent::CopyOnWriteArrayList< E >::operator= (const
Collection< E > & list) [inline]`

6.170.2.29 `template<typename E> virtual bool decaf::util::concurrent::Copy-
OnWriteArrayList< E >::remove (const E & value) [inline,
virtual]`

Removes a single instance of the specified element from the collection.

More formally, removes an element *e* such that (*value* == NULL ? *e* == NULL : *value* == *e*), if this collection contains one or more such elements. Returns true if this collection contained the specified element (or equivalently, if this collection changed as a result of the call).

6.170 decaf::util::concurrent::CopyOnWriteArrayList< E > Class Template Reference

1045

Parameters

<i>value</i>	The reference to the element to remove from this Collection (p. 851).
--------------	--

Returns

true if the collection was changed, false otherwise.

Exceptions

<i>UnsupportedOperationException</i>	if this is an unmodifiable collection.
<i>NullPointerException</i>	if the Collection (p. 851) is a container of pointers and does not allow NULL values.

Implements **decaf::util::Collection< E >** (p. 861).

```
6.170.2.30  template<typename E> virtual bool decaf::util::concurrent::CopyOn-  
            WriteArrayList< E >::removeAll ( const Collection< E > & collection )  
            [inline, virtual]
```

Removes all this collection's elements that are also contained in the specified collection (optional operation).

After this call returns, this collection will contain no elements in common with the specified collection.

Parameters

<i>collection</i>	The Collection (p. 851) whose elements are to be removed from this one.
-------------------	--

Returns

true if the collection changed as a result of this call.

Exceptions

<i>UnsupportedOperationException</i>	if this is an unmodifiable collection.
<i>NullPointerException</i>	if the Collection (p. 851) is a container of pointers and does not allow NULL values.

Implements **decaf::util::Collection< E >** (p. 862).

6.170.2.31 `template<typename E> virtual E decaf::util::concurrent::CopyOnWriteArrayList< E >::removeAt (int index) [inline, virtual]`

Removes the element at the specified position in this list.

Shifts any subsequent elements to the left (subtracts one from their indices). Returns the element that was removed from the list.

Parameters

<i>index</i>	- the index of the element to be removed.
--------------	---

Returns

the element previously at the specified position.

Exceptions

<i>IndexOutOfBoundsException</i>	if the index given is less than zero or greater than the List (p. 1658) size.
<i>UnsupportedOperationException</i>	if this is an unmodifiable collection.

Implements **decaf::util::List< E >** (p. 1668).

Referenced by `decaf::util::concurrent::CopyOnWriteArrayList< ServiceListener * >::remove()`.

6.170.2.32 `template<typename E> virtual bool decaf::util::concurrent::CopyOnWriteArrayList< E >::retainAll (const Collection< E > & collection) [inline, virtual]`

Retains only the elements in this collection that are contained in the specified collection (optional operation).

In other words, removes from this collection all of its elements that are not contained in the specified collection.

Parameters

<i>collection</i>	The Collection (p. 851) whose elements are to be retained.
-------------------	---

Returns

true if the collection changed as a result of this call.

6.170 decaf::util::concurrent::CopyOnWriteArrayList< E > Class Template Reference

1047

Exceptions

<i>UnsupportedOperationException</i>	if this is an unmodifiable collection.
<i>NullPointerException</i>	if the Collection (p. 851) is a container of pointers and does not allow NULL values.

Implements **decaf::util::Collection< E >** (p. 863).

6.170.2.33 `template<typename E> virtual E decaf::util::concurrent::CopyOnWriteArrayList< E >::set (int index, const E & element) [inline, virtual]`

Replaces the element at the specified position in this list with the specified element.

Parameters

<i>index</i>	The index of the element to replace.
<i>element</i>	The element to be stored at the specified position.

Returns

the element previously at the specified position.

Exceptions

<i>IndexOutOfBoundsException</i>	if the index given is less than zero or greater than the List (p. 1658) size.
<i>UnsupportedOperationException</i>	if this is an unmodifiable collection.
<i>NullPointerException</i>	if the Collection (p. 851) is a container of pointers and does not allow NULL values.
<i>IllegalArgumentException</i>	if some property of the element prevents it from being added to this collection
<i>IllegalStateException</i>	if the element cannot be added at this time due to insertion restrictions.

Implements **decaf::util::List< E >** (p. 1669).

6.170.2.34 `template<typename E> virtual int decaf::util::concurrent::CopyOnWriteArrayList< E >::size () const [inline, virtual]`

Returns the number of elements in this collection.

If this collection contains more than Integer::MAX_VALUE elements, returns Integer::MAX_VALUE.

Returns

the number of elements in this collection

Implements **decaf::util::Collection< E >** (p. 864).

Referenced by `decaf::util::concurrent::CopyOnWriteArrayList< ServiceListener * >::add()`, `decaf::util::concurrent::CopyOnWriteArrayList< ServiceListener * >::addAll()`, `decaf::util::concurrent::CopyOnWriteArrayList< ServiceListener * >::addAllAbsent()`, `decaf::util::concurrent::CopyOnWriteArrayList< ServiceListener * >::addIfAbsent()`, `decaf::util::concurrent::CopyOnWriteArrayList< ServiceListener * >::CopyOnWriteArrayList()`, `decaf::util::concurrent::CopyOnWriteArrayList< ServiceListener * >::equals()`, `decaf::util::concurrent::CopyOnWriteArrayList< ServiceListener * >::removeAll()`, `decaf::util::concurrent::CopyOnWriteArrayList< ServiceListener * >::removeAt()`, `decaf::util::concurrent::CopyOnWriteArrayList< ServiceListener * >::retainAll()`, and `decaf::util::concurrent::CopyOnWriteArrayList< ServiceListener * >::set()`.

6.170.2.35 `template<typename E> virtual std::vector<E> decaf::util::concurrent::CopyOnWriteArrayList< E >::toArray () const [inline, virtual]`

Returns an array containing all of the elements in this collection.

If the collection makes any guarantees as to what order its elements are returned by its iterator, this method must return the elements in the same order.

This method acts as bridge between array-based and collection-based APIs.

Returns

an array of the elements in this collection in the form of an STL vector.

Implements **decaf::util::Collection< E >** (p. 865).

6.170.2.36 `template<typename E> virtual std::string decaf::util::concurrent::CopyOnWriteArrayList< E >::toString () const [inline, virtual]`

6.170.2.37 `template<typename E> virtual bool decaf::util::concurrent::CopyOnWriteArrayList< E >::tryLock () [inline, virtual]`

Attempts to **Lock** (p. 1682) the object, if the lock is already held by another thread than this method returns false.

Returns

true if the lock was acquired, false if it is already held by another thread.

6.170 decaf::util::concurrent::CopyOnWriteArrayList< E > Class Template Reference

1049

Exceptions

<i>RuntimeException</i>	if an error occurs while locking the object.
-------------------------	--

Implements **decaf::util::concurrent::Synchronizable** (p. 2643).

6.170.2.38 `template<typename E> virtual void decaf::util::concurrent-
::CopyOnWriteArrayList< E >::unlock () [inline,
virtual]`

Unlocks the object.

Exceptions

<i>RuntimeException</i>	if an error occurs while unlocking the object.
-------------------------	--

Implements **decaf::util::concurrent::Synchronizable** (p. 2645).

6.170.2.39 `template<typename E> virtual void decaf::util::concurrent-
::CopyOnWriteArrayList< E >::wait () [inline,
virtual]`

Waits on a signal from this object, which is generated by a call to Notify.

Must have this object locked before calling.

Exceptions

<i>RuntimeException</i>	if an error occurs while waiting on the object.
<i>InterruptedException</i>	if the wait is interrupted before it completes.
<i>IllegalMonitorState- Exception</i>	- if the current thread is not the owner of the the Synchronizable (p. 2639) Object.

Implements **decaf::util::concurrent::Synchronizable** (p. 2646).

6.170.2.40 `template<typename E> virtual void decaf::util::concurrent::Copy-
OnWriteArrayList< E >::wait (long long millisecs) [inline,
virtual]`

Waits on a signal from this object, which is generated by a call to Notify.

Must have this object locked before calling. This wait will timeout after the specified time interval.

Parameters

<i>millisecs</i>	the time in milliseconds to wait, or WAIT_INFINITE
------------------	--

Exceptions

<i>RuntimeException</i>	if an error occurs while waiting on the object.
<i>InterruptedException</i>	if the wait is interrupted before it completes.
<i>IllegalMonitorStateException</i>	- if the current thread is not the owner of the the Synchronizable (p. 2639) Object.

Implements **decaf::util::concurrent::Synchronizable** (p. 2647).

6.170.2.41 `template<typename E> virtual void decaf::util::concurrent::CopyOnWriteArrayList< E >::wait (long long millisecs, int nanos) [inline, virtual]`

Waits on a signal from this object, which is generated by a call to Notify.

Must have this object locked before calling. This wait will timeout after the specified time interval. This method is similar to the one argument wait function except that it add a finer grained control over the amount of time that it waits by adding in the additional nanosecond argument.

NOTE: The ability to wait accurately at a nanosecond scale depends on the platform and OS that the Decaf API is running on, some systems do not provide an accurate enough clock to provide this level of granularity.

Parameters

<i>millisecs</i>	the time in milliseconds to wait, or WAIT_INFINITE
<i>nanos</i>	additional time in nanoseconds with a range of 0-999999

Exceptions

<i>IllegalArgument-Exception</i>	if an error occurs or the nanos argument is not in the range of [0-999999]
<i>RuntimeException</i>	if an error occurs while waiting on the object.
<i>InterruptedException</i>	if the wait is interrupted before it completes.
<i>IllegalMonitorStateException</i>	- if the current thread is not the owner of the the Synchronizable (p. 2639) Object.

Implements **decaf::util::concurrent::Synchronizable** (p. 2648).

6.170.3 Friends And Related Function Documentation

6.170.3.1 `template<typename E> friend class CopyOnWriteArraySet [friend]`

The documentation for this class was generated from the following file:

- src/main/decaf/util/concurrent/**CopyOnWriteArrayList.h**

6.171 decaf::util::concurrent::CopyOnWriteArraySet< E > Class - Template Reference

Since the **CopyOnWriteArraySet** (p. 1050) and the **CopyOnWriteArrayList** (p. 1030) share much of the same operational semantics this class uses the **CopyOnWriteArrayList** (p. 1030) for all its underlying operations.

```
#include <src/main/decaf/util/concurrent/CopyOnWrite-
ArraySet.h>
```

Inheritance diagram for decaf::util::concurrent::CopyOnWriteArraySet< E >:

Public Member Functions

- **CopyOnWriteArraySet** ()
- **CopyOnWriteArraySet** (const **Collection**< E > &collection)
- **CopyOnWriteArraySet** (const E *array, int **size**)
- virtual ~**CopyOnWriteArraySet** ()
- virtual void **copy** (const **Collection**< E > &collection)
*Renders this **Collection** (p. 851) as a Copy of the given **Collection** (p. 851).*
- virtual **decaf::util::Iterator** < E > * **iterator** ()
- virtual **decaf::util::Iterator** < E > * **iterator** () const
- virtual int **size** () const
Returns the number of elements in this collection.
- virtual bool **isEmpty** () const
Returns true if this collection contains no elements.
- virtual bool **add** (const E &value)
Returns true if this collection changed as a result of the call.
- virtual bool **addAll** (const **Collection**< E > &collection)
Adds all of the elements in the specified collection to this collection. The behavior of this operation is undefined if the specified collection is modified while the operation is in progress. (This implies that the behavior of this call is undefined if the specified collection is this collection, and this collection is nonempty.)

Parameters

collection	<i>The Collection (p. 851) whose elements are added to this one.</i>
------------	---

Returns

true if this collection changed as a result of the call

Exceptions

UnsupportedOperation-Exception	<i>if this is an unmodifiable collection.</i>
NullPointerException	<i>if the Collection (p. 851) is a container of pointers and does not allow NULL values.</i>
IllegalArgumentException	<i>if some property of an element prevents it from being added to this collection</i>
IllegalStateException	<i>if an element cannot be added at this time due to insertion restrictions.</i>

This implementation iterates over the specified collection, and adds each object returned by the iterator to this collection, in turn.

Note that this implementation will throw an `UnsupportedOperationException` unless `add` is overridden (assuming the specified collection is non-empty).

- virtual void **clear** ()

Removes all of the elements from this collection (optional operation).

- virtual bool **contains** (const E &value) const

Returns true if this collection contains the specified element.

More formally, returns true if and only if this collection contains at least one element `e` such that `(value == NULL ? e == NULL : value == e)`.

Parameters

value	<i>The value to check for presence in the collection.</i>
-------	---

Returns

true if there is at least one of the elements in the collection

Exceptions

NullPointerException	<i>if the Collection (p. 851) contains pointers and the Collection (p. 851) does not allow for NULL elements (optional check).</i>
----------------------	--

This implementation iterates over the elements in the collection, checking each element in turn for equality with the specified element.

- virtual bool **containsAll** (const **Collection**< E > &collection) const

Returns true if this collection contains all of the elements in the specified collection.

Parameters

collection	<i>The Collection (p. 851) to compare to this one.</i>
------------	---

Exceptions

NullPointerException	<i>if the Collection (p. 851) contains pointers and the Collection (p. 851) does not allow for NULL elements (optional check).</i>
----------------------	--

This implementation iterates over the specified collection, checking each element returned by the iterator in turn to see if it's contained in this collection. If all elements are so contained true is returned, otherwise false.

- virtual bool **remove** (const E &value)

Removes a single instance of the specified element from the collection.

More formally, removes an element `e` such that `(value == NULL ? e == NULL : value == e)`, if this collection contains one or more such elements. Returns true if this collection contained the specified element (or equivalently, if this collection changed as a result of the call).

Parameters

value	<i>The reference to the element to remove from this Collection (p. 851).</i>
-------	---

Returns

true if the collection was changed, false otherwise.

Exceptions

UnsupportedOperationException	<i>if this is an unmodifiable collection.</i>
NullPointerException	<i>if the Collection (p. 851) is a container of pointers and does not allow NULL values.</i>

This implementation iterates over the collection looking for the specified element. - If it finds the element, it removes the element from the collection using the iterator's remove method.

Note that this implementation throws an `UnsupportedOperationException` if the iterator returned by this collection's iterator method does not implement the remove method and this collection contains the specified object.

- virtual bool **removeAll** (const **Collection**< E > &collection)

Removes all this collection's elements that are also contained in the specified collection (optional operation).

After this call returns, this collection will contain no elements in common with the specified collection.

Parameters

collection	<i>The Collection (p. 851) whose elements are to be removed from this one.</i>
------------	---

Returns

true if the collection changed as a result of this call.

Exceptions

UnsupportedOperationException	<i>if this is an unmodifiable collection.</i>
NullPointerException	<i>if the Collection (p. 851) is a container of pointers and does not allow NULL values.</i>

This implementation iterates over this collection, checking each element returned by the iterator in turn to see if it's contained in the specified collection. If it's so contained, it's removed from this collection with the iterator's remove method.

Note that this implementation will throw an `UnsupportedOperationException` if the iterator returned by the iterator method does not implement the remove method and this collection contains one or more elements in common with the specified collection.

This implementation determines which is the smaller of this set and the specified collection, by invoking the size method on each. If this set has fewer elements, then the implementation iterates over this set, checking each element returned by the iterator in turn to see if it is contained in the specified collection. If it is so contained, it is removed from this set with the iterator's remove method. If the specified collection has fewer elements, then the implementation iterates over the specified collection, removing from this set each element returned by the iterator, using this set's remove method.

Note that this implementation will throw an `UnsupportedOperationException` if the iterator returned by the iterator method does not implement the remove method.

- virtual bool **retainAll** (const **Collection**< E > &collection)

Retains only the elements in this collection that are contained in the specified collection (optional operation).

In other words, removes from this collection all of its elements that are not contained in the specified collection.

Parameters

collection	<i>The Collection (p. 851) whose elements are to be retained.</i>
------------	--

Returns

true if the collection changed as a result of this call.

Exceptions

UnsupportedOperation- Exceptio	<i>if this is an unmodifiable collection.</i>
NullPointerException	<i>if the Collection (p. 851) is a container of pointers and does not allow NULL values.</i>

This implementation iterates over this collection, checking each element returned by the iterator in turn to see if it's contained in the specified collection. If it's not so contained, it's removed from this collection with the iterator's remove method.

Note that this implementation will throw an `UnsupportedOperationException` if the iterator returned by the iterator method does not implement the remove method and this collection contains one or more elements not present in the specified collection.

- virtual `std::vector< E > toArray () const`

*Answers an STL vector containing copies of all elements contained in this **Collection** (p. 851).*

- virtual `bool equals (const Collection< E > &collection) const`

*Answers true if this **Collection** (p. 851) and the one given are the same size and if each element contained in the **Collection** (p. 851) given is equal to an element contained in this collection.*

6.171.1 Detailed Description

```
template<typename E>class decaf::util::concurrent::CopyOnWriteArraySet< E >
```

Since the **CopyOnWriteArraySet** (p. 1050) and the **CopyOnWriteArrayList** (p. 1030) share much of the same operational semantics this class uses the **CopyOnWriteArrayList** (p. 1030) for all its underlying operations.

This collection is best used in applications where the **Set** (p. 2397) size is usually small and write operations are minimal as they result in a copy of the underlying array being created. Reads are generally fast and the iterators provided by this collection do not block as they operate on a snapshot of the data taken at the time of their creation.

Since

1.0

6.171.2 Constructor & Destructor Documentation

6.171.2.1 `template<typename E > decaf::util::concurrent::CopyOnWriteArraySet< E >::CopyOnWriteArraySet () [inline]`

6.171.2.2 `template<typename E > decaf::util::concurrent::CopyOnWriteArraySet< E >::CopyOnWriteArraySet (const Collection< E > & collection) [inline]`

References `decaf::util::concurrent::CopyOnWriteArraySet< E >::copy()`.

6.171.2.3 `template<typename E > decaf::util::concurrent::CopyOnWriteArraySet< E >::CopyOnWriteArraySet (const E * array, int size) [inline]`

References decaf::util::concurrent::CopyOnWriteArraySet< E >::size().

6.171.2.4 `template<typename E > virtual decaf::util::concurrent::CopyOnWriteArraySet< E >::~~CopyOnWriteArraySet () [inline, virtual]`

6.171.3 Member Function Documentation

6.171.3.1 `template<typename E > virtual bool decaf::util::concurrent::CopyOnWriteArraySet< E >::add (const E & value) [inline, virtual]`

Returns true if this collection changed as a result of the call.

(Returns false if this collection does not permit duplicates and already contains the specified element.)

Collections that support this operation may place limitations on what elements may be added to this collection. In particular, some collections will refuse to add null elements, and others will impose restrictions on the type of elements that may be added. **Collection** (p. 851) classes should clearly specify in their documentation any restrictions on what elements may be added.

If a collection refuses to add a particular element for any reason other than that it already contains the element, it must throw an exception (rather than returning false). This preserves the invariant that a collection always contains the specified element after this call returns.

For non-pointer values, i.e. class instances or string's the object will be copied into the collection, thus the object must support being copied, must not hide the copy constructor and assignment operator.

Parameters

<i>value</i>	The reference to the element to add to this Collection (p. 851).
--------------	---

Returns

true if the element was added to this **Collection** (p. 851).

Exceptions

<i>UnsupportedOperationException</i>	if this is an unmodifiable collection.
<i>NullPointerException</i>	if the Collection (p. 851) is a container of pointers and does not allow NULL values.
<i>IllegalArgumentException</i>	if some property of the element prevents it from being added to this collection

<i>IllegalStateException</i>	if the element cannot be added at this time due to insertion restrictions.
------------------------------	--

Implements **decaf::util::Collection< E >** (p. 853).

6.171.3.2 `template<typename E> virtual bool decaf::util::concurrent::CopyOnWriteArraySet< E >::addAll (const Collection< E > & collection) [inline, virtual]`

Adds all of the elements in the specified collection to this collection.

The behavior of this operation is undefined if the specified collection is modified while the operation is in progress. (This implies that the behavior of this call is undefined if the specified collection is this collection, and this collection is nonempty.)

Parameters

<i>collection</i>	The Collection (p. 851) whose elements are added to this one.
-------------------	--

Returns

true if this collection changed as a result of the call

Exceptions

<i>UnsupportedOperationException</i>	if this is an unmodifiable collection.
<i>NullPointerException</i>	if the Collection (p. 851) is a container of pointers and does not allow NULL values.
<i>IllegalArgumentException</i>	if some property of an element prevents it from being added to this collection
<i>IllegalStateException</i>	if an element cannot be added at this time due to insertion restrictions.

This implementation iterates over the specified collection, and adds each object returned by the iterator to this collection, in turn.

Note that this implementation will throw an `UnsupportedOperationException` unless `add` is overridden (assuming the specified collection is non-empty).

This implementation iterates over the specified collection, and adds each object returned by the iterator to this collection, in turn.

Note that this implementation will throw an `UnsupportedOperationException` unless `add` is overridden (assuming the specified collection is non-empty).

Reimplemented from **decaf::util::AbstractCollection< E >** (p. 110).

6.171.3.3 `template<typename E > virtual void decaf::util::concurrent-
::CopyOnWriteArraySet< E >::clear () [inline,
virtual]`

Removes all of the elements from this collection (optional operation).

The collection will be empty after this method returns.

This implementation iterates over this collection, removing each element using the - **Iterator.remove** (p. 1560) operation. Most implementations will probably choose to override this method for efficiency.

Note that this implementation will throw an UnsupportedOperationException if the iterator returned by this collection's iterator method does not implement the remove method and this collection is non-empty.

Exceptions

<i>Unsupported- OperationException</i>	if the clear operation is not supported by this collection
--	--

Reimplemented from **decaf::util::AbstractCollection< E >** (p. 111).

6.171.3.4 `template<typename E > virtual bool decaf::util::concurrent::CopyOn-
WriteArraySet< E >::contains (const E & value) const [inline,
virtual]`

Returns true if this collection contains the specified element.

More formally, returns true if and only if this collection contains at least one element e such that (value == NULL ? e == NULL : value == e).

Parameters

<i>value</i>	The value to check for presence in the collection.
--------------	--

Returns

true if there is at least one of the elements in the collection

Exceptions

<i>NullPointerException</i>	if the Collection (p. 851) contains pointers and the Collection (p. 851) does not allow for NULL elements (optional check).
-----------------------------	---

This implementation iterates over the elements in the collection, checking each element in turn for equality with the specified element.

This implementation iterates over the elements in the collection, checking each element in turn for equality with the specified element.

Reimplemented from **decaf::util::AbstractCollection< E >** (p. 112).

6.171.3.5 `template<typename E > virtual bool decaf::util::concurrent::CopyOnWrite-
ArraySet< E >::containsAll (const Collection< E > & collection) const
[inline, virtual]`

Returns true if this collection contains all of the elements in the specified collection.

Parameters

<i>collection</i>	The Collection (p. 851) to compare to this one.
-------------------	--

Exceptions

<i>NullPointerException</i>	if the Collection (p. 851) contains pointers and the Collection (p. 851) does not allow for NULL elements (optional check).
-----------------------------	---

This implementation iterates over the specified collection, checking each element returned by the iterator in turn to see if it's contained in this collection. If all elements are so contained true is returned, otherwise false.

This implementation iterates over the specified collection, checking each element returned by the iterator in turn to see if it's contained in this collection. If all elements are so contained true is returned, otherwise false.

Reimplemented from **decaf::util::AbstractCollection< E >** (p. 113).

6.171.3.6 `template<typename E > virtual void decaf::util::concurrent::CopyOnWrite-
ArraySet< E >::copy (const Collection< E > & collection) [inline,
virtual]`

Renders this **Collection** (p. 851) as a Copy of the given **Collection** (p. 851).

The default implementation iterates over the contents of the given collection adding each to this collection after first calling this **Collection** (p. 851)'s clear method.

Parameters

<i>collection</i>	The collection to mirror.
-------------------	---------------------------

Exceptions

<i>UnsupportedOperationException</i>	if this is an unmodifiable collection.
<i>IllegalStateException</i>	if the elements cannot be added at this time due to insertion restrictions.

Reimplemented from **decaf::util::AbstractCollection< E >** (p. 113).

Referenced by `decaf::util::concurrent::CopyOnWriteArraySet< E >::CopyOnWriteArraySet()`.

6.171.3.7 `template<typename E> virtual bool decaf::util::concurrent::CopyOnWriteArraySet< E >::equals (const Collection< E > & collection) const`
[inline, virtual]

Answers true if this **Collection** (p. 851) and the one given are the same size and if each element contained in the **Collection** (p. 851) given is equal to an element contained in this collection.

Parameters

<i>collection</i>	- The Collection (p. 851) to be compared to this one.
-------------------	--

Returns

true if this **Collection** (p. 851) is equal to the one given.

Reimplemented from **decaf::util::AbstractCollection< E >** (p. 114).

References `decaf::lang::Iterable< E >::iterator()`, `decaf::util::concurrent::CopyOnWriteArraySet< E >::size()`, and `decaf::util::Collection< E >::size()`.

6.171.3.8 `template<typename E> virtual bool decaf::util::concurrent::CopyOnWriteArraySet< E >::isEmpty () const` [inline, virtual]

Returns true if this collection contains no elements.

This implementation returns `size()` (p. 1061) == 0.

Returns

true if the size method return 0.

Reimplemented from **decaf::util::AbstractCollection< E >** (p. 114).

6.171.3.9 `template<typename E> virtual decaf::util::Iterator<E>* decaf::util::concurrent::CopyOnWriteArraySet< E >::iterator ()`
[inline, virtual]

Returns

an iterator over a set of elements of type T.

Implements **decaf::lang::Iterable< E >** (p. 1557).

6.171.3.10 `template<typename E> virtual decaf::util::Iterator<E>* decaf::util::concurrent::CopyOnWriteArraySet< E >::iterator () const`
[inline, virtual]

Implements **decaf::lang::Iterable< E >** (p. 1558).

6.171.3.11 `template<typename E> virtual bool decaf::util::concurrent::Copy-On-WriteArraySet< E>::remove (const E & value) [inline, virtual]`

Removes a single instance of the specified element from the collection.

More formally, removes an element *e* such that (*value* == NULL ? *e* == NULL : *value* == *e*), if this collection contains one or more such elements. Returns true if this collection contained the specified element (or equivalently, if this collection changed as a result of the call).

Parameters

<i>value</i>	The reference to the element to remove from this Collection (p. 851).
--------------	--

Returns

true if the collection was changed, false otherwise.

Exceptions

<i>UnsupportedOperationException</i>	if this is an unmodifiable collection.
<i>NullPointerException</i>	if the Collection (p. 851) is a container of pointers and does not allow NULL values.

This implementation iterates over the collection looking for the specified element. If it finds the element, it removes the element from the collection using the iterator's remove method.

Note that this implementation throws an `UnsupportedOperationException` if the iterator returned by this collection's iterator method does not implement the remove method and this collection contains the specified object.

This implementation iterates over the collection looking for the specified element. If it finds the element, it removes the element from the collection using the iterator's remove method.

Note that this implementation throws an `UnsupportedOperationException` if the iterator returned by this collection's iterator method does not implement the remove method and this collection contains the specified object.

Reimplemented from `decaf::util::AbstractCollection< E>` (p. 116).

6.171.3.12 `template<typename E> virtual bool decaf::util::concurrent::CopyOn-WriteArraySet< E>::removeAll (const Collection< E> & collection) [inline, virtual]`

Removes all this collection's elements that are also contained in the specified collection (optional operation).

After this call returns, this collection will contain no elements in common with the specified collection.

Parameters

<i>collection</i>	The Collection (p. 851) whose elements are to be removed from this one.
-------------------	--

Returns

true if the collection changed as a result of this call.

Exceptions

<i>UnsupportedOperationException</i>	if this is an unmodifiable collection.
<i>NullPointerException</i>	if the Collection (p. 851) is a container of pointers and does not allow NULL values.

This implementation iterates over this collection, checking each element returned by the iterator in turn to see if it's contained in the specified collection. If it's so contained, it's removed from this collection with the iterator's remove method.

Note that this implementation will throw an `UnsupportedOperationException` if the iterator returned by the iterator method does not implement the remove method and this collection contains one or more elements in common with the specified collection.

This implementation determines which is the smaller of this set and the specified collection, by invoking the size method on each. If this set has fewer elements, then the implementation iterates over this set, checking each element returned by the iterator in turn to see if it is contained in the specified collection. If it is so contained, it is removed from this set with the iterator's remove method. If the specified collection has fewer elements, then the implementation iterates over the specified collection, removing from this set each element returned by the iterator, using this set's remove method.

Note that this implementation will throw an `UnsupportedOperationException` if the iterator returned by the iterator method does not implement the remove method.

This implementation determines which is the smaller of this set and the specified collection, by invoking the size method on each. If this set has fewer elements, then the implementation iterates over this set, checking each element returned by the iterator in turn to see if it is contained in the specified collection. If it is so contained, it is removed from this set with the iterator's remove method. If the specified collection has fewer elements, then the implementation iterates over the specified collection, removing from this set each element returned by the iterator, using this set's remove method.

Note that this implementation will throw an `UnsupportedOperationException` if the iterator returned by the iterator method does not implement the remove method.

Reimplemented from **decaf::util::AbstractSet< E >** (p. 153).

6.171.3.13 `template<typename E> virtual bool decaf::util::concurrent::CopyOnWriteArraySet< E>::retainAll (const Collection< E> & collection)`
`[inline, virtual]`

Retains only the elements in this collection that are contained in the specified collection (optional operation).

In other words, removes from this collection all of its elements that are not contained in the specified collection.

Parameters

<i>collection</i>	The Collection (p. 851) whose elements are to be retained.
-------------------	---

Returns

true if the collection changed as a result of this call.

Exceptions

<i>UnsupportedOperationException</i>	if this is an unmodifiable collection.
<i>NullPointerException</i>	if the Collection (p. 851) is a container of pointers and does not allow NULL values.

This implementation iterates over this collection, checking each element returned by the iterator in turn to see if it's contained in the specified collection. If it's not so contained, it's removed from this collection with the iterator's remove method.

Note that this implementation will throw an UnsupportedOperationException if the iterator returned by the iterator method does not implement the remove method and this collection contains one or more elements not present in the specified collection.

This implementation iterates over this collection, checking each element returned by the iterator in turn to see if it's contained in the specified collection. If it's not so contained, it's removed from this collection with the iterator's remove method.

Note that this implementation will throw an UnsupportedOperationException if the iterator returned by the iterator method does not implement the remove method and this collection contains one or more elements not present in the specified collection.

Reimplemented from **decaf::util::AbstractCollection< E>** (p. 118).

6.171.3.14 `template<typename E> virtual int decaf::util::concurrent::CopyOnWriteArraySet< E>::size () const` `[inline, virtual]`

Returns the number of elements in this collection.

If this collection contains more than Integer::MAX_VALUE elements, returns Integer::MAX_VALUE.

Returns

the number of elements in this collection

Implements **decaf::util::Collection**< **E** > (p. 864).

Referenced by **decaf::util::concurrent::CopyOnWriteArraySet**< **E** >::CopyOnWriteArraySet(), and **decaf::util::concurrent::CopyOnWriteArraySet**< **E** >::equals().

6.171.3.15 `template<typename E> virtual std::vector<E> decaf::util::concurrent-
::CopyOnWriteArraySet< E >::toArray () const [inline,
virtual]`

Answers an STL vector containing copies of all elements contained in this **Collection** (p. 851).

All the elements in the array will not be referenced by the collection. The elements in the returned array will be sorted to the same order as those returned by the iterator of this collection itself if the collection guarantees the order.

Returns

an vector of copies of all the elements from this **Collection** (p. 851)

Reimplemented from **decaf::util::AbstractCollection**< **E** > (p. 119).

The documentation for this class was generated from the following file:

- `src/main/decaf/util/concurrent/CountDownLatch.h`

6.172 decaf::util::concurrent::CountDownLatch Class Reference

```
#include <src/main/decaf/util/concurrent/CountDownLatch.h>
```

Public Member Functions

- **CountDownLatch** (int count)

Constructor.

- virtual **~CountDownLatch** ()
- virtual void **await** ()

Causes the current thread to wait until the latch has counted down to zero, unless the thread is interrupted.

- virtual bool **await** (long long timeout)

Causes the current thread to wait until the latch has counted down to zero, unless the thread is interrupted, or the specified waiting time elapses.

- virtual bool **await** (long long timeout, const **TimeUnit** &unit)

Causes the current thread to wait until the latch has counted down to zero, unless the thread is interrupted, or the specified waiting time elapses.

- virtual void **countDown** ()

Counts down the latch, releasing all waiting threads when the count hits zero.

- virtual int **getCount** () const

Gets the current count.

6.172.1 Constructor & Destructor Documentation

6.172.1.1 **decaf::util::concurrent::CountDownLatch::CountDownLatch (int count)**

Constructor.

Parameters

<i>count</i>	- number to count down from.
--------------	------------------------------

6.172.1.2 **virtual decaf::util::concurrent::CountDownLatch::~~CountDownLatch ()** [virtual]

6.172.2 Member Function Documentation

6.172.2.1 **virtual void decaf::util::concurrent::CountDownLatch::await ()** [virtual]

Causes the current thread to wait until the latch has counted down to zero, unless the thread is interrupted.

If the current count is zero then this method returns immediately.

If the current count is greater than zero then the current thread becomes disabled for thread scheduling purposes and lies dormant until one of two things happen:

- * The count reaches zero due to invocations of the **countDown()** (p. 1065) method; or
- * Some other thread interrupts the current thread.

If the current thread:

- * has its interrupted status set on entry to this method; or
 - * is interrupted while waiting,
- then InterruptedException is thrown and the current thread's interrupted status is cleared.

Exceptions

<i>InterruptedException</i>	- if the current thread is interrupted while waiting.
<i>Exception</i>	- if any other error occurs.

6.172.2.2 `virtual bool decaf::util::concurrent::CountDownLatch::await (long long
timeOut) [virtual]`

Causes the current thread to wait until the latch has counted down to zero, unless the thread is interrupted, or the specified waiting time elapses.

If the current count is zero then this method returns immediately with the value true.

If the current count is greater than zero then the current thread becomes disabled for thread scheduling purposes and lies dormant until one of three things happen:

* The count reaches zero due to invocations of the **countDown()** (p. 1065) method; or * Some other thread interrupts the current thread; or * The specified waiting time elapses.

If the count reaches zero then the method returns with the value true.

If the current thread:

* has its interrupted status set on entry to this method; or * is interrupted while waiting, then InterruptedException is thrown and the current thread's interrupted status is cleared.

If the specified waiting time elapses then the value false is returned. If the time is less than or equal to zero, the method will not wait at all.

Parameters

<i>timeout</i>	- Time in milliseconds to wait for the count to reach zero.
----------------	---

Exceptions

<i>InterruptedException</i>	- if the current thread is interrupted while waiting.
<i>Exception</i>	- if any other error occurs.

6.172.2.3 `virtual bool decaf::util::concurrent::CountDownLatch::await (long long
timeout, const TimeUnit & unit) [virtual]`

Causes the current thread to wait until the latch has counted down to zero, unless the thread is interrupted, or the specified waiting time elapses.

If the current count is zero then this method returns immediately with the value true.

If the current count is greater than zero then the current thread becomes disabled for thread scheduling purposes and lies dormant until one of three things happen:

* The count reaches zero due to invocations of the **countDown()** (p. 1065) method; or * Some other thread interrupts the current thread; or * The specified waiting time elapses.

If the count reaches zero then the method returns with the value true.

If the current thread:

* has its interrupted status set on entry to this method; or * is interrupted while waiting, then InterruptedException is thrown and the current thread's interrupted status is

cleared.

If the specified waiting time elapses then the value false is returned. If the time is less than or equal to zero, the method will not wait at all.

Parameters

<i>timeout</i>	- Time to wait for the count to reach zero.
<i>unit</i>	- The units that the timeout specifies.

Exceptions

<i>InterruptedException</i>	- if the current thread is interrupted while waiting.
<i>Exception</i>	- if any other error occurs.

6.172.2.4 `virtual void decaf::util::concurrent::CountDownLatch::countDown ()`
[virtual]

Counts down the latch, releasing all waiting threads when the count hits zero.

6.172.2.5 `virtual int decaf::util::concurrent::CountDownLatch::getCount () const`
[inline, virtual]

Gets the current count.

Returns

int count value

The documentation for this class was generated from the following file:

- src/main/decaf/util/concurrent/**CountDownLatch.h**

6.173 decaf::util::zip::CRC32 Class Reference

Class that can be used to compute a CRC-32 checksum for a data stream.

```
#include <src/main/decaf/util/zip/CRC32.h>
```

Inheritance diagram for decaf::util::zip::CRC32:

Public Member Functions

- **CRC32** ()

- virtual **~CRC32** ()
- virtual long long **getValue** () const
- virtual void **reset** ()

Reset the checksum to its initial value.
- virtual void **update** (const std::vector< unsigned char > &buffer)

Updates the current checksum with the specified vector of bytes.
- virtual void **update** (const std::vector< unsigned char > &buffer, int offset, int length)

Updates the current checksum with the specified array of bytes.
- virtual void **update** (const unsigned char *buffer, int size, int offset, int length)

Updates the current checksum with the specified array of bytes.
- virtual void **update** (int byte)

Updates the current checksum with the specified byte value.

6.173.1 Detailed Description

Class that can be used to compute a CRC-32 checksum for a data stream.

Since

1.0

6.173.2 Constructor & Destructor Documentation

6.173.2.1 **decaf::util::zip::CRC32::CRC32** ()

6.173.2.2 **virtual decaf::util::zip::CRC32::~~CRC32** () [virtual]

6.173.3 Member Function Documentation

6.173.3.1 **virtual long long decaf::util::zip::CRC32::getValue** () const [virtual]

Returns

the current checksum value.

Implements **decaf::util::zip::Checksum** (p.811).

6.173.3.2 **virtual void decaf::util::zip::CRC32::reset** () [virtual]

Reset the checksum to its initial value.

Implements **decaf::util::zip::Checksum** (p.811).

6.173.3.3 `virtual void decaf::util::zip::CRC32::update (const std::vector< unsigned char > & buffer) [virtual]`

Updates the current checksum with the specified vector of bytes.

Parameters

<i>buffer</i>	The buffer to read the updated bytes from.
---------------	--

Implements **decaf::util::zip::Checksum** (p. 811).

6.173.3.4 `virtual void decaf::util::zip::CRC32::update (const std::vector< unsigned char > & buffer, int offset, int length) [virtual]`

Updates the current checksum with the specified array of bytes.

Parameters

<i>buffer</i>	The buffer to read the updated bytes from.
<i>offset</i>	The position in the buffer to start reading.
<i>length</i>	The amount of data to read from the byte buffer.

Exceptions

<i>IndexOutOfBoundsException</i>	if $\text{offset} + \text{length} > \text{size of the buffer}$.
----------------------------------	--

Implements **decaf::util::zip::Checksum** (p. 811).

6.173.3.5 `virtual void decaf::util::zip::CRC32::update (const unsigned char * buffer, int size, int offset, int length) [virtual]`

Updates the current checksum with the specified array of bytes.

Parameters

<i>buffer</i>	The buffer to read the updated bytes from.
<i>size</i>	The size of the passed buffer.
<i>offset</i>	The position in the buffer to start reading.
<i>length</i>	The amount of data to read from the byte buffer.

Exceptions

<i>NullPointerException</i>	if the passed buffer is NULL.
<i>IndexOutOfBoundsException</i>	if $\text{offset} + \text{length} > \text{size of the buffer}$.

Implements **decaf::util::zip::Checksum** (p. 812).

6.173.3.6 virtual void `decaf::util::zip::CRC32::update (int byte)` [virtual]

Updates the current checksum with the specified byte value.

Parameters

<i>byte</i>	The byte value to update the current Checksum (p. 810) with (0..255).
-------------	--

Implements `decaf::util::zip::Checksum` (p. 812).

The documentation for this class was generated from the following file:

- `src/main/decaf/util/zip/CRC32.h`

6.174 `ct_data_s` Struct Reference

```
#include <src/main/decaf/internal/util/zip/deflate.h>
```

Data Fields

- union {
 ush freq
 ush code
} **fc**
- union {
 ush dad
 ush len
} **dl**

6.174.1 Field Documentation

6.174.1.1 `ush ct_data_s::code`

6.174.1.2 `ush ct_data_s::dad`

6.174.1.3 `union { ... } ct_data_s::dl`

6.174.1.4 `union { ... } ct_data_s::fc`

6.174.1.5 `ush ct_data_s::freq`

6.174.1.6 `ush ct_data_s::len`

The documentation for this struct was generated from the following file:

- src/main/decaf/internal/util/zip/deflate.h

6.175 activemq::commands::DataArrayResponse Class Reference

```
#include <src/main/activemq/commands/DataArrayResponse.h>
```

Inheritance diagram for activemq::commands::DataArrayResponse:

Public Member Functions

- **DataArrayResponse** ()
- virtual **~DataArrayResponse** ()
- virtual unsigned char **getDataStructureType** () const
*Get the **DataStructure** (p. 1133) Type as defined in CommandTypes.h.*
- virtual **DataArrayResponse** * **cloneDataStructure** () const
Clone this object and return a new instance that the caller now owns, this will be an exact copy of this one.
- virtual void **copyDataStructure** (const **DataStructure** *src)
Copy the contents of the passed object into this objects members, overwriting any existing data.
- virtual std::string **toString** () const
*Returns a string containing the information for this **DataStructure** (p. 1133) such as its type and value of its elements.*
- virtual bool **equals** (const **DataStructure** *value) const
*Compares the **DataStructure** (p. 1133) passed in to this one, and returns if they are equivalent.*
- virtual const std::vector < **decaf::lang::Pointer** < **DataStructure** > > & **getData** () const
- virtual std::vector < **decaf::lang::Pointer** < **DataStructure** > > & **getData** ()
- virtual void **setData** (const std::vector< **decaf::lang::Pointer**< **DataStructure** > > &data)

Static Public Attributes

- static const unsigned char **ID_DATAARRAYRESPONSE** = 33

Protected Attributes

- std::vector < **decaf::lang::Pointer** < **DataStructure** > > **data**

6.175.1 Constructor & Destructor Documentation

6.175.1.1 `activemq::commands::DataArrayResponse::DataArrayResponse ()`

6.175.1.2 `virtual activemq::commands::DataArrayResponse::~~DataArrayResponse () [virtual]`

6.175.2 Member Function Documentation

6.175.2.1 `virtual DataArrayResponse* activemq::commands::DataArrayResponse::cloneDataStructure () const [virtual]`

Clone this object and return a new instance that the caller now owns, this will be an exact copy of this one.

Returns

new copy of this object.

Reimplemented from `activemq::commands::Response` (p. 2299).

6.175.2.2 `virtual void activemq::commands::DataArrayResponse::copyDataStructure (const DataStructure * src) [virtual]`

Copy the contents of the passed object into this objects members, overwriting any existing data.

Returns

src - Source Object

Reimplemented from `activemq::commands::Response` (p. 2299).

6.175.2.3 `virtual bool activemq::commands::DataArrayResponse::equals (const DataStructure * value) const [virtual]`

Compares the `DataStructure` (p. 1133) passed in to this one, and returns if they are equivalent.

Equivalent here means that they are of the same type, and that each element of the objects are the same.

Returns

true if `DataStructure` (p. 1133)'s are Equal.

Reimplemented from `activemq::commands::Response` (p. 2299).

- 6.175.2.4 `virtual const std::vector< decaf::lang::Pointer<DataStructure> >& activemq::commands::DataArrayResponse::getData () const`
[virtual]
- 6.175.2.5 `virtual std::vector< decaf::lang::Pointer<DataStructure> >& activemq::commands::DataArrayResponse::getData ()` [virtual]
- 6.175.2.6 `virtual unsigned char activemq::commands::DataArrayResponse::getDataStructureType () const`
[virtual]

Get the **DataStructure** (p. 1133) Type as defined in CommandTypes.h.

Returns

The type of the data structure

Reimplemented from **activemq::commands::Response** (p. 2300).

- 6.175.2.7 `virtual void activemq::commands::DataArrayResponse::setData (const std::vector< decaf::lang::Pointer< DataStructure > > & data)`
[virtual]
- 6.175.2.8 `virtual std::string activemq::commands::DataArrayResponse::toString () const` [virtual]

Returns a string containing the information for this **DataStructure** (p. 1133) such as its type and value of its elements.

Returns

formatted string useful for debugging.

Reimplemented from **activemq::commands::Response** (p. 2300).

6.175.3 Field Documentation

- 6.175.3.1 `std::vector< decaf::lang::Pointer<DataStructure> > activemq::commands::DataArrayResponse::data` [protected]
- 6.175.3.2 `const unsigned char activemq::commands::DataArrayResponse::ID_DATAARRAYRESPONSE = 33`
[static]

The documentation for this class was generated from the following file:

- src/main/activemq/commands/**DataArrayResponse.h**

6.176 activemq::wireformat::openwire::marshal::generated::DataArrayResponseMarshaller Class

Reference

1073

6.176 activemq::wireformat::openwire::marshal::generated::DataArrayResponseMarshaller Class Reference

Marshaling code for Open Wire Format for **DataArrayResponseMarshaller** (p. 1072).

```
#include <src/main/activemq/wireformat/openwire/marshal/generated/-  
DataArrayResponseMarshaller.h>
```

Inheritance diagram for activemq::wireformat::openwire::marshal::generated::DataArrayResponseMarshaller:

Public Member Functions

- **DataArrayResponseMarshaller** ()
- virtual ~**DataArrayResponseMarshaller** ()
- virtual **commands::DataStructure** * **createObject** () const
Creates a new instance of the class that this class is a marshaling director for.
- virtual unsigned char **getDataStructureType** () const
Gets the DataStructureType that this class marshals/unmarshals.
- virtual void **tightUnmarshal** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataInputStream** *dataIn, **utils::BooleanStream** *bs)
Tight Un-marhsal to the given stream.
- virtual int **tightMarshal1** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **utils::BooleanStream** *bs)
Tight Marshal to the given stream.
- virtual void **tightMarshal2** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataOutputStream** *dataOut, **utils::BooleanStream** *bs)
Tight Marshal to the given stream.
- virtual void **looseUnmarshal** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataInputStream** *dataIn)
Loose Un-marhsal to the given stream.
- virtual void **looseMarshal** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataOutputStream** *dataOut)
Tight Marshal to the given stream.

6.176.1 Detailed Description

Marshaling code for Open Wire Format for **DataArrayResponseMarshaller** (p. 1072).

NOTE!: This file is auto generated - do not modify! if you need to make a change, please see the Java Classes in the activemq-openwire-generator module

6.176.2 Constructor & Destructor Documentation

6.176.2.1 **activemq::wireformat::openwire::marshal::generated::Data-
ArrayResponseMarshaller::DataArrayResponseMarshaller ()**
[inline]

6.176.2.2 **virtual activemq::wireformat::openwire::marshal::generated::DataArray-
ResponseMarshaller::~~DataArrayResponseMarshaller ()** [inline,
virtual]

6.176.3 Member Function Documentation

6.176.3.1 **virtual commands::DataStructure* activemq::wireformat::openwire-
::marshal::generated::DataArrayResponseMarshaller::createObject ()**
const [virtual]

Creates a new instance of the class that this class is a marshaling director for.

Returns

newly allocated Command

Reimplemented from **activemq::wireformat::openwire::marshal::generated::-
ResponseMarshaller** (p. 2308).

6.176.3.2 **virtual unsigned char activemq::wireformat::openwire::marshal::generated-
::DataArrayResponseMarshaller::getDataStructureType () const**
[virtual]

Gets the DataStructureType that this class marshals/unmarshals.

Returns

byte Id of this classes DataStructureType

Reimplemented from **activemq::wireformat::openwire::marshal::generated::-
ResponseMarshaller** (p. 2309).

6.176.3.3 **virtual void activemq::wireformat::openwire::marshal::generated::Data-
ArrayResponseMarshaller::looseMarshal (OpenWireFormat * *format*,
commands::DataStructure * *command*, decaf::io::DataOutputStream * *ds*
)** [virtual]

Tight Marshal to the given stream.

Parameters

<i>format</i>	- The OpenWireFormat properties
<i>command</i>	- the object to Marshal
<i>ds</i>	- DataOutputStream to marshal to

6.176 activemq::wireformat::openwire::marshal::generated::DataArray-ResponseMarshaller Class

Reference

1075

Exceptions

<i>IOException</i>	if an error occurs.
--------------------	---------------------

Reimplemented from **activemq::wireformat::openwire::marshal::generated::ResponseMarshaller** (p. 2309).

6.176.3.4 **virtual void activemq::wireformat::openwire::marshal::generated::DataArrayResponseMarshaller::looseUnmarshal (OpenWireFormat * *format*, commands::DataStructure * *command*, decaf::io::DataInputStream * *dis*)**
[virtual]

Loose Un-marhsal to the given stream.

Parameters

<i>format</i>	- The OpenWireFormat properties
<i>command</i>	- the object to Un-Marshal
<i>dis</i>	- the DataInputStream to Un-Marshal from

Exceptions

<i>IOException</i>	if an error occurs.
--------------------	---------------------

Reimplemented from **activemq::wireformat::openwire::marshal::generated::ResponseMarshaller** (p. 2310).

6.176.3.5 **virtual int activemq::wireformat::openwire::marshal::generated::DataArrayResponseMarshaller::tightMarshal1 (OpenWireFormat * *format*, commands::DataStructure * *command*, utils::BooleanStream * *bs*)**
[virtual]

Tight Marhsal to the given stream.

Parameters

<i>format</i>	The OpenWireFormat properties
<i>command</i>	The object to Marshal
<i>bs</i>	The boolean stream to marshal to.

Exceptions

<i>IOException</i>	if an error occurs.
--------------------	---------------------

Reimplemented from **activemq::wireformat::openwire::marshal::generated::ResponseMarshaller** (p. 2310).

6.176.3.6 virtual void activemq::wireformat::openwire::marshal::generated::Data-
ArrayResponseMarshaller::tightMarshal2 (OpenWireFormat * *format*,
commands::DataStructure * *command*, decaf::io::DataOutputStream *
ds, utils::BooleanStream * *bs*) [virtual]

Tight Marhsal to the given stream.

Parameters

<i>format</i>	- The OpenWireFormat properties
<i>command</i>	- the object to Marshal
<i>ds</i>	- the DataOutputStream to Marshal to
<i>bs</i>	- boolean stream to marshal to.

Exceptions

<i>IOException</i>	if an error occurs.
--------------------	---------------------

Reimplemented from **activemq::wireformat::openwire::marshal::generated::ResponseMarshaller** (p.2311).

6.176.3.7 virtual void activemq::wireformat::openwire::marshal::generated::Data-
ArrayResponseMarshaller::tightUnmarshal (OpenWireFormat * *format*,
commands::DataStructure * *command*, decaf::io::DataInputStream * *dis*,
utils::BooleanStream * *bs*) [virtual]

Tight Un-marhsal to the given stream.

Parameters

<i>format</i>	- The OpenWireFormat properties
<i>command</i>	- the object to Un-Marshal
<i>dis</i>	- the DataInputStream to Un-Marshal from
<i>bs</i>	- boolean stream to unmarshal from.

Exceptions

<i>IOException</i>	if an error occurs.
--------------------	---------------------

Reimplemented from **activemq::wireformat::openwire::marshal::generated::ResponseMarshaller** (p.2311).

The documentation for this class was generated from the following file:

- src/main/activemq/wireformat/openwire/marshal/generated/**DataArrayResponseMarshaller.h**

6.177 decaf::util::zip::DataFormatException Class Reference

```
#include <src/main/decaf/util/zip/DataFormatException.h>
```

Inheritance diagram for decaf::util::zip::DataFormatException:

Public Member Functions

- **DataFormatException** () throw ()
Default Constructor.
- **DataFormatException** (const lang::Exception &ex) throw ()
Copy Constructor.
- **DataFormatException** (const DataFormatException &ex) throw ()
Copy Constructor.
- **DataFormatException** (const char *file, const int lineNumber, const std::exception *cause, const char *msg,...) throw ()
Constructor - Initializes the file name and line number where this message occurred.
- **DataFormatException** (const std::exception *cause) throw ()
Constructor.
- **DataFormatException** (const char *file, const int lineNumber, const char *msg,...) throw ()
Constructor.
- virtual **DataFormatException** * clone () const
Clones this exception.
- virtual ~**DataFormatException** () throw ()

6.177.1 Constructor & Destructor Documentation

6.177.1.1 decaf::util::zip::DataFormatException::DataFormatException () throw ()

Default Constructor.

6.177.1.2 decaf::util::zip::DataFormatException::DataFormatException (const lang::Exception & ex) throw () [inline]

Copy Constructor.

Parameters

ex	the exception to copy
----	-----------------------

6.177.1.3 **decaf::util::zip::DataFormatException::DataFormatException (const DataFormatException & ex) throw ()** [inline]

Copy Constructor.

Parameters

<i>ex</i>	the exception to copy, which is an instance of this type
-----------	--

6.177.1.4 **decaf::util::zip::DataFormatException::DataFormatException (const char * file, const int lineNumber, const std::exception * cause, const char * msg, ...) throw ()** [inline]

Constructor - Initializes the file name and line number where this message occurred.

Sets the message to report, using an optional list of arguments to parse into the message

Parameters

<i>file</i>	The file name where exception occurs
<i>lineNumber</i>	The line number where the exception occurred.
<i>cause</i>	The exception that was the cause for this one to be thrown.
<i>msg</i>	The message to report
<i>...</i>	list of primitives that are formatted into the message

6.177.1.5 **decaf::util::zip::DataFormatException::DataFormatException (const std::exception * cause) throw ()** [inline]

Constructor.

Parameters

<i>cause</i>	Pointer to the exception that caused this one to be thrown, the object is cloned caller retains ownership.
--------------	--

6.177.1.6 **decaf::util::zip::DataFormatException::DataFormatException (const char * file, const int lineNumber, const char * msg, ...) throw ()** [inline]

Constructor.

Parameters

<i>file</i>	The file name where exception occurs
<i>lineNumber</i>	The line number where the exception occurred.
<i>msg</i>	The message to report
<i>...</i>	list of primitives that are formatted into the message

6.177.1.7 virtual **decaf::util::zip::DataFormatException::~~DataFormatException** (
) throw () [virtual]

6.177.2 Member Function Documentation

6.177.2.1 virtual **DataFormatException*** **decaf::util::zip::Data-**
FormatException::clone () const [inline,
 virtual]

Clones this exception.

This is useful for cases where you need to preserve the type of the original exception as well as the message. All subclasses should override.

Returns

a new instance of an Exception that is a copy of this instance.

Reimplemented from **decaf::lang::Exception** (p. 1282).

The documentation for this class was generated from the following file:

- src/main/decaf/util/zip/**DataFormatException.h**

6.178 decaf::net::DatagramPacket Class Reference

Class that represents a single datagram packet.

```
#include <src/main/decaf/net/DatagramPacket.h>
```

Public Member Functions

- **DatagramPacket** (unsigned char *bytes, int size, int length)
*Creates a new **DatagramPacket** (p. 1078) for use in receiving a packet of the given length.*
- **DatagramPacket** (unsigned char *bytes, int size, int offset, int length)
*Creates a new **DatagramPacket** (p. 1078) for use in receiving a packet of the given length starting at the specified offset into the buffer.*
- **DatagramPacket** (unsigned char *bytes, int size, int offset, int length, const **InetAddress** &address, int port)
*Creates a new **DatagramPacket** (p. 1078) for use in sending a packet of the given length starting at the specified offset into the buffer to the specified host on the specified port.*
- **DatagramPacket** (unsigned char *bytes, int size, int length, const **InetAddress** &address, int port)
*Creates a new **DatagramPacket** (p. 1078) for use in sending a packet of the given length to the specified host on the specified port.*

- **DatagramPacket** (unsigned char *bytes, int size, int length, const **SocketAddress** &address)

*Creates a new **DatagramPacket** (p. 1078) for use in sending a packet of the given length into the buffer to the specified socket address.*

- **DatagramPacket** (unsigned char *bytes, int size, int offset, int length, const **SocketAddress** &address)

*Creates a new **DatagramPacket** (p. 1078) for use in sending a packet of the given length starting at the specified offset into the buffer to the specified socket address.*

- virtual **~DatagramPacket** ()
- **InetAddress** * **getAddress** () const
- void **setAddress** (const **InetAddress** &address)

Sets the IP address of the machine to which this datagram is being sent.

- **SocketAddress** * **getSocketAddress** () const

*Gets the **SocketAddress** (p. 2470) (usually IP address + port number) of the remote host that this packet is being sent to or is coming from.*

- void **setSocketAddress** (const **SocketAddress** &address)

*Sets the **SocketAddress** (p. 2470) (usually IP address + port number) of the remote host to which this datagram is being sent.*

- int **getPort** () const
- void **setPort** (int port)

Sets the port number on the remote host to which this datagram is being sent.

- int **getOffset** () const
- void **setOffset** (int offset)

Sets the offset into the data buffer where data to sent is to be read from or where the data that is received should start writing to.

- int **getLength** () const
- void **setLength** (int length)

Set the length for this packet.

- unsigned char * **getData** () const
- int **getSize** () const
- void **setData** (unsigned char *buffer, int size)

Set the data buffer for this packet.

- void **setData** (unsigned char *buffer, int size, int offset, int length)

Set the data buffer for this packet.

6.178.1 Detailed Description

Class that represents a single datagram packet.

Datagrams are sent in packets from machine to machine and can each be routed differently and can arrive in any order. Delivery of a packet is not guaranteed.

Since

1.0

6.178.2 Constructor & Destructor Documentation

6.178.2.1 decaf::net::DatagramPacket::DatagramPacket (unsigned char * *bytes*, int *size*, int *length*)

Creates a new **DatagramPacket** (p. 1078) for use in receiving a packet of the given length.

Parameters

<i>bytes</i>	The array of bytes to hold the incoming datagram buffer.
<i>size</i>	The size of the supplied byte array.
<i>length</i>	The number of byte to read starting at the supplied offset.

Exceptions

<i>NullPointerException</i>	if the pointer to the buffer is NULL.
<i>IndexOutOfBoundsException</i>	if the number of bytes to read exceeds the buffer size.

6.178.2.2 decaf::net::DatagramPacket::DatagramPacket (unsigned char * *bytes*, int *size*, int *offset*, int *length*)

Creates a new **DatagramPacket** (p. 1078) for use in receiving a packet of the given length starting at the specified offset into the buffer.

Parameters

<i>bytes</i>	The array of bytes to hold the incoming datagram buffer.
<i>size</i>	The size of the supplied byte array.
<i>offset</i>	The position in the array to start writing to.
<i>length</i>	The number of byte to read starting at the supplied offset.

Exceptions

<i>NullPointerException</i>	if the pointer to the buffer is NULL.
<i>IndexOutOfBoundsException</i>	if the number of bytes to copy exceeds the buffer size.

6.178.2.3 decaf::net::DatagramPacket::DatagramPacket (unsigned char * *bytes*, int *size*, int *offset*, int *length*, const InetAddress & *address*, int *port*)

Creates a new **DatagramPacket** (p. 1078) for use in sending a packet of the given length starting at the specified offset into the buffer to the specified host on the specified port.

Parameters

<i>bytes</i>	The array of bytes to hold the outgoing datagram buffer.
<i>size</i>	The size of the supplied byte array.
<i>offset</i>	The position in the array to start writing to.
<i>length</i>	The number of byte to read starting at the supplied offset.
<i>address</i>	The Address to send the packet to
<i>port</i>	The port on the destination that is to receive this packet.

Exceptions

<i>NullPointerException</i>	if the pointer to the buffer is NULL.
<i>IndexOutOfBoundsException</i>	if the number of bytes to copy exceeds the buffer size.

6.178.2.4 `decaf::net::DatagramPacket::DatagramPacket (unsigned char * bytes, int size, int length, const InetAddress & address, int port)`

Creates a new **DatagramPacket** (p. 1078) for use in sending a packet of the given length to the specified host on the specified port.

Parameters

<i>bytes</i>	The array of bytes to hold the outgoing datagram buffer.
<i>size</i>	The size of the supplied byte array.
<i>length</i>	The number of byte to read starting at the supplied offset.
<i>address</i>	The Address to send the packet to
<i>port</i>	The port on the destination that is to receive this packet.

Exceptions

<i>NullPointerException</i>	if the pointer to the buffer is NULL.
<i>IndexOutOfBoundsException</i>	if the number of bytes to copy exceeds the buffer size.

6.178.2.5 `decaf::net::DatagramPacket::DatagramPacket (unsigned char * bytes, int size, int length, const SocketAddress & address)`

Creates a new **DatagramPacket** (p. 1078) for use in sending a packet of the given length into the buffer to the specified socket address.

Parameters

<i>bytes</i>	The array of bytes to hold the outgoing datagram buffer.
<i>size</i>	The size of the supplied byte array.
<i>length</i>	The number of byte to read starting at the supplied offset.
<i>address</i>	The Address to send the packet to

Exceptions

<i>NullPointerException</i>	if the pointer to the buffer is NULL.
<i>IndexOutOfBoundsException</i>	if the number of bytes to copy exceeds the buffer size.

6.178.2.6 `decaf::net::DatagramPacket::DatagramPacket (unsigned char * bytes, int size, int offset, int length, const SocketAddress & address)`

Creates a new **DatagramPacket** (p. 1078) for use in sending a packet of the given length starting at the specified offset into the buffer to the specified socket address.

Parameters

<i>bytes</i>	The array of bytes to hold the outgoing datagram buffer.
<i>size</i>	The size of the supplied byte array.
<i>offset</i>	The position in the array to start writing to.
<i>length</i>	The number of byte to read starting at the supplied offset.
<i>address</i>	The Address to send the packet to

Exceptions

<i>NullPointerException</i>	if the pointer to the buffer is NULL.
<i>IndexOutOfBoundsException</i>	if the number of bytes to copy exceeds the buffer size.

6.178.2.7 `virtual decaf::net::DatagramPacket::~~DatagramPacket () [virtual]`

6.178.3 Member Function Documentation

6.178.3.1 `InetAddress* decaf::net::DatagramPacket::getAddress () const`

Returns

the IP address that this datagram packet is being sent to or was received from.

6.178.3.2 `unsigned char* decaf::net::DatagramPacket::getData () const`

Returns

the data buffer. The data received or the data to be sent starts from the offset in the buffer, and continues for length bytes.

6.178.3.3 `int decaf::net::DatagramPacket::getLength () const`

Returns

the length of the data to be sent or the length of the data received.

6.178.3.4 int decaf::net::DatagramPacket::getOffset () const**Returns**

the offset of the data to be sent or the offset of the data received.

6.178.3.5 int decaf::net::DatagramPacket::getPort () const**Returns**

the port number that this datagram packet is being sent to or was received from.

6.178.3.6 int decaf::net::DatagramPacket::getSize () const**Returns**

the size of the buffer used in this datagram packet.

6.178.3.7 SocketAddress* decaf::net::DatagramPacket::getSocketAddress () const

Gets the **SocketAddress** (p. 2470) (usually IP address + port number) of the remote host that this packet is being sent to or is coming from.

Returns

the **SocketAddress** (p. 2470) for this datagram packet.

6.178.3.8 void decaf::net::DatagramPacket::setAddress (const InetAddress & address)

Sets the IP address of the machine to which this datagram is being sent.

Parameters

<i>address</i>	The IP address.
----------------	-----------------

6.178.3.9 void decaf::net::DatagramPacket::setData (unsigned char * buffer, int size)

Set the data buffer for this packet.

With the offset of this **DatagramPacket** (p. 1078) set to 0, and the length set to the size value specified.

Parameters

<i>buffer</i>	The new data buffer to use for this datagram packet.
<i>size</i>	The size of the buffer.

Exceptions

<i>NullPointerException</i>	if the buffer pointer is NULL.
-----------------------------	--------------------------------

6.178.3.10 void decaf::net::DatagramPacket::setData (unsigned char * *buffer*, int *size*, int *offset*, int *length*)

Set the data buffer for this packet.

With the offset of this **DatagramPacket** (p. 1078) set to 0, and the length set to the size value specified.

Parameters

<i>buffer</i>	The new data buffer to use for this datagram packet.
<i>size</i>	The size of the buffer.
<i>offset</i>	The position in the buffer to read from or write to.
<i>length</i>	The number of bytes that will be read into the buffer or sent from the buffer.

Exceptions

<i>NullPointerException</i>	if the buffer pointer is NULL.
-----------------------------	--------------------------------

6.178.3.11 void decaf::net::DatagramPacket::setLength (int *length*)

Set the length for this packet.

The length of the packet is the number of bytes from the packet's data buffer that will be sent, or the number of bytes of the packet's data buffer that will be used for receiving data. The length must be lesser or equal to the offset plus the length of the packet's buffer.

Parameters

<i>length</i>	The length value to set for this packet.
---------------	--

Exceptions

<i>IllegalArgument-Exception</i>	if the value is negative or exceeds the data buffers length.
----------------------------------	--

6.178.3.12 void decaf::net::DatagramPacket::setOffset (int *offset*)

Sets the offset into the data buffer where data to sent is to be read from or where the data that is received should start writing to.

Parameters

<i>offset</i>	The buffer offset value.
---------------	--------------------------

Exceptions

<i>IllegalArgument-Exception</i>	if the offset value is greater than the buffer size.
----------------------------------	--

6.178.3.13 void decaf::net::DatagramPacket::setPort (int *port*)

Sets the port number on the remote host to which this datagram is being sent.

Parameters

<i>port</i>	The port on the remote host.
-------------	------------------------------

Exceptions

<i>IllegalArgument-Exception</i>	if the port value is not in the range [0..65535].
----------------------------------	---

6.178.3.14 void decaf::net::DatagramPacket::setSocketAddress (const **SocketAddress** & *address*)

Sets the **SocketAddress** (p. 2470) (usually IP address + port number) of the remote host to which this datagram is being sent.

Parameters

<i>address</i>	The SocketAddress (p. 2470) (IP + port) for this datagram packet.
----------------	--

Exceptions

<i>IllegalArgument-Exception</i>	if the subclass of address is not supported by this Socket (p. 2452).
----------------------------------	--

The documentation for this class was generated from the following file:

- src/main/decaf/net/**DatagramPacket.h**

6.179 decaf::io::DataInput Class Reference

The **DataInput** (p. 1086) interface provides for reading bytes from a binary stream and reconstructing from them data in any of the C++ primitive types.

```
#include <src/main/decaf/io/DataInput.h>
```

Public Member Functions

- virtual **~DataInput** ()
- virtual bool **readBoolean** ()=0
Reads in one byte and returns true if that byte is nonzero, false if that byte is zero.
- virtual char **readByte** ()=0
Reads and returns one input byte.
- virtual unsigned char **readUnsignedByte** ()=0
Reads one input byte, zero-extends it to type int, and returns the result, which is therefore in the range 0 through 255.
- virtual char **readChar** ()=0
Reads an input char and returns the char value.
- virtual double **readDouble** ()=0
Reads eight input bytes and returns a double value.
- virtual float **readFloat** ()=0
Reads four input bytes and returns a float value.
- virtual int **readInt** ()=0
Reads four input bytes and returns an int value.
- virtual long long **readLong** ()=0
Reads eight input bytes and returns a long value.
- virtual short **readShort** ()=0
Reads two input bytes and returns a short value.
- virtual unsigned short **readUnsignedShort** ()=0
Reads two input bytes and returns an int value in the range 0 through 65535.
- virtual std::string **readString** ()=0
Reads an NULL terminated ASCII string to the stream and returns the string to the caller.
- virtual std::string **readLine** ()=0
Reads the next line of text from the input stream.
- virtual std::string **readUTF** ()=0
Reads a modified UTF-8 encoded string in ASCII format and returns it, this is only useful if you know for sure that the string that is to be read was a string that contained all ASCII values (0-255), if so this method will throw a UTFFormatException.

- virtual void **readFully** (unsigned char *buffer, int size)=0
Reads some bytes from an input stream and stores them into the buffer array buffer.
- virtual void **readFully** (unsigned char *buffer, int size, int offset, int length)=0
Reads length bytes from an input stream.
- virtual long long **skipBytes** (long long num)=0
Makes an attempt to skip over n bytes of data from the input stream, discarding the skipped bytes.

6.179.1 Detailed Description

The **DataInput** (p. 1086) interface provides for reading bytes from a binary stream and reconstructing from them data in any of the C++ primitive types.

There is also a facility for reconstructing Strings from data in the Java standard modified UTF-8 format.

It is generally true of all the reading routines in this interface that if end of file is reached before the desired number of bytes has been read, an **EOFException** (p. 1275) is thrown. If any byte cannot be read for any reason other than end of file, an **IOException** (p. 1545) other than **EOFException** (p. 1275) is thrown. for example, an **IOException** (p. 1545) may be thrown if the underlying input stream has been closed.

See also

- DataOutput** (p. 1103)
- DataInputStream** (p. 1094)

Since

1.0

6.179.2 Constructor & Destructor Documentation

6.179.2.1 virtual decaf::io::DataInput::~~DataInput () [inline, virtual]

6.179.3 Member Function Documentation

6.179.3.1 virtual bool decaf::io::DataInput::readBoolean () [pure virtual]

Reads in one byte and returns true if that byte is nonzero, false if that byte is zero.

Returns

the boolean value of the read in byte (0=false, 1=true).

Exceptions

IOException (p. 1545)	if an I/O Error occurs.
EOFException (p. 1275)	if the end of input is reached

6.179.3.2 virtual char **decaf::io::DataInput::readByte**() [pure virtual]

Reads and returns one input byte.

The byte is treated as a signed value in the range -128 through 127, inclusive.

Returns

the 8-bit value read.

Exceptions

<i>IOException</i> (p. 1545)	if an I/O Error occurs.
<i>EOFException</i> (p. 1275)	if the end of input is reached.

6.179.3.3 virtual char **decaf::io::DataInput::readChar**() [pure virtual]

Reads an input char and returns the char value.

A ascii char is made up of one bytes. This returns the same result as `readByte`

Returns

the 8 bit char read.

Exceptions

<i>IOException</i> (p. 1545)	if an I/O Error occurs.
<i>EOFException</i> (p. 1275)	if the end of input is reached.

6.179.3.4 virtual double **decaf::io::DataInput::readDouble**() [pure virtual]

Reads eight input bytes and returns a double value.

It does this by first constructing a long long value in exactly the manner of the `readlong` method, then converting this long value to a double in exactly the manner of the method `Double::longBitsToDouble`.

Returns

the double value read.

Exceptions

<i>IOException</i> (p. 1545)	if an I/O Error occurs.
<i>EOFException</i> (p. 1275)	if the end of input is reached.

6.179.3.5 virtual float **decaf::io::DataInput::readFloat** () [pure virtual]

Reads four input bytes and returns a float value.

It does this by first constructing an int value in exactly the manner of the readInt method, then converting this int value to a float in exactly the manner of the method Float::int-BitsToFloat.

Returns

the float value read.

Exceptions

<i>IOException</i> (p. 1545)	if an I/O Error occurs.
<i>EOFException</i> (p. 1275)	if the end of input is reached.

6.179.3.6 virtual void **decaf::io::DataInput::readFully** (unsigned char * *buffer*, int *size*) [pure virtual]

Reads some bytes from an input stream and stores them into the buffer array *buffer*.

The number of bytes read is equal to the length of *buffer*.

This method blocks until one of the following conditions occurs: * *buffer*'s size bytes of input data are available, in which case a normal return is made. * End of file is detected, in which case an **EOFException** (p. 1275) is thrown. * An I/O error occurs, in which case an **IOException** (p. 1545) other than **EOFException** (p. 1275) is thrown.

If *buffer* size is zero, then no bytes are read. Otherwise, the first byte read is stored into element *b*[0], the next one into *b*[1], and so on. If an exception is thrown from this method, then it may be that some but not all bytes of *buffer* have been updated with data from the input stream.

Parameters

<i>buffer</i>	The byte array to insert read data into.
<i>size</i>	The size in bytes of the given byte buffer.

Exceptions

<i>IOException</i> (p. 1545)	if an I/O Error occurs.
<i>EOFException</i> (p. 1275)	if the end of input is reached.
<i>IndexOutOfBoundsException</i>	if the size value is negative.

6.179.3.7 virtual void **decaf::io::DataInput::readFully** (unsigned char * *buffer*, int *size*, int *offset*, int *length*) [pure virtual]

Reads length bytes from an input stream.

This method blocks until one of the following conditions occurs: * length bytes of input data are available, in which case a normal return is made. * End of file is detected, in which case an **EOFException** (p. 1275) is thrown. * An I/O error occurs, in which case an **IOException** (p. 1545) other than **EOFException** (p. 1275) is thrown.

If buffer is NULL, a NullPointerException is thrown. If offset+length is greater than the length of the array buffer, then an IndexOutOfBoundsException is thrown. If length is zero, then no bytes are read. Otherwise, the first byte read is stored into element buffer[offset], the next one into buffer[offset+1], and so on. The number of bytes read is, at most, equal to length.

Parameters

<i>buffer</i>	The byte array to insert read data into.
<i>size</i>	The size in bytes of the given byte buffer.
<i>offset</i>	The location in buffer to start writing.
<i>length</i>	The number of bytes to read from the buffer.

Exceptions

<i>IOException</i> (p. 1545)	if an I/O Error occurs.
<i>EOFException</i> (p. 1275)	if the end of input is reached.
<i>NullPointerException</i>	if the buffer is NULL.
<i>IndexOutOfBoundsException</i>	if the offset + length > size, or an int param is negative.

6.179.3.8 virtual int **decaf::io::DataInput::readInt** () [pure virtual]

Reads four input bytes and returns an int value.

Let a be the first byte read, b be the second byte, c be the third byte, and d be the fourth byte. The value returned is:

$((a \& 0xff) \ll 24) \mid ((b \& 0xff) \ll 16) \mid ((c \& 0xff) \ll 8) \mid (d \& 0xff)$

Returns

the int value read.

Exceptions

<i>IOException</i> (p. 1545)	if an I/O Error occurs.
<i>EOFException</i> (p. 1275)	if the end of input is reached.

6.179.3.9 virtual std::string decaf::io::DataInput::readLine () [pure virtual]

Reads the next line of text from the input stream.

It reads successive bytes, converting each byte to an ASCII char separately, until it encounters a line terminator or end of file; the characters read are then returned as a standard String. Note that because this method processes bytes, it does not support input of the full Unicode character set.

If end of file is encountered before even one byte can be read, then an empty string is returned. Otherwise, each byte that is read is converted to type char. If the character '

' is encountered, it is discarded and reading ceases. If the character " is encountered, it is discarded and, if the following byte converts to the character '

', then that is discarded also; reading then ceases. If end of file is encountered before either of the characters '

' and " is encountered, reading ceases. Once reading has ceased, a String is returned that contains all the characters read and not discarded, taken in order.

Returns

the next line of text read from the input stream or empty string if at EOF.

Exceptions

<i>IOException</i> (p. 1545)	if an I/O Error occurs.
--	-------------------------

6.179.3.10 virtual long long decaf::io::DataInput::readLong () [pure virtual]

Reads eight input bytes and returns a long value.

Let a be the first byte read, b be the second byte, c be the third byte, d be the fourth byte, e be the fifth byte, f be the sixth byte, g be the seventh byte, and h be the eighth byte. The value returned is:

```
((long)(a & 0xff) << 56) | ((long)(b & 0xff) << 48) | ((long)(c & 0xff) << 40) | ((long)(d & 0xff) << 32) | ((long)(e & 0xff) << 24) | ((long)(f & 0xff) << 16) | ((long)(g & 0xff) << 8) | ((long)(h & 0xff))
```

Returns

the 64 bit long long read.

Exceptions

<i>IOException</i> (p. 1545)	if an I/O Error occurs.
<i>EOFException</i> (p. 1275)	if the end of input is reached.

6.179.3.11 virtual short decaf::io::DataInput::readShort () [pure virtual]

Reads two input bytes and returns a short value.

Let a be the first byte read and b be the second byte. The value returned is:

```
(short)((a << 8) | (b & 0xff))
```

Returns

the 16 bit short value read.

Exceptions

<i>IOException</i> (p. 1545)	if an I/O Error occurs.
<i>EOFException</i> (p. 1275)	if the end of input is reached.

6.179.3.12 virtual std::string decaf::io::DataInput::readString () [pure virtual]

Reads an NULL terminated ASCII string to the stream and returns the string to the caller.

Returns

string object containing the string read.

Exceptions

<i>IOException</i> (p. 1545)	if an I/O Error occurs.
<i>EOFException</i> (p. 1275)	if the end of input is reached.

6.179.3.13 `virtual unsigned char decaf::io::DataInput::readUnsignedByte ()` [pure virtual]

Reads one input byte, zero-extends it to type int, and returns the result, which is therefore in the range 0 through 255.

Returns

the 8 bit unsigned value read.

Exceptions

<i>IOException</i> (p. 1545)	if an I/O Error occurs.
<i>EOFException</i> (p. 1275)	if the end of input is reached.

6.179.3.14 `virtual unsigned short decaf::io::DataInput::readUnsignedShort ()` [pure virtual]

Reads two input bytes and returns an int value in the range 0 through 65535.

Let a be the first byte read and b be the second byte. The value returned is:

$((a \& 0xff) \ll 8) \mid (b \& 0xff)$

Returns

the 16 bit unsigned short read.

Exceptions

<i>IOException</i> (p. 1545)	if an I/O Error occurs.
<i>EOFException</i> (p. 1275)	if the end of input is reached.

6.179.3.15 virtual std::string decaf::io::DataInput::readUTF () [pure virtual]

Reads a modified UTF-8 encoded string in ASCII format and returns it, this is only useful if you know for sure that the string that is to be read was a string that contained all ASCII values (0-255), if so this method will throw a UTFFormatException.

This method reads String value written from a Java **DataOutputStream** (p. 1109) and assumes that the length prefix the precedes the encoded UTF-8 bytes is an unsigned short, which implies that the String will be no longer than 65535 characters.

Returns

The decoded string read from stream.

Exceptions

<i>IOException</i> (p. 1545)	if an I/O Error occurs.
<i>EOFException</i> (p. 1275)	if the end of input is reached.
<i>UTFDataFormatException</i> (p. 2874)	if the bytes are not valid modified UTF-8 values.

6.179.3.16 virtual long long decaf::io::DataInput::skipBytes (long long *num*) [pure virtual]

Makes an attempt to skip over n bytes of data from the input stream, discarding the skipped bytes.

However, it may skip over some smaller number of bytes, possibly zero. This may result from any of a number of conditions; reaching end of file before n bytes have been skipped is only one possibility. This method never throws an **EOFException** (p. 1275). The actual number of bytes skipped is returned.

Parameters

<i>num</i>	The number of bytes to skip over.
------------	-----------------------------------

Returns

the total number of bytes skipped.

Exceptions

<i>IOException</i> (p. 1545)	if an I/O Error occurs.
--	-------------------------

The documentation for this class was generated from the following file:

- `src/main/decaf/io/DataInput.h`

6.180 decaf::io::DataInputStream Class Reference

A data input stream lets an application read primitive Java data types from an underlying input stream in a machine-independent way.

```
#include <src/main/decaf/io/DataInputStream.h>
```

Inheritance diagram for decaf::io::DataInputStream:

Public Member Functions

- **DataInputStream** (**InputStream** ***inputStream**, bool **own**=false)
*Creates a **DataInputStream** (p. 1094) that uses the specified underlying **InputStream** (p. 1464).*
- virtual ~**DataInputStream** ()
- virtual bool **readBoolean** ()
Reads in one byte and returns true if that byte is nonzero, false if that byte is zero.
- virtual char **readByte** ()
Reads and returns one input byte.
- virtual unsigned char **readUnsignedByte** ()
Reads one input byte, zero-extends it to type int, and returns the result, which is therefore in the range 0 through 255.
- virtual char **readChar** ()
Reads an input char and returns the char value.
- virtual double **readDouble** ()
Reads eight input bytes and returns a double value.
- virtual float **readFloat** ()
Reads four input bytes and returns a float value.
- virtual int **readInt** ()
Reads four input bytes and returns an int value.
- virtual long long **readLong** ()
Reads eight input bytes and returns a long value.
- virtual short **readShort** ()
Reads two input bytes and returns a short value.
- virtual unsigned short **readUnsignedShort** ()
Reads two input bytes and returns an int value in the range 0 through 65535.
- virtual std::string **readString** ()
Reads an NULL terminated ASCII string to the stream and returns the string to the caller.

- virtual std::string **readLine** ()
Reads the next line of text from the input stream.
- virtual std::string **readUTF** ()
Reads a modified UTF-8 encoded string in ASCII format and returns it, this is only useful if you know for sure that the string that is to be read was a string that contained all ASCII values (0-255), if so this method will throw a UTFFormatException.
- virtual void **readFully** (unsigned char *buffer, int size)
Reads some bytes from an input stream and stores them into the buffer array buffer.
- virtual void **readFully** (unsigned char *buffer, int size, int offset, int length)
Reads length bytes from an input stream.
- virtual long long **skipBytes** (long long num)
Makes an attempt to skip over n bytes of data from the input stream, discarding the skipped bytes.

6.180.1 Detailed Description

A data input stream lets an application read primitive Java data types from an underlying input stream in a machine-independent way.

An application uses a data output stream to write data that can later be read by a data input stream.

Due to the lack of garbage collection in C++ a design decision was made to add a boolean parameter to the constructor indicating if the wrapped **InputStream** (p. 1464) is owned by this object. That way creation of the underlying stream can occur in a Java like way. Ex:

```
DataInputStream (p. 1094) os = new DataInputStream (p. 1094)( new InputStream()
(p. 1466), true )
```

Since

1.0

6.180.2 Constructor & Destructor Documentation

6.180.2.1 decaf::io::DataInputStream::DataInputStream (**InputStream** * *inputStream*, bool *own* = false)

Creates a **DataInputStream** (p. 1094) that uses the specified underlying **InputStream** (p. 1464).

Parameters

<i>inputStream</i>	the InputStream (p. 1464) instance to wrap.
<i>own</i>	indicates if this class owns the wrapped string defaults to false.

6.180.2.2 virtual `decaf::io::DataInputStream::~~DataInputStream ()` [virtual]

6.180.3 Member Function Documentation

6.180.3.1 virtual `bool decaf::io::DataInputStream::readBoolean ()` [virtual]

Reads in one byte and returns true if that byte is nonzero, false if that byte is zero.

Returns

the boolean value of the read in byte (0=false, 1=true).

Exceptions

<i>IOException</i> (p. 1545)	if an I/O Error occurs.
<i>EOFException</i> (p. 1275)	if the end of input is reached.

6.180.3.2 virtual `char decaf::io::DataInputStream::readByte ()` [virtual]

Reads and returns one input byte.

The byte is treated as a signed value in the range -128 through 127, inclusive.

Returns

the 8-bit value read.

Exceptions

<i>IOException</i> (p. 1545)	if an I/O Error occurs.
<i>EOFException</i> (p. 1275)	if the end of input is reached.

6.180.3.3 virtual `char decaf::io::DataInputStream::readChar ()` [virtual]

Reads an input char and returns the char value.

A ascii char is made up of one bytes. This returns the same result as `readByte`

Returns

the 8 bit char read.

Exceptions

<i>IOException</i> (p. 1545)	if an I/O Error occurs.
<i>EOFException</i> (p. 1275)	if the end of input is reached.

6.180.3.4 virtual double decaf::io::DataInputStream::readDouble () [virtual]

Reads eight input bytes and returns a double value.

It does this by first constructing a long long value in exactly the manner of the readLong method, then converting this long value to a double in exactly the manner of the method Double::longBitsToDouble.

Returns

the double value read.

Exceptions

<i>IOException</i> (p. 1545)	if an I/O Error occurs.
<i>EOFException</i> (p. 1275)	if the end of input is reached.

6.180.3.5 virtual float decaf::io::DataInputStream::readFloat () [virtual]

Reads four input bytes and returns a float value.

It does this by first constructing an int value in exactly the manner of the readInt method, then converting this int value to a float in exactly the manner of the method Float::intBitsToFloat.

Returns

the float value read.

Exceptions

<i>IOException</i> (p. 1545)	if an I/O Error occurs.
<i>EOFException</i> (p. 1275)	if the end of input is reached.

6.180.3.6 virtual void **decaf::io::DataInputStream::readFully** (unsigned char * *buffer*,
int *size*) [virtual]

Reads some bytes from an input stream and stores them into the buffer array *buffer*.

The number of bytes read is equal to the length of *buffer*.

This method blocks until one of the following conditions occurs: * *buffer*'s size bytes of input data are available, in which case a normal return is made. * End of file is detected, in which case an **EOFException** (p. 1275) is thrown. * An I/O error occurs, in which case an **IOException** (p. 1545) other than **EOFException** (p. 1275) is thrown.

If *buffer* size is zero, then no bytes are read. Otherwise, the first byte read is stored into element *b*[0], the next one into *buffer*[1], and so on. If an exception is thrown from this method, then it may be that some but not all bytes of *buffer* have been updated with data from the input stream.

Parameters

<i>buffer</i>	The byte array to insert read data into.
<i>size</i>	The size in bytes of the given byte <i>buffer</i> .

Exceptions

IOException (p. 1545)	if an I/O Error occurs.
EOFException (p. 1275)	if the end of input is reached.
IndexOutOfBoundsException	if the <i>size</i> value is negative.

6.180.3.7 virtual void **decaf::io::DataInputStream::readFully** (unsigned char * *buffer*,
int *size*, int *offset*, int *length*) [virtual]

Reads *length* bytes from an input stream.

This method blocks until one of the following conditions occurs: * *length* bytes of input data are available, in which case a normal return is made. * End of file is detected, in which case an **EOFException** (p. 1275) is thrown. * An I/O error occurs, in which case an **IOException** (p. 1545) other than **EOFException** (p. 1275) is thrown.

If *buffer* is NULL, a **NullPointerException** is thrown. If *offset*+*length* is greater than the length of the array *buffer*, then an **IndexOutOfBoundsException** is thrown. If *length* is zero, then no bytes are read. Otherwise, the first byte read is stored into element *buffer*[*off*], the next one into *buffer*[*offset*+1], and so on. The number of bytes read is, at most, equal to *length*.

Parameters

<i>buffer</i>	The byte array to insert read data into.
<i>size</i>	The size in bytes of the given byte <i>buffer</i> .

<i>offset</i>	The location in buffer to start writing.
<i>length</i>	The number of bytes to read from the buffer.

Exceptions

<i>IOException</i> (p. 1545)	if an I/O Error occurs.
<i>EOFException</i> (p. 1275)	if the end of input is reached.
<i>NullPointerException</i>	if the buffer is NULL.
<i>IndexOutOfBoundsException</i>	if the offset + length > size.

6.180.3.8 virtual int decaf::io::DataInputStream::readInt () [virtual]

Reads four input bytes and returns an int value.

Let a be the first byte read, b be the second byte, c be the third byte, and d be the fourth byte. The value returned is:

$$(((a \& 0xff) << 24) | ((b \& 0xff) << 16) | ((c \& 0xff) << 8) | (d \& 0xff))$$
Returns

the int value read.

Exceptions

<i>IOException</i> (p. 1545)	if an I/O Error occurs.
<i>EOFException</i> (p. 1275)	if the end of input is reached.

6.180.3.9 virtual std::string decaf::io::DataInputStream::readLine () [virtual]

Reads the next line of text from the input stream.

It reads successive bytes, converting each byte to an ASCII char separately, until it encounters a line terminator or end of file; the characters read are then returned as a standard String. Note that because this method processes bytes, it does not support input of the full Unicode character set.

If end of file is encountered before even one byte can be read, then an empty string is returned. Otherwise, each byte that is read is converted to type char. If the character ' ' is encountered, it is discarded and reading ceases. If the character " " is encountered, it is discarded and, if the following byte converts to the character ' '.

' , then that is discarded also; reading then ceases. If end of file is encountered before either of the characters ' '

' and " is encountered, reading ceases. Once reading has ceased, a String is returned that contains all the characters read and not discarded, taken in order.

Returns

the next line of text read from the input stream or empty string if at EOF.

Exceptions

<i>IOException</i> (p. 1545)	if an I/O Error occurs.
--	-------------------------

6.180.3.10 virtual long long decaf::io::DataInputStream::readLong () [virtual]

Reads eight input bytes and returns a long value.

Let a be the first byte read, b be the second byte, c be the third byte, d be the fourth byte, e be the fifth byte, f be the sixth byte, g be the seventh byte, and h be the eighth byte. The value returned is:

$((\text{long})(a \& 0\text{xff}) \ll 56) \mid ((\text{long})(b \& 0\text{xff}) \ll 48) \mid ((\text{long})(c \& 0\text{xff}) \ll 40) \mid ((\text{long})(d \& 0\text{xff}) \ll 32) \mid ((\text{long})(e \& 0\text{xff}) \ll 24) \mid ((\text{long})(f \& 0\text{xff}) \ll 16) \mid ((\text{long})(g \& 0\text{xff}) \ll 8) \mid ((\text{long})(h \& 0\text{xff}))$

Returns

the 64 bit long long read.

Exceptions

<i>IOException</i> (p. 1545)	if an I/O Error occurs.
<i>EOFException</i> (p. 1275)	if the end of input is reached.

6.180.3.11 virtual short decaf::io::DataInputStream::readShort () [virtual]

Reads two input bytes and returns a short value.

Let a be the first byte read and b be the second byte. The value returned is:

$(\text{short})((a \ll 8) \mid (b \& 0\text{xff}))$

Returns

the 16 bit short value read.

Exceptions

<i>IOException</i> (p. 1545)	if an I/O Error occurs.
<i>EOFException</i> (p. 1275)	if the end of input is reached.

6.180.3.12 virtual std::string decaf::io::DataInputStream::readString () [virtual]

Reads an NULL terminated ASCII string to the stream and returns the string to the caller.

Returns

string object containing the string read.

Exceptions

<i>IOException</i> (p. 1545)	if an I/O Error occurs.
<i>EOFException</i> (p. 1275)	if the end of input is reached.

6.180.3.13 virtual unsigned char decaf::io::DataInputStream::readUnsignedByte () [virtual]

Reads one input byte, zero-extends it to type int, and returns the result, which is therefore in the range 0 through 255.

Returns

the 8 bit unsigned value read.

Exceptions

<i>IOException</i> (p. 1545)	if an I/O Error occurs.
<i>EOFException</i> (p. 1275)	if the end of input is reached.

6.180.3.14 virtual unsigned short decaf::io::DataInputStream::readUnsignedShort () [virtual]

Reads two input bytes and returns an int value in the range 0 through 65535.

Let a be the first byte read and b be the second byte. The value returned is:

$((a \& 0xff) << 8) | (b \& 0xff)$

Returns

the 16 bit unsigned short read.

Exceptions

<i>IOException</i> (p. 1545)	if an I/O Error occurs.
<i>EOFException</i> (p. 1275)	if the end of input is reached.

6.180.3.15 virtual std::string decaf::io::DataInputStream::readUTF () [virtual]

Reads a modified UTF-8 encoded string in ASCII format and returns it, this is only useful if you know for sure that the string that is to be read was a string that contained all ASCII values (0-255), if so this method will throw a UTFFormatException.

This method reads String value written from a Java **DataOutputStream** (p. 1109) and assumes that the length prefix the precedes the encoded UTF-8 bytes is an unsigned short, which implies that the String will be no longer than 65535 characters.

Returns

The decoded string read from stream.

Exceptions

<i>IOException</i> (p. 1545)	if an I/O Error occurs.
<i>EOFException</i> (p. 1275)	if the end of input is reached.
<i>UTFDataFormatException</i> (p. 2874)	if the bytes are not valid modified UTF-8 values.

6.180.3.16 virtual long long decaf::io::DataInputStream::skipBytes (long long num) [virtual]

Makes an attempt to skip over n bytes of data from the input stream, discarding the skipped bytes.

However, it may skip over some smaller number of bytes, possibly zero. This may result from any of a number of conditions; reaching end of file before n bytes have been

skipped is only one possibility. This method never throws an **EOFException** (p. 1275). The actual number of bytes skipped is returned.

Parameters

<i>num</i>	The number of bytes to skip over.
------------	-----------------------------------

Returns

the total number of bytes skipped.

Exceptions

<i>IOException</i> (p. 1545)	if an I/O Error occurs.
--	-------------------------

The documentation for this class was generated from the following file:

- src/main/decaf/io/**DataInputStream.h**

6.181 decaf::io::DataOutput Class Reference

The **DataOutput** (p. 1103) interface provides for converting data from any of the C++ primitive types to a series of bytes and writing these bytes to a binary stream.

```
#include <src/main/decaf/io/DataOutput.h>
```

Public Member Functions

- virtual **~DataOutput** ()
- virtual void **writeBoolean** (bool value)=0
Writes a boolean to the underlying output stream as a 1-byte value.
- virtual void **writeByte** (unsigned char value)=0
Writes out a byte to the underlying output stream as a 1-byte value.
- virtual void **writeShort** (short value)=0
Writes a short to the underlying output stream as two bytes, high byte first.
- virtual void **writeUnsignedShort** (unsigned short value)=0
Writes a unsigned short to the bytes message stream as a 2 byte value.
- virtual void **writeChar** (char value)=0
Writes out a char to the underlying output stream as a one byte value If no exception is thrown, the counter written is incremented by 1.
- virtual void **writeInt** (int value)=0
Writes an int to the underlying output stream as four bytes, high byte first.
- virtual void **writeLong** (long long value)=0
Writes an 64 bit long to the underlying output stream as eight bytes, high byte first.

- virtual void **writeFloat** (float value)=0
Converts the float argument to an int using the floatToIntBits method in class Float, and then writes that int value to the underlying output stream as a 4-byte quantity, high byte first.
- virtual void **writeDouble** (double value)=0
Converts the double argument to a long using the doubleToLongBits method in class Double, and then writes that long value to the underlying output stream as an 8-byte quantity, high byte first.
- virtual void **writeBytes** (const std::string &value)=0
Writes out the string to the underlying output stream as a sequence of bytes.
- virtual void **writeChars** (const std::string &value)=0
Writes a string to the underlying output stream as a sequence of characters.
- virtual void **writeUTF** (const std::string &value)=0
Writes out the string to the underlying output stream as a modified UTF-8 encoded sequence of bytes.

6.181.1 Detailed Description

The **DataOutput** (p. 1103) interface provides for converting data from any of the C++ primitive types to a series of bytes and writing these bytes to a binary stream.

There is also a facility for converting Strings into the Java standard modified UTF-8 format and writing the resulting series of bytes.

If a method in this interface encounters an error while writing it will throw an **IOException** (p. 1545).

See also

DataInput (p. 1086)

DataOutputStream (p. 1109)

Since

1.0

6.181.2 Constructor & Destructor Documentation

6.181.2.1 virtual decaf::io::DataOutput::~DataOutput() [inline, virtual]

6.181.3 Member Function Documentation

6.181.3.1 virtual void decaf::io::DataOutput::writeBoolean(bool value) [pure virtual]

Writes a boolean to the underlying output stream as a 1-byte value.

The value true is written out as the value (byte)1; the value false is written out as the value (byte)0. If no exception is thrown, the counter written is incremented by 1.

Parameters

<i>value</i>	The boolean to write as a byte (1=true, 0=false).
--------------	---

Exceptions

<i>IOException</i> (p. 1545)	if an I/O error is encountered.
--	---------------------------------

6.181.3.2 `virtual void decaf::io::DataOutput::writeByte (unsigned char value)` [pure virtual]

Writes out a byte to the underlying output stream as a 1-byte value.

If no exception is thrown, the counter written is incremented by 1.

Parameters

<i>value</i>	The unsigned char value to write.
--------------	-----------------------------------

Exceptions

<i>IOException</i> (p. 1545)	if an I/O error is encountered.
--	---------------------------------

6.181.3.3 `virtual void decaf::io::DataOutput::writeBytes (const std::string & value)` [pure virtual]

Writes out the string to the underlying output stream as a sequence of bytes.

Each character in the string is written out, in sequence, by discarding its high eight bits. If no exception is thrown, the counter written is incremented by the length of value. The value written does not include a trailing null as that is not part of the sequence of bytes, if the null is needed, then use the writeChars method.

Parameters

<i>value</i>	The vector of bytes to write.
--------------	-------------------------------

Exceptions

<i>IOException</i> (p. 1545)	if an I/O error is encountered.
--	---------------------------------

6.181.3.4 `virtual void decaf::io::DataOutput::writeChar (char value)` [pure virtual]

Writes out a char to the underlying output stream as a one byte value. If no exception is thrown, the counter written is incremented by 1.

Parameters

<i>value</i>	The signed char value to write.
--------------	---------------------------------

Exceptions

<i>IOException</i> (p. 1545)	if an I/O error is encountered.
--	---------------------------------

6.181.3.5 `virtual void decaf::io::DataOutput::writeChars (const std::string & value)` [pure virtual]

Writes a string to the underlying output stream as a sequence of characters.

Each character is written to the data output stream as if by the writeChar method. If no exception is thrown, the counter written is incremented by the length of value. The trailing NULL character is written by this method.

Parameters

<i>value</i>	The string value to write as raw bytes.
--------------	---

Exceptions

<i>IOException</i> (p. 1545)	if an I/O error is encountered.
--	---------------------------------

6.181.3.6 `virtual void decaf::io::DataOutput::writeDouble (double value)` [pure virtual]

Converts the double argument to a long using the doubleToLongBits method in class Double, and then writes that long value to the underlying output stream as an 8-byte quantity, high byte first.

If no exception is thrown, the counter written is incremented by 8.

Parameters

<i>value</i>	The 64bit double value to write.
--------------	----------------------------------

Exceptions

<i>IOException</i> (p. 1545)	if an I/O error is encountered.
--	---------------------------------

6.181.3.7 `virtual void decaf::io::DataOutput::writeFloat (float value)` [pure virtual]

Converts the float argument to an int using the floatToIntBits method in class Float, and then writes that int value to the underlying output stream as a 4-byte quantity, high byte first.

If no exception is thrown, the counter written is incremented by 4.

Parameters

<i>value</i>	The 32bit floating point value to write.
--------------	--

Exceptions

<i>IOException</i> (p. 1545)	if an I/O error is encountered.
--	---------------------------------

6.181.3.8 `virtual void decaf::io::DataOutput::writeInt (int value)` [pure virtual]

Writes an int to the underlying output stream as four bytes, high byte first.

If no exception is thrown, the counter written is incremented by 4.

Parameters

<i>value</i>	The signed integer value to write.
--------------	------------------------------------

Exceptions

<i>IOException</i> (p. 1545)	if an I/O error is encountered.
--	---------------------------------

6.181.3.9 `virtual void decaf::io::DataOutput::writeLong (long long value)` [pure virtual]

Writes an 64 bit long to the underlying output stream as eight bytes, high byte first.

If no exception is thrown, the counter written is incremented by 8.

Parameters

<i>value</i>	The signed 64bit long value to write.
--------------	---------------------------------------

Exceptions

<i>IOException</i> (p. 1545)	if an I/O error is encountered.
--	---------------------------------

6.181.3.10 `virtual void decaf::io::DataOutput::writeShort (short value)` [pure virtual]

Writes a short to the underlying output stream as two bytes, high byte first.

If no exception is thrown, the counter written is incremented by 2.

Parameters

<i>value</i>	The signed short value to write.
--------------	----------------------------------

Exceptions

<i>IOException</i> (p. 1545)	if an I/O error is encountered.
--	---------------------------------

6.181.3.11 `virtual void decaf::io::DataOutput::writeUnsignedShort (unsigned short value)` [pure virtual]

Writes a unsigned short to the bytes message stream as a 2 byte value.

Parameters

<i>value</i>	The unsigned short to write to the stream.
--------------	--

Exceptions

<i>IOException</i> (p. 1545)	if an I/O error is encountered.
--	---------------------------------

6.181.3.12 `virtual void decaf::io::DataOutput::writeUTF (const std::string & value)` [pure virtual]

Writes out the string to the underlying output stream as a modeified UTF-8 encoded sequence of bytes.

The first two bytes written are indicate its encoded length followed by the rest of the

string's characters encoded as modified UTF-8. The length represent the encoded length of the data not the actual length of the string.

Parameters

<i>value</i>	The string value value to write as modified UTF-8.
--------------	--

Exceptions

<i>IOException</i> (p. 1545)	if an I/O error is encountered.
<i>UTFDataFormat-Exception</i> (p. 2874)	if the encoded size if greater than 65535

The documentation for this class was generated from the following file:

- src/main/decaf/io/**DataOutput.h**

6.182 decaf::io::DataOutputStream Class Reference

A data output stream lets an application write primitive Java data types to an output stream in a portable way.

```
#include <src/main/decaf/io/DataOutputStream.h>
```

Inheritance diagram for decaf::io::DataOutputStream:

Public Member Functions

- **DataOutputStream** (**OutputStream** ***outputStream**, bool **own**=false)
Creates a new data output stream to write data to the specified underlying output stream.
- virtual ~**DataOutputStream** ()
- virtual long long **size** () const
Returns the current value of the counter written, the number of bytes written to this data output stream so far.
- virtual void **writeBoolean** (bool value)
- virtual void **writeByte** (unsigned char value)
- virtual void **writeShort** (short value)
- virtual void **writeUnsignedShort** (unsigned short value)

- virtual void **writeChar** (char value)
- virtual void **writeInt** (int value)
- virtual void **writeLong** (long long value)
- virtual void **writeFloat** (float value)
- virtual void **writeDouble** (double value)
- virtual void **writeBytes** (const std::string &value)
- virtual void **writeChars** (const std::string &value)
- virtual void **writeUTF** (const std::string &value)

Protected Member Functions

- virtual void **doWriteByte** (unsigned char value)
- virtual void **doWriteArrayBounded** (const unsigned char ***buffer**, int **size**, int offset, int length)

Protected Attributes

- long long **written**
- unsigned char **buffer** [8]

6.182.1 Detailed Description

A data output stream lets an application write primitive Java data types to an output stream in a portable way.

An application can then use a data input stream to read the data back in.

6.182.2 Constructor & Destructor Documentation

6.182.2.1 **decaf::io::DataOutputStream::DataOutputStream (OutputStream * *outputStream*, bool *own* = false)**

Creates a new data output stream to write data to the specified underlying output stream.

Parameters

<i>output-Stream</i>	a stream to wrap with this one.
<i>own</i>	true if this objects owns the stream that it wraps.

6.182.2.2 virtual **decaf::io::DataOutputStream::~DataOutputStream** ()
[virtual]

6.182.3 Member Function Documentation

6.182.3.1 virtual void **decaf::io::DataOutputStream::doWriteArrayBounded** (const unsigned char * *buffer*, int *size*, int *offset*, int *length*) [protected, virtual]

Reimplemented from **decaf::io::FilterOutputStream** (p. 1343).

6.182.3.2 virtual void **decaf::io::DataOutputStream::doWriteByte** (unsigned char *value*) [protected, virtual]

Reimplemented from **decaf::io::FilterOutputStream** (p. 1343).

6.182.3.3 virtual long long **decaf::io::DataOutputStream::size** () const [inline, virtual]

Returns the current value of the counter written, the number of bytes written to this data output stream so far.

If the counter overflows, it will be wrapped to **decaf::lang::Long::MAX_VALUE** (p. 1741).

Returns

the value of the written field.

6.182.3.4 virtual void **decaf::io::DataOutputStream::writeBoolean** (bool *value*)
[virtual]

Referenced by **activemq::wireformat::openwire::marshal::BaseDataStreamMarshaller::looseMarshalObjectArray()**.

6.182.3.5 virtual void **decaf::io::DataOutputStream::writeByte** (unsigned char *value*)
[virtual]

- 6.182.3.6 `virtual void decaf::io::DataOutputStream::writeBytes (const std::string & value)` [virtual]
- 6.182.3.7 `virtual void decaf::io::DataOutputStream::writeChar (char value)` [virtual]
- 6.182.3.8 `virtual void decaf::io::DataOutputStream::writeChars (const std::string & value)` [virtual]
- 6.182.3.9 `virtual void decaf::io::DataOutputStream::writeDouble (double value)` [virtual]
- 6.182.3.10 `virtual void decaf::io::DataOutputStream::writeFloat (float value)` [virtual]
- 6.182.3.11 `virtual void decaf::io::DataOutputStream::writeInt (int value)` [virtual]
- 6.182.3.12 `virtual void decaf::io::DataOutputStream::writeLong (long long value)` [virtual]
- 6.182.3.13 `virtual void decaf::io::DataOutputStream::writeShort (short value)` [virtual]

Referenced by `activemq::wireformat::openwire::marshal::BaseDataStreamMarshaller::looseMarshalObjectArray()`, and `activemq::wireformat::openwire::marshal::BaseDataStreamMarshaller::tightMarshalObjectArray2()`.

- 6.182.3.14 `virtual void decaf::io::DataOutputStream::writeUnsignedShort (unsigned short value)` [virtual]
- 6.182.3.15 `virtual void decaf::io::DataOutputStream::writeUTF (const std::string & value)` [virtual]

6.182.4 Field Documentation

- 6.182.4.1 `unsigned char decaf::io::DataOutputStream::buffer[8]` [protected]
- 6.182.4.2 `long long decaf::io::DataOutputStream::written` [protected]

The documentation for this class was generated from the following file:

- `src/main/decaf/io/DataOutputStream.h`

6.183 activemq::commands::DataResponse Class Reference

```
#include <src/main/activemq/commands/DataResponse.h>
```

Inheritance diagram for activemq::commands::DataResponse:

Public Member Functions

- **DataResponse** ()
- virtual **~DataResponse** ()
- virtual unsigned char **getDataStructureType** () const
*Get the **DataStructure** (p. 1133) Type as defined in CommandTypes.h.*
- virtual **DataResponse** * **cloneDataStructure** () const
Clone this object and return a new instance that the caller now owns, this will be an exact copy of this one.
- virtual void **copyDataStructure** (const **DataStructure** *src)
Copy the contents of the passed object into this objects members, overwriting any existing data.
- virtual std::string **toString** () const
*Returns a string containing the information for this **DataStructure** (p. 1133) such as its type and value of its elements.*
- virtual bool **equals** (const **DataStructure** *value) const
*Compares the **DataStructure** (p. 1133) passed in to this one, and returns if they are equivalent.*
- virtual const **Pointer** < **DataStructure** > & **getData** () const
- virtual **Pointer**< **DataStructure** > & **getData** ()
- virtual void **setData** (const **Pointer**< **DataStructure** > &data)

Static Public Attributes

- static const unsigned char **ID_DATARESPONSE** = 32

Protected Attributes

- **Pointer**< **DataStructure** > **data**

6.183.1 Constructor & Destructor Documentation

6.183.1.1 **activemq::commands::DataResponse::DataResponse** ()

6.183.1.2 **virtual activemq::commands::DataResponse::~~DataResponse** ()
[virtual]

6.183.2 Member Function Documentation

6.183.2.1 `virtual DataResponse* activemq::commands::DataResponse::clone-
DataStructure () const` [virtual]

Clone this object and return a new instance that the caller now owns, this will be an exact copy of this one.

Returns

new copy of this object.

Reimplemented from `activemq::commands::Response` (p. 2299).

6.183.2.2 `virtual void activemq::commands::DataResponse::copyDataStructure (`
`const DataStructure * src)` [virtual]

Copy the contents of the passed object into this objects members, overwriting any existing data.

Returns

src - Source Object

Reimplemented from `activemq::commands::Response` (p. 2299).

6.183.2.3 `virtual bool activemq::commands::DataResponse::equals (const`
`DataStructure * value) const` [virtual]

Compares the `DataStructure` (p. 1133) passed in to this one, and returns if they are equivalent.

Equivalent here means that they are of the same type, and that each element of the objects are the same.

Returns

true if `DataStructure` (p. 1133)'s are Equal.

Reimplemented from `activemq::commands::Response` (p. 2299).

6.183.2.4 `virtual const Pointer<DataStructure>& activemq-
::commands::DataResponse::getData () const`
[virtual]

6.183.2.5 `virtual Pointer<DataStructure>& activemq::commands::DataResponse-
::getData ()` [virtual]

6.184

activemq::wireformat::openwire::marshal::generated::DataResponseMarshaller

Class Reference

1117

6.183.2.6 virtual unsigned char **activemq::commands::DataResponse::getData-
StructureType**() const [virtual]

Get the **DataStructure** (p. 1133) Type as defined in CommandTypes.h.

Returns

The type of the data structure

Reimplemented from **activemq::commands::Response** (p. 2300).

6.183.2.7 virtual void **activemq::commands::DataResponse::setData**(const
Pointer< DataStructure > & data) [virtual]

6.183.2.8 virtual std::string **activemq::commands::DataResponse::toString**() const
[virtual]

Returns a string containing the information for this **DataStructure** (p. 1133) such as its type and value of its elements.

Returns

formatted string useful for debugging.

Reimplemented from **activemq::commands::Response** (p. 2300).

6.183.3 Field Documentation

6.183.3.1 **Pointer<DataStructure> activemq::commands::DataResponse::data**
[protected]

6.183.3.2 const unsigned char **activemq::commands::DataResponse::ID_DATARES-
PONSE = 32** [static]

The documentation for this class was generated from the following file:

- src/main/activemq/commands/**DataResponse.h**

6.184 activemq::wireformat::openwire::marshal::generated::Data- ResponseMarshaller Class Reference

Marshaling code for Open Wire Format for **DataResponseMarshaller** (p. 1115).

```
#include <src/main/activemq/wireformat/openwire/marshal/generated/-  
DataResponseMarshaller.h>
```

Inheritance diagram for `activemq::wireformat::openwire::marshal::generated::DataResponseMarshaller`:

Public Member Functions

- **DataResponseMarshaller** ()
- virtual `~DataResponseMarshaller` ()
- virtual **commands::DataStructure** * **createObject** () const
Creates a new instance of the class that this class is a marshaling director for.
- virtual unsigned char **getDataStructureType** () const
Gets the DataStructureType that this class marshals/unmarshals.
- virtual void **tightUnmarshal** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataInputStream** *dataIn, **utils::BooleanStream** *bs)
Tight Un-marhsal to the given stream.
- virtual int **tightMarshal1** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **utils::BooleanStream** *bs)
Tight Marhsal to the given stream.
- virtual void **tightMarshal2** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataOutputStream** *dataOut, **utils::BooleanStream** *bs)
Tight Marhsal to the given stream.
- virtual void **looseUnmarshal** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataInputStream** *dataIn)
Loose Un-marhsal to the given stream.
- virtual void **looseMarshal** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataOutputStream** *dataOut)
Tight Marhsal to the given stream.

6.184.1 Detailed Description

Marshaling code for Open Wire Format for **DataResponseMarshaller** (p. 1115).

NOTE!: This file is auto generated - do not modify! if you need to make a change, please see the Java Classes in the `activemq-openwire-generator` module

6.184.2 Constructor & Destructor Documentation

- 6.184.2.1 **activemq::wireformat::openwire::marshal::generated::DataResponseMarshaller::DataResponseMarshaller** ()
`[inline]`

6.184

activemq::wireformat::openwire::marshal::generated::DataResponseMarshaller

Class Reference

1119

6.184.2.2 `virtual activemq::wireformat::openwire::marshal::generated::DataResponseMarshaller::~DataResponseMarshaller () [inline, virtual]`

6.184.3 Member Function Documentation

6.184.3.1 `virtual commands::DataStructure* activemq::wireformat::openwire::marshal::generated::DataResponseMarshaller::createObject () const [virtual]`

Creates a new instance of the class that this class is a marshaling director for.

Returns

newly allocated Command

Reimplemented from **activemq::wireformat::openwire::marshal::generated::DataResponseMarshaller** (p. 2308).

6.184.3.2 `virtual unsigned char activemq::wireformat::openwire::marshal::generated::DataResponseMarshaller::getDataStructureType () const [virtual]`

Gets the DataStructureType that this class marshals/unmarshals.

Returns

byte Id of this classes DataStructureType

Reimplemented from **activemq::wireformat::openwire::marshal::generated::DataResponseMarshaller** (p. 2309).

6.184.3.3 `virtual void activemq::wireformat::openwire::marshal::generated::DataResponseMarshaller::looseMarshal (OpenWireFormat * format, commands::DataStructure * command, decaf::io::DataOutputStream * ds) [virtual]`

Tight Marshal to the given stream.

Parameters

<i>format</i>	- The OpenWireFormat properties
<i>command</i>	- the object to Marshal
<i>ds</i>	- DataOutputStream to marshal to

Exceptions

<i>IOException</i>	if an error occurs.
--------------------	---------------------

Reimplemented from **activemq::wireformat::openwire::marshal::generated::ResponseMarshaller** (p. 2309).

6.184.3.4 `virtual void activemq::wireformat::openwire::marshal::generated::DataResponseMarshaller::looseUnmarshal (OpenWireFormat * format, commands::DataStructure * command, decaf::io::DataInputStream * dis)`
[virtual]

Loose Un-marhsal to the given stream.

Parameters

<i>format</i>	- The OpenWireFormat properties
<i>command</i>	- the object to Un-Marshal
<i>dis</i>	- the DataInputStream to Un-Marshal from

Exceptions

<i>IOException</i>	if an error occurs.
--------------------	---------------------

Reimplemented from **activemq::wireformat::openwire::marshal::generated::ResponseMarshaller** (p. 2310).

6.184.3.5 `virtual int activemq::wireformat::openwire::marshal::generated::DataResponseMarshaller::tightMarshal1 (OpenWireFormat * format, commands::DataStructure * command, utils::BooleanStream * bs)`
[virtual]

Tight Marhsal to the given stream.

Parameters

<i>format</i>	The OpenWireFormat properties
<i>command</i>	The object to Marshal
<i>bs</i>	The boolean stream to marshal to.

Exceptions

<i>IOException</i>	if an error occurs.
--------------------	---------------------

Reimplemented from **activemq::wireformat::openwire::marshal::generated::ResponseMarshaller** (p. 2310).

6.184

activemq::wireformat::openwire::marshal::generated::DataResponseMarshaller
Class Reference 1121

6.184.3.6 virtual void **activemq::wireformat::openwire::marshal::generated::DataResponseMarshaller::tightMarshal2** (**OpenWireFormat** * *format*,
commands::DataStructure * *command*, **decaf::io::DataOutputStream** *
ds, **utils::BooleanStream** * *bs*) [virtual]

Tight Marhsal to the given stream.

Parameters

<i>format</i>	- The OpenWireFormat properties
<i>command</i>	- the object to Marshal
<i>ds</i>	- the DataOutputStream to Marshal to
<i>bs</i>	- boolean stream to marshal to.

Exceptions

<i>IOException</i>	if an error occurs.
--------------------	---------------------

Reimplemented from **activemq::wireformat::openwire::marshal::generated::ResponseMarshaller** (p. 2311).

6.184.3.7 virtual void **activemq::wireformat::openwire::marshal::generated::DataResponseMarshaller::tightUnmarshal** (**OpenWireFormat** * *format*,
commands::DataStructure * *command*, **decaf::io::DataInputStream** * *dis*,
utils::BooleanStream * *bs*) [virtual]

Tight Un-marhsal to the given stream.

Parameters

<i>format</i>	- The OpenWireFormat properties
<i>command</i>	- the object to Un-Marshal
<i>dis</i>	- the DataInputStream to Un-Marshal from
<i>bs</i>	- boolean stream to unmarshal from.

Exceptions

<i>IOException</i>	if an error occurs.
--------------------	---------------------

Reimplemented from **activemq::wireformat::openwire::marshal::generated::ResponseMarshaller** (p. 2311).

The documentation for this class was generated from the following file:

- src/main/activemq/wireformat/openwire/marshal/generated/**DataResponseMarshaller.h**

6.185 activemq::wireformat::openwire::marshal::DataStream-Marshaller Class Reference

Base class for all classes that marshal commands for Openwire.

```
#include <src/main/activemq/wireformat/openwire/marshal/-
DataStreamMarshaller.h>
```

Inheritance diagram for activemq::wireformat::openwire::marshal::DataStream-Marshaller:

Public Member Functions

- virtual **~DataStreamMarshaller** ()
- virtual unsigned char **getDataStructureType** () const =0
Gets the DataStructureType that this class marshals/unmarshals.
- virtual **commands::DataStructure * createObject** () const =0
Creates a new instance of the class that this class is a marshaling director for.
- virtual int **tightMarshal1** (**OpenWireFormat** *format, **commands::Data-Structure** *command, **utils::BooleanStream** *bs)=0
Tight Marshal to the given stream.
- virtual void **tightMarshal2** (**OpenWireFormat** *format, **commands::Data-Structure** *command, **decaf::io::DataOutputStream** *ds, **utils::BooleanStream** *bs)=0
Tight Marshal to the given stream.
- virtual void **tightUnmarshal** (**OpenWireFormat** *format, **commands::Data-Structure** *command, **decaf::io::DataInputStream** *dis, **utils::BooleanStream** *bs)=0
Tight Un-marshal to the given stream.
- virtual void **looseMarshal** (**OpenWireFormat** *format, **commands::Data-Structure** *command, **decaf::io::DataOutputStream** *ds)=0
Tight Marshal to the given stream.
- virtual void **looseUnmarshal** (**OpenWireFormat** *format, **commands::Data-Structure** *command, **decaf::io::DataInputStream** *dis)=0
Loose Un-marshal to the given stream.

6.185.1 Detailed Description

Base class for all classes that marshal commands for Openwire.

6.185.2 Constructor & Destructor Documentation

6.185.2.1 virtual activemq::wireformat::openwire::marshal::DataStreamMarshaller::~DataStreamMarshaller () [inline, virtual]

6.185.3 Member Function Documentation

6.185.3.1 virtual commands::DataStructure* activemq::wireformat::openwire::marshal::DataStreamMarshaller::createObject () const [pure virtual]

Creates a new instance of the class that this class is a marshaling director for.

Returns

newly allocated Command

Implemented in **activemq::wireformat::openwire::marshal::generated::ActiveMQBlobMessageMarshaller** (p. 163), **activemq::wireformat::openwire::marshal::generated::ActiveMQBytesMessageMarshaller** (p. 184), **activemq::wireformat::openwire::marshal::generated::ActiveMQMapMessageMarshaller** (p. 285), **activemq::wireformat::openwire::marshal::generated::ActiveMQMessageMarshaller** (p. 292), **activemq::wireformat::openwire::marshal::generated::ActiveMQObjectMessageMarshaller** (p. 305), **activemq::wireformat::openwire::marshal::generated::ActiveMQQueueMarshaller** (p. 330), **activemq::wireformat::openwire::marshal::generated::ActiveMQStreamMessageMarshaller** (p. 376), **activemq::wireformat::openwire::marshal::generated::ActiveMQTempQueueMarshaller** (p. 393), **activemq::wireformat::openwire::marshal::generated::ActiveMQTempTopicMarshaller** (p. 402), **activemq::wireformat::openwire::marshal::generated::ActiveMQTextMessageMarshaller** (p. 411), **activemq::wireformat::openwire::marshal::generated::ActiveMQTopicMarshaller** (p. 420), **activemq::wireformat::openwire::marshal::generated::BrokerIdMarshaller** (p. 569), **activemq::wireformat::openwire::marshal::generated::BrokerInfoMarshaller** (p. 579), **activemq::wireformat::openwire::marshal::generated::ConnectionControlMarshaller** (p. 945), **activemq::wireformat::openwire::marshal::generated::ConnectionErrorMarshaller** (p. 952), **activemq::wireformat::openwire::marshal::generated::ConnectionIdMarshaller** (p. 965), **activemq::wireformat::openwire::marshal::generated::ConnectionInfoMarshaller** (p. 975), **activemq::wireformat::openwire::marshal::generated::ConsumerControlMarshaller** (p. 997), **activemq::wireformat::openwire::marshal::generated::ConsumerIdMarshaller** (p. 1006), **activemq::wireformat::openwire::marshal::generated::ConsumerInfoMarshaller** (p. 1019), **activemq::wireformat::openwire::marshal::generated::ControlCommandMarshaller** (p. 1027), **activemq::wireformat::openwire::marshal::generated::DataArrayResponseMarshaller** (p. 1073), **activemq::wireformat::openwire::marshal::generated::DataResponseMarshaller** (p. 1116), **activemq::wireformat::openwire::marshal::generated::DestinationInfoMarshaller** (p. 1219), **activemq::wireformat::openwire::marshal::generated::DiscoveryEventMarshaller** (p. 1231), **activemq::wireformat::openwire::marshal::generated::ExceptionResponseMarshaller**

(p. 1292), `activemq::wireformat::openwire::marshal::generated::FlushCommand-Marshaller` (p. 1385), `activemq::wireformat::openwire::marshal::generated::IntegerResponseMarshaller` (p. 1520), `activemq::wireformat::openwire::marshal::generated::JournalQueueAckMarshaller` (p. 1565), `activemq::wireformat::openwire::marshal::generated::JournalTopicAckMarshaller` (p. 1574), `activemq::wireformat::openwire::marshal::generated::JournalTraceMarshaller` (p. 1581), `activemq::wireformat::openwire::marshal::generated::JournalTransaction-Marshaller` (p. 1588), `activemq::wireformat::openwire::marshal::generated::KeepAliveInfoMarshaller` (p. 1595), `activemq::wireformat::openwire::marshal::generated::LastPartialCommandMarshaller` (p. 1609), `activemq::wireformat::openwire::marshal::generated::LocalTransactionIdMarshaller` (p. 1679), `activemq::wireformat::openwire::marshal::generated::MessageAckMarshaller` (p. 1874), `activemq::wireformat::openwire::marshal::generated::MessageDispatchMarshaller` (p. 1892), `activemq::wireformat::openwire::marshal::generated::MessageDispatchNotificationMarshaller` (p. 1901), `activemq::wireformat::openwire::marshal::generated::MessageIdMarshaller` (p. 1913), `activemq::wireformat::openwire::marshal::generated::MessagePullMarshaller` (p. 1945), `activemq::wireformat::openwire::marshal::generated::NetworkBridge-FilterMarshaller` (p. 1975), `activemq::wireformat::openwire::marshal::generated::PartialCommandMarshaller` (p. 2080), `activemq::wireformat::openwire::marshal::generated::ProducerAckMarshaller` (p. 2176), `activemq::wireformat::openwire::marshal::generated::ProducerIdMarshaller` (p. 2187), `activemq::wireformat::openwire::marshal::generated::ProducerInfoMarshaller` (p. 2196), `activemq::wireformat::openwire::marshal::generated::RemoveInfoMarshaller` (p. 2272), `activemq::wireformat::openwire::marshal::generated::RemoveSubscription-InfoMarshaller` (p. 2281), `activemq::wireformat::openwire::marshal::generated::ReplayCommandMarshaller` (p. 2288), `activemq::wireformat::openwire::marshal::generated::ResponseMarshaller` (p. 2308), `activemq::wireformat::openwire::marshal::generated::SessionIdMarshaller` (p. 2383), `activemq::wireformat::openwire::marshal::generated::SessionInfoMarshaller` (p. 2391), `activemq::wireformat::openwire::marshal::generated::ShutdownInfoMarshaller` (p. 2435), `activemq::wireformat::openwire::marshal::generated::Subscription-InfoMarshaller` (p. 2636), `activemq::wireformat::openwire::marshal::generated::TransactionInfoMarshaller` (p. 2779), `activemq::wireformat::openwire::marshal::generated::WireFormatInfoMarshaller` (p. 2901), and `activemq::wireformat::openwire::marshal::generated::XATransactionIdMarshaller` (p. 2944).

```
6.185.3.2 virtual unsigned char activemq::wireformat::openwire::marshal::-
        DataStreamMarshaller::getDataStructureType ( ) const [pure
        virtual]
```

Gets the `DataStreamType` that this class marshals/unmarshals.

Returns

byte Id of this classes `DataStreamType`

Implemented in `activemq::wireformat::openwire::marshal::generated::ActiveM-QBlobMessageMarshaller` (p. 163), `activemq::wireformat::openwire::marshal-`

generated::ActiveMQBytesMessageMarshaller (p. 185), **activemq::wireformat::openwire::marshal::generated::ActiveMQMapMessageMarshaller** (p. 285), **activemq::wireformat::openwire::marshal::generated::ActiveMQMessageMarshaller** (p. 292), **activemq::wireformat::openwire::marshal::generated::ActiveMQObjectMessageMarshaller** (p. 305), **activemq::wireformat::openwire::marshal::generated::ActiveMQQueueMarshaller** (p. 330), **activemq::wireformat::openwire::marshal::generated::ActiveMQStreamMessageMarshaller** (p. 376), **activemq::wireformat::openwire::marshal::generated::ActiveMQTempQueueMarshaller** (p. 393), **activemq::wireformat::openwire::marshal::generated::ActiveMQTempTopicMarshaller** (p. 402), **activemq::wireformat::openwire::marshal::generated::ActiveMQTextMessageMarshaller** (p. 412), **activemq::wireformat::openwire::marshal::generated::ActiveMQTopicMarshaller** (p. 420), **activemq::wireformat::openwire::marshal::generated::BrokerIdMarshaller** (p. 569), **activemq::wireformat::openwire::marshal::generated::BrokerInfoMarshaller** (p. 580), **activemq::wireformat::openwire::marshal::generated::ConnectionControlMarshaller** (p. 945), **activemq::wireformat::openwire::marshal::generated::ConnectionErrorMarshaller** (p. 953), **activemq::wireformat::openwire::marshal::generated::ConnectionIdMarshaller** (p. 965), **activemq::wireformat::openwire::marshal::generated::ConnectionInfoMarshaller** (p. 975), **activemq::wireformat::openwire::marshal::generated::ConsumerControlMarshaller** (p. 998), **activemq::wireformat::openwire::marshal::generated::ConsumerIdMarshaller** (p. 1006), **activemq::wireformat::openwire::marshal::generated::ConsumerInfoMarshaller** (p. 1019), **activemq::wireformat::openwire::marshal::generated::ControlCommandMarshaller** (p. 1027), **activemq::wireformat::openwire::marshal::generated::DataArrayResponseMarshaller** (p. 1073), **activemq::wireformat::openwire::marshal::generated::DataResponseMarshaller** (p. 1117), **activemq::wireformat::openwire::marshal::generated::DestinationInfoMarshaller** (p. 1219), **activemq::wireformat::openwire::marshal::generated::DiscoveryEventMarshaller** (p. 1231), **activemq::wireformat::openwire::marshal::generated::ExceptionResponseMarshaller** (p. 1292), **activemq::wireformat::openwire::marshal::generated::FlushCommandMarshaller** (p. 1385), **activemq::wireformat::openwire::marshal::generated::IntegerResponseMarshaller** (p. 1520), **activemq::wireformat::openwire::marshal::generated::JournalQueueAckMarshaller** (p. 1565), **activemq::wireformat::openwire::marshal::generated::JournalTopicAckMarshaller** (p. 1574), **activemq::wireformat::openwire::marshal::generated::JournalTraceMarshaller** (p. 1581), **activemq::wireformat::openwire::marshal::generated::JournalTransactionMarshaller** (p. 1588), **activemq::wireformat::openwire::marshal::generated::KeepAliveInfoMarshaller** (p. 1596), **activemq::wireformat::openwire::marshal::generated::LastPartialCommandMarshaller** (p. 1610), **activemq::wireformat::openwire::marshal::generated::LocalTransactionIdMarshaller** (p. 1680), **activemq::wireformat::openwire::marshal::generated::MessageAckMarshaller** (p. 1874), **activemq::wireformat::openwire::marshal::generated::MessageDispatchMarshaller** (p. 1892), **activemq::wireformat::openwire::marshal::generated::MessageDispatchNotificationMarshaller** (p. 1901), **activemq::wireformat::openwire::marshal::generated::MessageIdMarshaller** (p. 1914), **activemq::wireformat::openwire::marshal::generated::MessagePullMarshaller** (p. 1945), **activemq::wireformat::openwire::marshal::generated::NetworkBridgeFilterMarshaller** (p. 1976), **activemq::wireformat::openwire::marshal::generated::PartialCommandMarshaller** (p. 2081), **activemq::wireformat::openwire::marshal::generated::ProducerAckMarshaller** (p. 2176), **activemq::wireformat::openwire::**

`::marshal::generated::ProducerIdMarshaller` (p. 2187), `activemq::wireformat::openwire::marshal::generated::ProducerInfoMarshaller` (p. 2196), `activemq::wireformat::openwire::marshal::generated::RemoveInfoMarshaller` (p. 2273), `activemq::wireformat::openwire::marshal::generated::RemoveSubscriptionInfoMarshaller` (p. 2281), `activemq::wireformat::openwire::marshal::generated::ReplayCommandMarshaller` (p. 2288), `activemq::wireformat::openwire::marshal::generated::ResponseMarshaller` (p. 2309), `activemq::wireformat::openwire::marshal::generated::SessionIdMarshaller` (p. 2384), `activemq::wireformat::openwire::marshal::generated::SessionInfoMarshaller` (p. 2391), `activemq::wireformat::openwire::marshal::generated::ShutdownInfoMarshaller` (p. 2436), `activemq::wireformat::openwire::marshal::generated::SubscriptionInfoMarshaller` (p. 2636), `activemq::wireformat::openwire::marshal::generated::TransactionInfoMarshaller` (p. 2779), `activemq::wireformat::openwire::marshal::generated::WireFormatInfoMarshaller` (p. 2901), and `activemq::wireformat::openwire::marshal::generated::XATransactionIdMarshaller` (p. 2944).

6.185.3.3 `virtual void activemq::wireformat::openwire::marshal::DataStreamMarshaller::looseMarshal (OpenWireFormat * format, commands::DataStructure * command, decaf::io::DataOutputStream * ds) [pure virtual]`

Tight Marshal to the given stream.

Parameters

<i>format</i>	- The OpenWireFormat properties
<i>command</i>	- the object to Marshal
<i>ds</i>	- DataOutputStream to marshal to

Exceptions

<i>IOException</i>	if an error occurs.
--------------------	---------------------

Implemented in `activemq::wireformat::openwire::marshal::generated::ActiveMQBlobMessageMarshaller` (p. 163), `activemq::wireformat::openwire::marshal::generated::ActiveMQBytesMessageMarshaller` (p. 185), `activemq::wireformat::openwire::marshal::generated::ActiveMQMapMessageMarshaller` (p. 286), `activemq::wireformat::openwire::marshal::generated::ActiveMQMessageMarshaller` (p. 292), `activemq::wireformat::openwire::marshal::generated::ActiveMQObjectMessageMarshaller` (p. 305), `activemq::wireformat::openwire::marshal::generated::ActiveMQQueueMarshaller` (p. 331), `activemq::wireformat::openwire::marshal::generated::ActiveMQStreamMessageMarshaller` (p. 376), `activemq::wireformat::openwire::marshal::generated::ActiveMQTempQueueMarshaller` (p. 393), `activemq::wireformat::openwire::marshal::generated::ActiveMQTempTopicMarshaller` (p. 403), `activemq::wireformat::openwire::marshal::generated::ActiveMQTextMessageMarshaller` (p. 412), `activemq::wireformat::openwire::marshal::generated::ActiveMQTopicMarshaller` (p. 421), `activemq::wireformat::openwire::marshal::generated::BrokerIdMarshaller` (p. 569), `activemq::wireformat::openwire::marshal::generated::BrokerInfoMarshaller` (p. 580), `activemq-`

::wireformat::openwire::marshal::generated::ConnectionControlMarshaller (p. 945), activemq::wireformat::openwire::marshal::generated::ConnectionErrorMarshaller (p. 953), activemq::wireformat::openwire::marshal::generated::ConnectionIdMarshaller (p. 965), activemq::wireformat::openwire::marshal::generated::ConnectionInfoMarshaller (p. 976), activemq::wireformat::openwire::marshal::generated::ConsumerControlMarshaller (p. 998), activemq::wireformat::openwire::marshal::generated::ConsumerIdMarshaller (p. 1006), activemq::wireformat::openwire::marshal::generated::ConsumerInfoMarshaller (p. 1019), activemq::wireformat::openwire::marshal::generated::ControlCommandMarshaller (p. 1027), activemq::wireformat::openwire::marshal::generated::DataArrayResponseMarshaller (p. 1073), activemq::wireformat::openwire::marshal::generated::DataResponseMarshaller (p. 1117), activemq::wireformat::openwire::marshal::generated::DestinationInfoMarshaller (p. 1220), activemq::wireformat::openwire::marshal::generated::DiscoveryEventMarshaller (p. 1231), activemq::wireformat::openwire::marshal::generated::ExceptionResponseMarshaller (p. 1292), activemq::wireformat::openwire::marshal::generated::FlushCommandMarshaller (p. 1385), activemq::wireformat::openwire::marshal::generated::IntegerResponseMarshaller (p. 1521), activemq::wireformat::openwire::marshal::generated::JournalQueueAckMarshaller (p. 1566), activemq::wireformat::openwire::marshal::generated::JournalTopicAckMarshaller (p. 1574), activemq::wireformat::openwire::marshal::generated::JournalTraceMarshaller (p. 1581), activemq::wireformat::openwire::marshal::generated::JournalTransactionMarshaller (p. 1589), activemq::wireformat::openwire::marshal::generated::KeepAliveInfoMarshaller (p. 1596), activemq::wireformat::openwire::marshal::generated::LastPartialCommandMarshaller (p. 1610), activemq::wireformat::openwire::marshal::generated::LocalTransactionIdMarshaller (p. 1680), activemq::wireformat::openwire::marshal::generated::MessageAckMarshaller (p. 1874), activemq::wireformat::openwire::marshal::generated::MessageDispatchMarshaller (p. 1892), activemq::wireformat::openwire::marshal::generated::MessageDispatchNotificationMarshaller (p. 1901), activemq::wireformat::openwire::marshal::generated::MessageIdMarshaller (p. 1914), activemq::wireformat::openwire::marshal::generated::MessagePullMarshaller (p. 1946), activemq::wireformat::openwire::marshal::generated::NetworkBridgeFilterMarshaller (p. 1976), activemq::wireformat::openwire::marshal::generated::PartialCommandMarshaller (p. 2081), activemq::wireformat::openwire::marshal::generated::ProducerAckMarshaller (p. 2177), activemq::wireformat::openwire::marshal::generated::ProducerIdMarshaller (p. 2188), activemq::wireformat::openwire::marshal::generated::ProducerInfoMarshaller (p. 2197), activemq::wireformat::openwire::marshal::generated::RemoveInfoMarshaller (p. 2273), activemq::wireformat::openwire::marshal::generated::RemoveSubscriptionInfoMarshaller (p. 2281), activemq::wireformat::openwire::marshal::generated::ReplayCommandMarshaller (p. 2289), activemq::wireformat::openwire::marshal::generated::ResponseMarshaller (p. 2309), activemq::wireformat::openwire::marshal::generated::SessionIdMarshaller (p. 2384), activemq::wireformat::openwire::marshal::generated::SessionInfoMarshaller (p. 2391), activemq::wireformat::openwire::marshal::generated::ShutdownInfoMarshaller (p. 2436), activemq::wireformat::openwire::marshal::generated::SubscriptionInfoMarshaller (p. 2637), activemq::wireformat::openwire::marshal::generated::TransactionInfoMarshaller (p. 2780), activemq::wireformat::openwire::marshal::generated::WireFormatInfoMarshaller (p. 2902), activemq::wireformat::openwire::

`::marshal::generated::XATransactionIdMarshaller` (p. 2944), `activemq::wireformat::openwire::marshal::generated::ActiveMQDestinationMarshaller` (p. 259), `activemq::wireformat::openwire::marshal::generated::ActiveMQTempDestinationMarshaller` (p. 384), `activemq::wireformat::openwire::marshal::generated::BaseCommandMarshaller` (p. 500), `activemq::wireformat::openwire::marshal::generated::MessageMarshaller` (p. 1919), and `activemq::wireformat::openwire::marshal::generated::TransactionIdMarshaller` (p. 2771).

6.185.3.4 `virtual void activemq::wireformat::openwire::marshal::DataStreamMarshaller::looseUnmarshal (OpenWireFormat * format, commands::DataStructure * command, decaf::io::DataInputStream * dis)`
`[pure virtual]`

Loose Un-marhsal to the given stream.

Parameters

<i>format</i>	- The OpenWireFormat properties
<i>command</i>	- the object to Un-Marshal
<i>dis</i>	- the DataInputStream to Un-Marshal from

Exceptions

<i>IOException</i>	if an error occurs.
--------------------	---------------------

Implemented in `activemq::wireformat::openwire::marshal::generated::ActiveMQBlobMessageMarshaller` (p. 164), `activemq::wireformat::openwire::marshal::generated::ActiveMQBytesMessageMarshaller` (p. 185), `activemq::wireformat::openwire::marshal::generated::ActiveMQMapMessageMarshaller` (p. 286), `activemq::wireformat::openwire::marshal::generated::ActiveMQMessageMarshaller` (p. 293), `activemq::wireformat::openwire::marshal::generated::ActiveMQObjectMessageMarshaller` (p. 306), `activemq::wireformat::openwire::marshal::generated::ActiveMQQueueMarshaller` (p. 331), `activemq::wireformat::openwire::marshal::generated::ActiveMQStreamMessageMarshaller` (p. 377), `activemq::wireformat::openwire::marshal::generated::ActiveMQTempQueueMarshaller` (p. 394), `activemq::wireformat::openwire::marshal::generated::ActiveMQTempTopicMarshaller` (p. 403), `activemq::wireformat::openwire::marshal::generated::ActiveMQTextMessageMarshaller` (p. 412), `activemq::wireformat::openwire::marshal::generated::ActiveMQTopicMarshaller` (p. 421), `activemq::wireformat::openwire::marshal::generated::BrokerIdMarshaller` (p. 570), `activemq::wireformat::openwire::marshal::generated::BrokerInfoMarshaller` (p. 580), `activemq::wireformat::openwire::marshal::generated::ConnectionControlMarshaller` (p. 946), `activemq::wireformat::openwire::marshal::generated::ConnectionErrorMarshaller` (p. 953), `activemq::wireformat::openwire::marshal::generated::ConnectionIdMarshaller` (p. 966), `activemq::wireformat::openwire::marshal::generated::ConnectionInfoMarshaller` (p. 976), `activemq::wireformat::openwire::marshal::generated::ConsumerControlMarshaller` (p. 998), `activemq::wireformat::openwire::marshal::generated::ConsumerIdMarshaller` (p. 1007), `activemq::wireformat::openwire::marshal::generated::ConsumerInfoMarshaller` (p. 1020),

activemq::wireformat::openwire::marshal::generated::ControlCommandMarshaller (p. 1028), activemq::wireformat::openwire::marshal::generated::DataArrayResponseMarshaller (p. 1074), activemq::wireformat::openwire::marshal::generated::DataResponseMarshaller (p. 1117), activemq::wireformat::openwire::marshal::generated::DestinationInfoMarshaller (p. 1220), activemq::wireformat::openwire::marshal::generated::DiscoveryEventMarshaller (p. 1232), activemq::wireformat::openwire::marshal::generated::ExceptionResponseMarshaller (p. 1293), activemq::wireformat::openwire::marshal::generated::FlushCommandMarshaller (p. 1386), activemq::wireformat::openwire::marshal::generated::IntegerResponseMarshaller (p. 1521), activemq::wireformat::openwire::marshal::generated::JournalQueueAckMarshaller (p. 1566), activemq::wireformat::openwire::marshal::generated::JournalTopicAckMarshaller (p. 1575), activemq::wireformat::openwire::marshal::generated::JournalTraceMarshaller (p. 1582), activemq::wireformat::openwire::marshal::generated::JournalTransactionMarshaller (p. 1589), activemq::wireformat::openwire::marshal::generated::KeepAliveInfoMarshaller (p. 1596), activemq::wireformat::openwire::marshal::generated::LastPartialCommandMarshaller (p. 1610), activemq::wireformat::openwire::marshal::generated::LocalTransactionIdMarshaller (p. 1680), activemq::wireformat::openwire::marshal::generated::MessageAckMarshaller (p. 1875), activemq::wireformat::openwire::marshal::generated::MessageDispatchMarshaller (p. 1893), activemq::wireformat::openwire::marshal::generated::MessageDispatchNotificationMarshaller (p. 1902), activemq::wireformat::openwire::marshal::generated::MessageIdMarshaller (p. 1914), activemq::wireformat::openwire::marshal::generated::MessagePullMarshaller (p. 1946), activemq::wireformat::openwire::marshal::generated::NetworkBridgeFilterMarshaller (p. 1976), activemq::wireformat::openwire::marshal::generated::PartialCommandMarshaller (p. 2081), activemq::wireformat::openwire::marshal::generated::ProducerAckMarshaller (p. 2177), activemq::wireformat::openwire::marshal::generated::ProducerIdMarshaller (p. 2188), activemq::wireformat::openwire::marshal::generated::ProducerInfoMarshaller (p. 2197), activemq::wireformat::openwire::marshal::generated::RemoveInfoMarshaller (p. 2273), activemq::wireformat::openwire::marshal::generated::RemoveSubscriptionInfoMarshaller (p. 2282), activemq::wireformat::openwire::marshal::generated::ReplayCommandMarshaller (p. 2289), activemq::wireformat::openwire::marshal::generated::ResponseMarshaller (p. 2310), activemq::wireformat::openwire::marshal::generated::SessionIdMarshaller (p. 2384), activemq::wireformat::openwire::marshal::generated::SessionInfoMarshaller (p. 2392), activemq::wireformat::openwire::marshal::generated::ShutdownInfoMarshaller (p. 2436), activemq::wireformat::openwire::marshal::generated::SubscriptionInfoMarshaller (p. 2637), activemq::wireformat::openwire::marshal::generated::TransactionInfoMarshaller (p. 2780), activemq::wireformat::openwire::marshal::generated::WireFormatInfoMarshaller (p. 2902), activemq::wireformat::openwire::marshal::generated::XATransactionIdMarshaller (p. 2945), activemq::wireformat::openwire::marshal::generated::ActiveMQDestinationMarshaller (p. 259), activemq::wireformat::openwire::marshal::generated::ActiveMQTempDestinationMarshaller (p. 384), activemq::wireformat::openwire::marshal::generated::BaseCommandMarshaller (p. 501), activemq::wireformat::openwire::marshal::generated::MessageMarshaller (p. 1919), and activemq::wireformat::openwire::marshal::generated::TransactionIdMarshaller (p. 2772).

```
6.185.3.5 virtual int activemq::wireformat::openwire::marshal::Data-
StreamMarshaller::tightMarshal1 ( OpenWireFormat * format,
commands::DataStructure * command, utils::BooleanStream * bs )
[pure virtual]
```

Tight Marshal to the given stream.

Parameters

<i>format</i>	The OpenwireFormat properties
<i>command</i>	The object to Marshal
<i>bs</i>	The boolean stream to marshal to.

Exceptions

<i>IOException</i>	if an error occurs.
--------------------	---------------------

Implemented in `activemq::wireformat::openwire::marshal::generated::ActiveMQBlobMessageMarshaller` (p. 164), `activemq::wireformat::openwire::marshal::generated::ActiveMQBytesMessageMarshaller` (p. 186), `activemq::wireformat::openwire::marshal::generated::ActiveMQMapMessageMarshaller` (p. 286), `activemq::wireformat::openwire::marshal::generated::ActiveMQMessageMarshaller` (p. 293), `activemq::wireformat::openwire::marshal::generated::ActiveMQObjectMessageMarshaller` (p. 306), `activemq::wireformat::openwire::marshal::generated::ActiveMQQueueMarshaller` (p. 331), `activemq::wireformat::openwire::marshal::generated::ActiveMQStreamMessageMarshaller` (p. 377), `activemq::wireformat::openwire::marshal::generated::ActiveMQTempQueueMarshaller` (p. 394), `activemq::wireformat::openwire::marshal::generated::ActiveMQTempTopicMarshaller` (p. 403), `activemq::wireformat::openwire::marshal::generated::ActiveMQTextMessageMarshaller` (p. 413), `activemq::wireformat::openwire::marshal::generated::ActiveMQTopicMarshaller` (p. 421), `activemq::wireformat::openwire::marshal::generated::BrokerIdMarshaller` (p. 570), `activemq::wireformat::openwire::marshal::generated::BrokerInfoMarshaller` (p. 581), `activemq::wireformat::openwire::marshal::generated::ConnectionControlMarshaller` (p. 946), `activemq::wireformat::openwire::marshal::generated::ConnectionErrorMarshaller` (p. 954), `activemq::wireformat::openwire::marshal::generated::ConnectionIdMarshaller` (p. 966), `activemq::wireformat::openwire::marshal::generated::ConnectionInfoMarshaller` (p. 976), `activemq::wireformat::openwire::marshal::generated::ConsumerControlMarshaller` (p. 999), `activemq::wireformat::openwire::marshal::generated::ConsumerIdMarshaller` (p. 1007), `activemq::wireformat::openwire::marshal::generated::ConsumerInfoMarshaller` (p. 1020), `activemq::wireformat::openwire::marshal::generated::ControlCommandMarshaller` (p. 1028), `activemq::wireformat::openwire::marshal::generated::DataArrayResponseMarshaller` (p. 1074), `activemq::wireformat::openwire::marshal::generated::DataResponseMarshaller` (p. 1118), `activemq::wireformat::openwire::marshal::generated::DestinationInfoMarshaller` (p. 1220), `activemq::wireformat::openwire::marshal::generated::DiscoveryEventMarshaller` (p. 1232), `activemq::wireformat::openwire::marshal::generated::ExceptionResponseMarshaller` (p. 1293), `activemq::wireformat::openwire::marshal::generated::FlushCommand-`

Marshaller (p. 1386), activemq::wireformat::openwire::marshal::generated::IntegerResponseMarshaller (p. 1521), activemq::wireformat::openwire::marshal::generated::JournalQueueAckMarshaller (p. 1566), activemq::wireformat::openwire::marshal::generated::JournalTopicAckMarshaller (p. 1575), activemq::wireformat::openwire::marshal::generated::JournalTraceMarshaller (p. 1582), activemq::wireformat::openwire::marshal::generated::JournalTransactionMarshaller (p. 1590), activemq::wireformat::openwire::marshal::generated::KeepAliveInfoMarshaller (p. 1597), activemq::wireformat::openwire::marshal::generated::LastPartialCommandMarshaller (p. 1611), activemq::wireformat::openwire::marshal::generated::LocalTransactionIdMarshaller (p. 1681), activemq::wireformat::openwire::marshal::generated::MessageAckMarshaller (p. 1875), activemq::wireformat::openwire::marshal::generated::MessageDispatchMarshaller (p. 1893), activemq::wireformat::openwire::marshal::generated::MessageDispatchNotificationMarshaller (p. 1902), activemq::wireformat::openwire::marshal::generated::MessageIdMarshaller (p. 1915), activemq::wireformat::openwire::marshal::generated::MessagePullMarshaller (p. 1946), activemq::wireformat::openwire::marshal::generated::NetworkBridgeFilterMarshaller (p. 1977), activemq::wireformat::openwire::marshal::generated::PartialCommandMarshaller (p. 2082), activemq::wireformat::openwire::marshal::generated::ProducerAckMarshaller (p. 2177), activemq::wireformat::openwire::marshal::generated::ProducerIdMarshaller (p. 2189), activemq::wireformat::openwire::marshal::generated::ProducerInfoMarshaller (p. 2197), activemq::wireformat::openwire::marshal::generated::RemoveInfoMarshaller (p. 2274), activemq::wireformat::openwire::marshal::generated::RemoveSubscriptionInfoMarshaller (p. 2282), activemq::wireformat::openwire::marshal::generated::ReplayCommandMarshaller (p. 2290), activemq::wireformat::openwire::marshal::generated::ResponseMarshaller (p. 2310), activemq::wireformat::openwire::marshal::generated::SessionIdMarshaller (p. 2385), activemq::wireformat::openwire::marshal::generated::SessionInfoMarshaller (p. 2392), activemq::wireformat::openwire::marshal::generated::ShutdownInfoMarshaller (p. 2437), activemq::wireformat::openwire::marshal::generated::SubscriptionInfoMarshaller (p. 2638), activemq::wireformat::openwire::marshal::generated::TransactionInfoMarshaller (p. 2781), activemq::wireformat::openwire::marshal::generated::WireFormatInfoMarshaller (p. 2902), activemq::wireformat::openwire::marshal::generated::XATransactionIdMarshaller (p. 2945), activemq::wireformat::openwire::marshal::generated::ActiveMQDestinationMarshaller (p. 260), activemq::wireformat::openwire::marshal::generated::ActiveMQTempDestinationMarshaller (p. 385), activemq::wireformat::openwire::marshal::generated::BaseCommandMarshaller (p. 503), activemq::wireformat::openwire::marshal::generated::MessageMarshaller (p. 1920), and activemq::wireformat::openwire::marshal::generated::TransactionIdMarshaller (p. 2772).

6.185.3.6 virtual void activemq::wireformat::openwire::marshal::DataStreamMarshaller::tightMarshal2 (OpenWireFormat * *format*, commands::DataStructure * *command*, decaf::io::DataOutputStream * *ds*, utils::BooleanStream * *bs*) [pure virtual]

Tight Marshal to the given stream.

Parameters

<i>format</i>	- The OpenwireFormat properties
<i>command</i>	- the object to Marshal
<i>ds</i>	- the DataOutputStream to Marshal to
<i>bs</i>	- boolean stream to marshal to.

Exceptions

<i>IOException</i>	if an error occurs.
--------------------	---------------------

Implemented in `activemq::wireformat::openwire::marshal::generated::ActiveMQBlobMessageMarshaller` (p. 165), `activemq::wireformat::openwire::marshal::generated::ActiveMQBytesMessageMarshaller` (p. 186), `activemq::wireformat::openwire::marshal::generated::ActiveMQMapMessageMarshaller` (p. 287), `activemq::wireformat::openwire::marshal::generated::ActiveMQMessageMarshaller` (p. 294), `activemq::wireformat::openwire::marshal::generated::ActiveMQObjectMessageMarshaller` (p. 307), `activemq::wireformat::openwire::marshal::generated::ActiveMQQueueMarshaller` (p. 332), `activemq::wireformat::openwire::marshal::generated::ActiveMQStreamMessageMarshaller` (p. 378), `activemq::wireformat::openwire::marshal::generated::ActiveMQTempQueueMarshaller` (p. 395), `activemq::wireformat::openwire::marshal::generated::ActiveMQTempTopicMarshaller` (p. 404), `activemq::wireformat::openwire::marshal::generated::ActiveMQTextMessageMarshaller` (p. 413), `activemq::wireformat::openwire::marshal::generated::ActiveMQTopicMarshaller` (p. 422), `activemq::wireformat::openwire::marshal::generated::BrokerIdMarshaller` (p. 571), `activemq::wireformat::openwire::marshal::generated::BrokerInfoMarshaller` (p. 581), `activemq::wireformat::openwire::marshal::generated::ConnectionControlMarshaller` (p. 947), `activemq::wireformat::openwire::marshal::generated::ConnectionErrorMarshaller` (p. 954), `activemq::wireformat::openwire::marshal::generated::ConnectionIdMarshaller` (p. 967), `activemq::wireformat::openwire::marshal::generated::ConnectionInfoMarshaller` (p. 977), `activemq::wireformat::openwire::marshal::generated::ConsumerControlMarshaller` (p. 999), `activemq::wireformat::openwire::marshal::generated::ConsumerIdMarshaller` (p. 1008), `activemq::wireformat::openwire::marshal::generated::ConsumerInfoMarshaller` (p. 1021), `activemq::wireformat::openwire::marshal::generated::ControlCommandMarshaller` (p. 1029), `activemq::wireformat::openwire::marshal::generated::DataArrayResponseMarshaller` (p. 1075), `activemq::wireformat::openwire::marshal::generated::DataResponseMarshaller` (p. 1118), `activemq::wireformat::openwire::marshal::generated::DestinationInfoMarshaller` (p. 1221), `activemq::wireformat::openwire::marshal::generated::DiscoveryEventMarshaller` (p. 1233), `activemq::wireformat::openwire::marshal::generated::ExceptionResponseMarshaller` (p. 1293), `activemq::wireformat::openwire::marshal::generated::FlushCommandMarshaller` (p. 1386), `activemq::wireformat::openwire::marshal::generated::IntegerResponseMarshaller` (p. 1522), `activemq::wireformat::openwire::marshal::generated::JournalQueueAckMarshaller` (p. 1567), `activemq::wireformat::openwire::marshal::generated::JournalTopicAckMarshaller` (p. 1576), `activemq::wireformat::openwire::marshal::generated::JournalTraceMarshaller` (p. 1582), `activemq::wireformat::openwire::marshal::generated::JournalTransactionMarshaller` (p. 1590), `activemq::wireformat::openwire::marshal::generated::-`

KeepAliveInfoMarshaller (p. 1597), activemq::wireformat::openwire::marshal::generated::LastPartialCommandMarshaller (p. 1611), activemq::wireformat::openwire::marshal::generated::LocalTransactionIdMarshaller (p. 1681), activemq::wireformat::openwire::marshal::generated::MessageAckMarshaller (p. 1876), activemq::wireformat::openwire::marshal::generated::MessageDispatchMarshaller (p. 1893), activemq::wireformat::openwire::marshal::generated::MessageDispatchNotificationMarshaller (p. 1902), activemq::wireformat::openwire::marshal::generated::MessageIdMarshaller (p. 1915), activemq::wireformat::openwire::marshal::generated::MessagePullMarshaller (p. 1947), activemq::wireformat::openwire::marshal::generated::NetworkBridgeFilterMarshaller (p. 1977), activemq::wireformat::openwire::marshal::generated::PartialCommandMarshaller (p. 2082), activemq::wireformat::openwire::marshal::generated::ProducerAckMarshaller (p. 2178), activemq::wireformat::openwire::marshal::generated::ProducerIdMarshaller (p. 2189), activemq::wireformat::openwire::marshal::generated::ProducerInfoMarshaller (p. 2198), activemq::wireformat::openwire::marshal::generated::RemoveInfoMarshaller (p. 2274), activemq::wireformat::openwire::marshal::generated::RemoveSubscriptionInfoMarshaller (p. 2283), activemq::wireformat::openwire::marshal::generated::ReplayCommandMarshaller (p. 2290), activemq::wireformat::openwire::marshal::generated::ResponseMarshaller (p. 2311), activemq::wireformat::openwire::marshal::generated::SessionIdMarshaller (p. 2385), activemq::wireformat::openwire::marshal::generated::SessionInfoMarshaller (p. 2393), activemq::wireformat::openwire::marshal::generated::ShutdownInfoMarshaller (p. 2437), activemq::wireformat::openwire::marshal::generated::SubscriptionInfoMarshaller (p. 2638), activemq::wireformat::openwire::marshal::generated::TransactionInfoMarshaller (p. 2781), activemq::wireformat::openwire::marshal::generated::WireFormatInfoMarshaller (p. 2903), activemq::wireformat::openwire::marshal::generated::XATransactionIdMarshaller (p. 2945), activemq::wireformat::openwire::marshal::generated::ActiveMQDestinationMarshaller (p. 260), activemq::wireformat::openwire::marshal::generated::ActiveMQTempDestinationMarshaller (p. 385), activemq::wireformat::openwire::marshal::generated::BaseCommandMarshaller (p. 504), activemq::wireformat::openwire::marshal::generated::MessageMarshaller (p. 1921), and activemq::wireformat::openwire::marshal::generated::TransactionIdMarshaller (p. 2773).

6.185.3.7 virtual void activemq::wireformat::openwire::marshal::DataStreamMarshaller::tightUnmarshal (OpenWireFormat * *format*, commands::DataStructure * *command*, decaf::io::DataInputStream * *dis*, utils::BooleanStream * *bs*) [pure virtual]

Tight Un-marhsal to the given stream.

Parameters

<i>format</i>	- The OpenWireFormat properties
<i>command</i>	- the object to Un-Marshal
<i>dis</i>	- the DataInputStream to Un-Marshal from
<i>bs</i>	- boolean stream to unmarshal from.

Exceptions

<i>IOException</i> if an error occurs.
--

Implemented in `activemq::wireformat::openwire::marshal::generated::ActiveMQBlobMessageMarshaller` (p. 165), `activemq::wireformat::openwire::marshal::generated::ActiveMQBytesMessageMarshaller` (p. 187), `activemq::wireformat::openwire::marshal::generated::ActiveMQMapMessageMarshaller` (p. 287), `activemq::wireformat::openwire::marshal::generated::ActiveMQMessageMarshaller` (p. 294), `activemq::wireformat::openwire::marshal::generated::ActiveMQObjectMessageMarshaller` (p. 307), `activemq::wireformat::openwire::marshal::generated::ActiveMQQueueMarshaller` (p. 332), `activemq::wireformat::openwire::marshal::generated::ActiveMQStreamMessageMarshaller` (p. 378), `activemq::wireformat::openwire::marshal::generated::ActiveMQTempQueueMarshaller` (p. 395), `activemq::wireformat::openwire::marshal::generated::ActiveMQTempTopicMarshaller` (p. 404), `activemq::wireformat::openwire::marshal::generated::ActiveMQTextMessageMarshaller` (p. 414), `activemq::wireformat::openwire::marshal::generated::ActiveMQTopicMarshaller` (p. 422), `activemq::wireformat::openwire::marshal::generated::BrokerIdMarshaller` (p. 571), `activemq::wireformat::openwire::marshal::generated::BrokerInfoMarshaller` (p. 582), `activemq::wireformat::openwire::marshal::generated::ConnectionControlMarshaller` (p. 947), `activemq::wireformat::openwire::marshal::generated::ConnectionErrorMarshaller` (p. 955), `activemq::wireformat::openwire::marshal::generated::ConnectionIdMarshaller` (p. 967), `activemq::wireformat::openwire::marshal::generated::ConnectionInfoMarshaller` (p. 977), `activemq::wireformat::openwire::marshal::generated::ConsumerControlMarshaller` (p. 1000), `activemq::wireformat::openwire::marshal::generated::ConsumerIdMarshaller` (p. 1008), `activemq::wireformat::openwire::marshal::generated::ConsumerInfoMarshaller` (p. 1021), `activemq::wireformat::openwire::marshal::generated::ControlCommandMarshaller` (p. 1029), `activemq::wireformat::openwire::marshal::generated::DataArrayResponseMarshaller` (p. 1075), `activemq::wireformat::openwire::marshal::generated::DataResponseMarshaller` (p. 1119), `activemq::wireformat::openwire::marshal::generated::DestinationInfoMarshaller` (p. 1221), `activemq::wireformat::openwire::marshal::generated::DiscoveryEventMarshaller` (p. 1233), `activemq::wireformat::openwire::marshal::generated::ExceptionResponseMarshaller` (p. 1294), `activemq::wireformat::openwire::marshal::generated::FlushCommandMarshaller` (p. 1387), `activemq::wireformat::openwire::marshal::generated::IntegerResponseMarshaller` (p. 1522), `activemq::wireformat::openwire::marshal::generated::JournalQueueAckMarshaller` (p. 1567), `activemq::wireformat::openwire::marshal::generated::JournalTopicAckMarshaller` (p. 1576), `activemq::wireformat::openwire::marshal::generated::JournalTraceMarshaller` (p. 1583), `activemq::wireformat::openwire::marshal::generated::JournalTransactionMarshaller` (p. 1591), `activemq::wireformat::openwire::marshal::generated::KeepAliveInfoMarshaller` (p. 1598), `activemq::wireformat::openwire::marshal::generated::LastPartialCommandMarshaller` (p. 1612), `activemq::wireformat::openwire::marshal::generated::LocalTransactionIdMarshaller` (p. 1682), `activemq::wireformat::openwire::marshal::generated::MessageAckMarshaller` (p. 1876), `activemq::wireformat::openwire::marshal::generated::MessageDispatchMarshaller` (p. 1894), `activemq::wireformat::openwire::marshal::generated::MessageDispatchNotificationMarshaller` (p. 1903), `activemq-`

::wireformat::openwire::marshal::generated::MessageIdMarshaller (p. 1916),
 activemq::wireformat::openwire::marshal::generated::MessagePullMarshaller
 (p. 1947), activemq::wireformat::openwire::marshal::generated::NetworkBridge-
 FilterMarshaller (p. 1978), activemq::wireformat::openwire::marshal::generated::-
 PartialCommandMarshaller (p. 2083), activemq::wireformat::openwire::marshal-
 ::generated::ProducerAckMarshaller (p. 2178), activemq::wireformat::openwire-
 ::marshal::generated::ProducerIdMarshaller (p. 2189), activemq::wireformat-
 ::openwire::marshal::generated::ProducerInfoMarshaller (p. 2198), activemq-
 ::wireformat::openwire::marshal::generated::RemoveInfoMarshaller (p. 2275),
 activemq::wireformat::openwire::marshal::generated::RemoveSubscription-
 InfoMarshaller (p. 2283), activemq::wireformat::openwire::marshal::generated-
 ::ReplayCommandMarshaller (p. 2290), activemq::wireformat::openwire-
 ::marshal::generated::ResponseMarshaller (p. 2311), activemq::wireformat-
 ::openwire::marshal::generated::SessionIdMarshaller (p. 2386), activemq-
 ::wireformat::openwire::marshal::generated::SessionInfoMarshaller (p. 2393),
 activemq::wireformat::openwire::marshal::generated::ShutdownInfoMarshaller
 (p. 2438), activemq::wireformat::openwire::marshal::generated::Subscription-
 InfoMarshaller (p. 2638), activemq::wireformat::openwire::marshal::generated::-
 TransactionInfoMarshaller (p. 2781), activemq::wireformat::openwire::marshal-
 ::generated::WireFormatInfoMarshaller (p. 2903), activemq::wireformat::openwire-
 ::marshal::generated::XATransactionIdMarshaller (p. 2946), activemq::wireformat-
 ::openwire::marshal::generated::ActiveMQDestinationMarshaller (p. 261),
 activemq::wireformat::openwire::marshal::generated::ActiveMQTempDestination-
 Marshaller (p. 386), activemq::wireformat::openwire::marshal::generated::-
 BaseCommandMarshaller (p. 505), activemq::wireformat::openwire::marshal-
 ::generated::MessageMarshaller (p. 1921), and activemq::wireformat::openwire-
 ::marshal::generated::TransactionIdMarshaller (p. 2773).

The documentation for this class was generated from the following file:

- src/main/activemq/wireformat/openwire/marshal/**DataStreamMarshaller.h**

6.186 activemq::commands::DataStructure Class Reference

```
#include <src/main/activemq/commands/DataStructure.h>
```

Inheritance diagram for activemq::commands::DataStructure:

Public Member Functions

- virtual **~DataStructure** ()
- virtual unsigned char **getDataStructureType** () const =0
*Get the **DataStructure** (p. 1133) Type as defined in CommandTypes.h.*
- virtual **DataStructure** * **cloneDataStructure** () const =0
Clone this obbect and return a new instance that the caller now owns, this will be an exact copy of this one.

- virtual void **copyDataStructure** (const **DataSet** *src)=0
Copy the contents of the passed object into this objects members, overwriting any existing data.
- virtual std::string **toString** () const =0
*Returns a string containing the information for this **DataSet** (p. 1133) such as its type and value of its elements.*
- virtual bool **equals** (const **DataSet** *value) const =0
*Compares the **DataSet** (p. 1133) passed in to this one, and returns if they are equivalent.*

6.186.1 Constructor & Destructor Documentation

6.186.1.1 virtual **activemq::commands::DataSet::~DataSet** ()
[inline, virtual]

6.186.2 Member Function Documentation

6.186.2.1 virtual **DataSet*** **activemq::commands::DataSet::cloneDataSet** () const [pure virtual]

Clone this object and return a new instance that the caller now owns, this will be an exact copy of this one.

Returns

new copy of this object.

Implemented in **activemq::commands::Message** (p. 1826), **activemq::commands::ActiveMQDestination** (p. 249), **activemq::commands::ConsumerInfo** (p. 1011), **activemq::commands::MessageId** (p. 1910), **activemq::commands::SessionId** (p. 2380), **activemq::commands::ActiveMQBytesMessage** (p. 169), **activemq::commands::BrokerInfo** (p. 574), **activemq::commands::ConnectionInfo** (p. 969), **activemq::commands::ProducerId** (p. 2183), **activemq::commands::ConnectionId** (p. 961), **activemq::commands::MessageAck** (p. 1869), **activemq::commands::ConsumerId** (p. 1002), **activemq::commands::ConsumerControl** (p. 993), **activemq::commands::JournalTopicAck** (p. 1569), **activemq::commands::ProducerInfo** (p. 2191), **activemq::commands::BrokerError** (p. 560), **activemq::commands::ConnectionControl** (p. 940), **activemq::commands::DestinationInfo** (p. 1215), **activemq::commands::MessagePull** (p. 1941), **activemq::commands::SessionInfo** (p. 2387), **activemq::commands::MessageDispatch** (p. 1883), **activemq::commands::MessageDispatchNotification** (p. 1896), **activemq::commands::SubscriptionInfo** (p. 2632), **activemq::commands::XATransactionId** (p. 2938), **activemq::commands::TransactionInfo** (p. 2775), **activemq::commands::ConnectionError** (p. 949), **activemq::commands::JournalQueueAck** (p. 1562), **activemq::commands::JournalTransaction** (p. 1585), **activemq::commands::RemoveSubscriptionInfo** (p. 2277), **activemq::commands::ActiveMQStreamMessage** (p. 362), **activemq::commands::LocalTransactionId** (p. 1676), **activemq::commands::NetworkBridgeFilter** (p. 1972), **activemq::commands::ProducerAck**

(p. 2172), **activemq::commands::RemoveInfo** (p. 2269), **activemq::commands::DataArrayResponse** (p. 1070), **activemq::commands::DataResponse** (p. 1113), **activemq::commands::DiscoveryEvent** (p. 1228), **activemq::commands::ExceptionResponse** (p. 1289), **activemq::commands::PartialCommand** (p. 2077), **activemq::commands::ReplayCommand** (p. 2285), **activemq::commands::BrokerId** (p. 565), **activemq::commands::ControlCommand** (p. 1023), **activemq::commands::IntegerResponse** (p. 1517), **activemq::commands::JournalTrace** (p. 1578), **activemq::commands::Response** (p. 2299), **activemq::commands::ActiveMQMapMessage** (p. 272), **activemq::commands::FlushCommand** (p. 1382), **activemq::commands::KeepAliveInfo** (p. 1592), **activemq::commands::LastPartialCommand** (p. 1606), **activemq::commands::ShutdownInfo** (p. 2432), **activemq::commands::TransactionId** (p. 2768), **activemq::commands::ActiveMQTempDestination** (p. 380), **activemq::commands::ActiveMQBlobMessage** (p. 158), **activemq::commands::WireFormatInfo** (p. 2892), **activemq::commands::ActiveMQQueue** (p. 322), **activemq::commands::ActiveMQTopic** (p. 416), **activemq::commands::ActiveMQTextMessage** (p. 407), **activemq::commands::ActiveMQMessage** (p. 289), **activemq::commands::ActiveMQTempQueue** (p. 388), **activemq::commands::ActiveMQTempTopic** (p. 397), **activemq::commands::ActiveMQObjectMessage** (p. 302), and **activemq::commands::BooleanExpression** (p. 551).

6.186.2.2 **virtual void activemq::commands::DataStructure::copyDataStructure (const DataStructure * src)** [pure virtual]

Copy the contents of the passed object into this objects members, overwriting any existing data.

Returns

src - Source Object

Implemented in **activemq::commands::Message** (p. 1827), **activemq::commands::ActiveMQDestination** (p. 249), **activemq::commands::ConsumerInfo** (p. 1012), **activemq::commands::BrokerError** (p. 560), **activemq::commands::MessageId** (p. 1910), **activemq::commands::SessionId** (p. 2380), **activemq::commands::ActiveMQBytesMessage** (p. 169), **activemq::commands::BrokerInfo** (p. 574), **activemq::commands::ConnectionInfo** (p. 970), **activemq::commands::ProducerId** (p. 2184), **activemq::commands::ConnectionId** (p. 962), **activemq::commands::MessageAck** (p. 1869), **activemq::commands::ConsumerId** (p. 1002), **activemq::commands::ConsumerControl** (p. 993), **activemq::commands::JournalTopicAck** (p. 1570), **activemq::commands::ProducerInfo** (p. 2192), **activemq::commands::ConnectionControl** (p. 940), **activemq::commands::DestinationInfo** (p. 1215), **activemq::commands::MessagePull** (p. 1941), **activemq::commands::SessionInfo** (p. 2388), **activemq::commands::MessageDispatch** (p. 1883), **activemq::commands::MessageDispatchNotification** (p. 1896), **activemq::commands::SubscriptionInfo** (p. 2632), **activemq::commands::XATransactionId** (p. 2939), **activemq::commands::TransactionInfo** (p. 2775), **activemq::commands::ConnectionError** (p. 949), **activemq::commands::JournalQueueAck** (p. 1562), **activemq::commands::JournalTransaction** (p. 1585), **activemq::commands::**

RemoveSubscriptionInfo (p. 2277), **activemq::commands::ActiveMQStreamMessage** (p. 362), **activemq::commands::LocalTransactionId** (p. 1676), **activemq::commands::NetworkBridgeFilter** (p. 1972), **activemq::commands::ProducerAck** (p. 2173), **activemq::commands::RemoveInfo** (p. 2269), **activemq::commands::DataArrayResponse** (p. 1070), **activemq::commands::DataResponse** (p. 1113), **activemq::commands::DiscoveryEvent** (p. 1228), **activemq::commands::ExceptionResponse** (p. 1289), **activemq::commands::PartialCommand** (p. 2077), **activemq::commands::ReplayCommand** (p. 2285), **activemq::commands::ActiveMQTempDestination** (p. 380), **activemq::commands::BrokerId** (p. 566), **activemq::commands::ControlCommand** (p. 1024), **activemq::commands::IntegerResponse** (p. 1517), **activemq::commands::JournalTrace** (p. 1578), **activemq::commands::Response** (p. 2299), **activemq::commands::ActiveMQMapMessage** (p. 272), **activemq::commands::FlushCommand** (p. 1382), **activemq::commands::KeepAliveInfo** (p. 1592), **activemq::commands::LastPartialCommand** (p. 1607), **activemq::commands::ShutdownInfo** (p. 2432), **activemq::commands::TransactionId** (p. 2768), **activemq::commands::ActiveMQBlobMessage** (p. 158), **activemq::commands::WireFormatInfo** (p. 2892), **activemq::commands::ActiveMQQueue** (p. 323), **activemq::commands::ActiveMQTopic** (p. 416), **activemq::commands::ActiveMQTextMessage** (p. 407), **activemq::commands::ActiveMQTempQueue** (p. 388), **activemq::commands::ActiveMQTempTopic** (p. 398), **activemq::commands::ActiveMQObjectMessage** (p. 302), **activemq::commands::BaseCommand** (p. 493), and **activemq::commands::ActiveMQMessage** (p. 289).

```
6.186.2.3 virtual bool activemq::commands::DataStructure::equals ( const
                DataStructure * value ) const [pure virtual]
```

Compares the **DataStructure** (p. 1133) passed in to this one, and returns if they are equivalent.

Equivalent here means that they are of the same type, and that each element of the objects are the same.

Returns

true if **DataStructure** (p. 1133)'s are Equal.

Implemented in **activemq::commands::Message** (p. 1827), **activemq::commands::ActiveMQDestination** (p. 250), **activemq::commands::ConsumerInfo** (p. 1012), **activemq::commands::MessageId** (p. 1910), **activemq::commands::SessionId** (p. 2380), **activemq::commands::BaseCommand** (p. 494), **activemq::commands::ActiveMQBytesMessage** (p. 169), **activemq::commands::BrokerInfo** (p. 574), **activemq::commands::ConnectionInfo** (p. 970), **activemq::commands::ProducerId** (p. 2184), **activemq::commands::ConnectionId** (p. 962), **activemq::commands::MessageAck** (p. 1869), **activemq::commands::ConsumerId** (p. 1003), **activemq::commands::ConsumerControl** (p. 994), **activemq::commands::JournalTopicAck** (p. 1570), **activemq::commands::ProducerInfo** (p. 2192), **activemq::commands::ConnectionControl** (p. 940), **activemq::commands::DestinationInfo** (p. 1215), **activemq::commands::MessagePull** (p. 1941), **activemq::commands::SessionInfo** (p. 2388), **activemq::commands::MessageDispatch** (p. 1883), **activemq-**

activemq::commands::MessageDispatchNotification (p. 1896), **activemq::commands::SubscriptionInfo** (p. 2633), **activemq::commands::XATransactionId** (p. 2939), **activemq::commands::TransactionInfo** (p. 2776), **activemq::commands::ConnectionError** (p. 949), **activemq::commands::JournalQueueAck** (p. 1562), **activemq::commands::JournalTransaction** (p. 1585), **activemq::commands::RemoveSubscriptionInfo** (p. 2277), **activemq::commands::ActiveMQStreamMessage** (p. 362), **activemq::commands::ActiveMQTempDestination** (p. 381), **activemq::commands::LocalTransactionId** (p. 1676), **activemq::commands::NetworkBridgeFilter** (p. 1973), **activemq::commands::ProducerAck** (p. 2173), **activemq::commands::RemoveInfo** (p. 2269), **activemq::commands::ActiveMQMapMessage** (p. 272), **activemq::commands::DataArrayResponse** (p. 1070), **activemq::commands::DataResponse** (p. 1114), **activemq::commands::DiscoveryEvent** (p. 1228), **activemq::commands::ExceptionResponse** (p. 1289), **activemq::commands::PartialCommand** (p. 2078), **activemq::commands::ReplayCommand** (p. 2285), **activemq::commands::BrokerId** (p. 566), **activemq::commands::ControlCommand** (p. 1024), **activemq::commands::IntegerResponse** (p. 1518), **activemq::commands::JournalTrace** (p. 1578), **activemq::commands::Response** (p. 2299), **activemq::commands::FlushCommand** (p. 1382), **activemq::commands::KeepAliveInfo** (p. 1593), **activemq::commands::LastPartialCommand** (p. 1607), **activemq::commands::ShutdownInfo** (p. 2433), **activemq::commands::TransactionId** (p. 2768), **activemq::commands::ActiveMQMessageTemplate< T >** (p. 297), **activemq::commands::ActiveMQMessageTemplate< cms::BytesMessage >** (p. 297), **activemq::commands::ActiveMQMessageTemplate< cms::MapMessage >** (p. 297), **activemq::commands::ActiveMQMessageTemplate< cms::Message >** (p. 297), **activemq::commands::ActiveMQMessageTemplate< cms::StreamMessage >** (p. 297), **activemq::commands::ActiveMQMessageTemplate< cms::TextMessage >** (p. 297), **activemq::commands::ActiveMQMessageTemplate< cms::ObjectMessage >** (p. 297), **activemq::commands::ActiveMQBlobMessage** (p. 159), **activemq::commands::WireFormatInfo** (p. 2892), **activemq::commands::ActiveMQQueue** (p. 323), **activemq::commands::ActiveMQTopic** (p. 416), **activemq::commands::ActiveMQTextMessage** (p. 408), **activemq::commands::ActiveMQTempQueue** (p. 389), **activemq::commands::ActiveMQTempTopic** (p. 398), **activemq::commands::ActiveMQObjectMessage** (p. 303), **activemq::commands::ActiveMQMessage** (p. 290), and **activemq::commands::BooleanExpression** (p. 552).

6.186.2.4 virtual unsigned char **activemq::commands::DataStructure::getDataStructureType** () const [pure virtual]

Get the **DataStructure** (p. 1133) Type as defined in CommandTypes.h.

Returns

The type of the data structure

Implemented in **activemq::commands::Message** (p. 1829), **activemq::commands::ActiveMQDestination** (p. 251), **activemq::commands::ConsumerInfo** (p. 1013), **activemq::commands::MessageId** (p. 1911), **activemq::commands::SessionId**

(p. 2381), `activemq::commands::ActiveMQBytesMessage` (p. 170), `activemq::commands::BrokerInfo` (p. 575), `activemq::commands::ConnectionInfo` (p. 971), `activemq::commands::ProducerId` (p. 2184), `activemq::commands::ConnectionId` (p. 962), `activemq::commands::MessageAck` (p. 1870), `activemq::commands::ConsumerId` (p. 1003), `activemq::commands::ConsumerControl` (p. 994), `activemq::commands::JournalTopicAck` (p. 1570), `activemq::commands::ProducerInfo` (p. 2192), `activemq::commands::ConnectionControl` (p. 941), `activemq::commands::DestinationInfo` (p. 1216), `activemq::commands::MessagePull` (p. 1942), `activemq::commands::SessionInfo` (p. 2388), `activemq::commands::MessageDispatch` (p. 1884), `activemq::commands::MessageDispatchNotification` (p. 1897), `activemq::commands::SubscriptionInfo` (p. 2633), `activemq::commands::XATransactionId` (p. 2940), `activemq::commands::TransactionInfo` (p. 2776), `activemq::commands::ConnectionError` (p. 950), `activemq::commands::JournalQueueAck` (p. 1563), `activemq::commands::JournalTransaction` (p. 1585), `activemq::commands::RemoveSubscriptionInfo` (p. 2278), `activemq::commands::ActiveMQStreamMessage` (p. 363), `activemq::commands::LocalTransactionId` (p. 1677), `activemq::commands::NetworkBridgeFilter` (p. 1973), `activemq::commands::ProducerAck` (p. 2173), `activemq::commands::RemoveInfo` (p. 2269), `activemq::commands::DataArrayResponse` (p. 1071), `activemq::commands::DataResponse` (p. 1114), `activemq::commands::DiscoveryEvent` (p. 1228), `activemq::commands::ExceptionResponse` (p. 1289), `activemq::commands::PartialCommand` (p. 2078), `activemq::commands::ReplayCommand` (p. 2286), `activemq::commands::BrokerError` (p. 561), `activemq::commands::BrokerId` (p. 566), `activemq::commands::ControlCommand` (p. 1024), `activemq::commands::IntegerResponse` (p. 1518), `activemq::commands::JournalTrace` (p. 1578), `activemq::commands::Response` (p. 2300), `activemq::commands::FlushCommand` (p. 1382), `activemq::commands::KeepAliveInfo` (p. 1593), `activemq::commands::LastPartialCommand` (p. 1607), `activemq::commands::ShutdownInfo` (p. 2433), `activemq::commands::TransactionId` (p. 2769), `activemq::commands::ActiveMQTempDestination` (p. 381), `activemq::commands::ActiveMQMapMessage` (p. 274), `activemq::commands::ActiveMQBlobMessage` (p. 159), `activemq::commands::WireFormatInfo` (p. 2893), `activemq::commands::ActiveMQQueue` (p. 324), `activemq::commands::ActiveMQTopic` (p. 417), `activemq::commands::ActiveMQTextMessage` (p. 408), `activemq::commands::ActiveMQTempQueue` (p. 390), `activemq::commands::ActiveMQTempTopic` (p. 399), `activemq::commands::ActiveMQObjectMessage` (p. 303), and `activemq::commands::ActiveMQMessage` (p. 290).

```
6.186.2.5  virtual std::string activemq::commands::DataStructure::toString ( ) const
           [pure virtual]
```

Returns a string containing the information for this **DataStructure** (p. 1133) such as its type and value of its elements.

Returns

formatted string useful for debugging.

Implemented in `activemq::commands::Message` (p. 1836), `activemq::commands::ActiveMQDestination` (p. 255), `activemq::commands::ConsumerInfo` (p. 1016), `activemq::commands::MessageId` (p. 1911), `activemq::commands::SessionId` (p. 2381), `activemq::commands::ActiveMQBytesMessage` (p. 178), `activemq::commands::BrokerInfo` (p. 577), `activemq::commands::ConnectionInfo` (p. 972), `activemq::commands::ProducerId` (p. 2185), `activemq::commands::ConnectionId` (p. 963), `activemq::commands::MessageAck` (p. 1871), `activemq::commands::ConsumerId` (p. 1004), `activemq::commands::ConsumerControl` (p. 995), `activemq::commands::JournalTopicAck` (p. 1572), `activemq::commands::ProducerInfo` (p. 2194), `activemq::commands::ConnectionControl` (p. 942), `activemq::commands::DestinationInfo` (p. 1217), `activemq::commands::MessagePull` (p. 1943), `activemq::commands::SessionInfo` (p. 2389), `activemq::commands::MessageDispatch` (p. 1885), `activemq::commands::MessageDispatchNotification` (p. 1898), `activemq::commands::SubscriptionInfo` (p. 2634), `activemq::commands::XATransactionId` (p. 2942), `activemq::commands::TransactionInfo` (p. 2777), `activemq::commands::ConnectionError` (p. 950), `activemq::commands::JournalQueueAck` (p. 1563), `activemq::commands::JournalTransaction` (p. 1586), `activemq::commands::RemoveSubscriptionInfo` (p. 2279), `activemq::commands::ActiveMQStreamMessage` (p. 369), `activemq::commands::ActiveMQTempDestination` (p. 382), `activemq::commands::LocalTransactionId` (p. 1678), `activemq::commands::NetworkBridgeFilter` (p. 1974), `activemq::commands::ProducerAck` (p. 2174), `activemq::commands::RemoveInfo` (p. 2270), `activemq::commands::ActiveMQMapMessage` (p. 283), `activemq::commands::DataArrayResponse` (p. 1071), `activemq::commands::DataResponse` (p. 1114), `activemq::commands::DiscoveryEvent` (p. 1229), `activemq::commands::ExceptionResponse` (p. 1290), `activemq::commands::PartialCommand` (p. 2078), `activemq::commands::ReplayCommand` (p. 2286), `activemq::commands::BrokerId` (p. 567), `activemq::commands::ControlCommand` (p. 1025), `activemq::commands::IntegerResponse` (p. 1518), `activemq::commands::JournalTrace` (p. 1579), `activemq::commands::Response` (p. 2300), `activemq::commands::FlushCommand` (p. 1383), `activemq::commands::KeepAliveInfo` (p. 1593), `activemq::commands::LastPartialCommand` (p. 1608), `activemq::commands::ShutdownInfo` (p. 2433), `activemq::commands::TransactionId` (p. 2770), `activemq::commands::BaseCommand` (p. 498), `activemq::commands::ActiveMQBlobMessage` (p. 161), `activemq::commands::Command` (p. 870), `activemq::commands::WireFormatInfo` (p. 2899), `activemq::commands::ActiveMQQueue` (p. 325), `activemq::commands::BaseDataStructure` (p. 531), `activemq::commands::ActiveMQTopic` (p. 418), `activemq::commands::ActiveMQTextMessage` (p. 409), `activemq::commands::ActiveMQTempQueue` (p. 391), `activemq::commands::ActiveMQTempTopic` (p. 400), `activemq::commands::ActiveMQObjectMessage` (p. 303), `activemq::commands::ActiveMQMessage` (p. 290), and `activemq::commands::BooleanExpression` (p. 552).

The documentation for this class was generated from the following file:

- `src/main/activemq/commands/DataStructure.h`

6.187 decaf::util::Date Class Reference

Wrapper class around a time value in milliseconds.

```
#include <src/main/decaf/util/Date.h>
```

Inheritance diagram for decaf::util::Date:

Public Member Functions

- **Date** ()
Default constructor - sets time to the current System time, rounded to the nearest millisecond.
- **Date** (long long milliseconds)
Constructs the date with a given time value.
- **Date** (const **Date** &source)
Copy constructor.
- **Date & operator=** (const **Date** &value)
*Assigns the value of one **Date** (p. 1139) object to another.*
- virtual ~**Date** ()
- long long **getTime** () const
Gets the underlying time.
- void **setTime** (long long milliseconds)
Sets the underlying time.
- bool **after** (const **Date** &when) const
Determines whether or not this date falls after the specified time.
- bool **before** (const **Date** &when) const
Determines whether or not this date falls before the specified time.
- std::string **toString** () const
*Converts this **Date** (p. 1139) object to a String of the form:*
- virtual int **compareTo** (const **Date** &value) const
- virtual bool **equals** (const **Date** &value) const
- virtual bool **operator==** (const **Date** &value) const
- virtual bool **operator<** (const **Date** &value) const

6.187.1 Detailed Description

Wrapper class around a time value in milliseconds.

This class is comparable to Java's java.util.Date class.

Since

1.0

6.187.2 Constructor & Destructor Documentation

6.187.2.1 decaf::util::Date::Date ()

Default constructor - sets time to the current System time, rounded to the nearest millisecond.

6.187.2.2 decaf::util::Date::Date (long long *milliseconds*)

Constructs the date with a given time value.

Parameters

<i>milliseconds</i>	The time in milliseconds;
---------------------	---------------------------

6.187.2.3 decaf::util::Date::Date (const Date & *source*)

Copy constructor.

Parameters

<i>source</i>	The Date (p. 1139) instance to copy into this one.
---------------	---

6.187.2.4 virtual decaf::util::Date::~~Date () [virtual]

6.187.3 Member Function Documentation

6.187.3.1 bool decaf::util::Date::after (const Date & *when*) const

Determines whether or not this date falls after the specified time.

Parameters

<i>when</i>	The date to compare
-------------	---------------------

Returns

true if this date falls after when.

6.187.3.2 bool decaf::util::Date::before (const Date & *when*) const

Determines whether or not this date falls before the specified time.

Parameters

<i>when</i>	The date to compare
-------------	---------------------

Returns

true if this date falls before when.

6.187.3.3 `virtual int decaf::util::Date::compareTo (const Date & value) const`
[virtual]

6.187.3.4 `virtual bool decaf::util::Date::equals (const Date & value) const`
[virtual]

6.187.3.5 `long long decaf::util::Date::getTime () const`

Gets the underlying time.

Returns

The underlying time value in milliseconds.

6.187.3.6 `virtual bool decaf::util::Date::operator< (const Date & value) const`
[virtual]

6.187.3.7 `Date& decaf::util::Date::operator= (const Date & value)`

Assigns the value of one **Date** (p. 1139) object to another.

Parameters

<i>value</i>	The value to be copied into this Date (p. 1139) object.
--------------	--

Returns

reference to this object with the newly assigned value.

6.187.3.8 `virtual bool decaf::util::Date::operator== (const Date & value) const`
[virtual]

6.187.3.9 `void decaf::util::Date::setTime (long long milliseconds)`

Sets the underlying time.

Parameters

<i>milliseconds</i>	The underlying time value in milliseconds.
---------------------	--

6.187.3.10 std::string decaf::util::Date::toString () const

Converts this **Date** (p. 1139) object to a String of the form:

dow mon dd hh:mm:ss zzz yyyy

where:

- dow is the day of the week (Sun, Mon, Tue, Wed, Thu, Fri, Sat).
- mon is the month (Jan, Feb, Mar, Apr, May, Jun, Jul, Aug, Sep, Oct, Nov, Dec).
- dd is the day of the month (01 through 31), as two decimal digits.
- hh is the hour of the day (00 through 23), as two decimal digits.
- mm is the minute within the hour (00 through 59), as two decimal digits.
- ss is the second within the minute (00 through 61, as two decimal digits).
- zzz is the time zone (and may reflect daylight saving time). Standard time zone abbreviations include those recognized by the method parse. If time zone information is not available, then zzz is empty - that is, it consists of no characters at all.
- yyyy is the year, as four decimal digits.

Returns

the String representation of the **Date** (p. 1139) object.

The documentation for this class was generated from the following file:

- src/main/decaf/util/**Date.h**

6.188 decaf::internal::DecafRuntime Class Reference

Handles APR initialization and termination.

```
#include <src/main/decaf/internal/DecafRuntime.h>
```

Inheritance diagram for decaf::internal::DecafRuntime:

Public Member Functions

- **DecafRuntime ()**

Initializes the APR Runtime for a library.

- virtual **~DecafRuntime ()**

Terminates the APR Runtime for a library.

- apr_pool_t * **getGlobalPool ()** const

Grants access to the Global APR Pool instance that should be used when creating new threads.

- **decaf::util::concurrent::Mutex * getGlobalLock ()**

Gets a pointer to the Decaf Runtime's Global Lock object, this can be used by Decaf APIs to synchronize around certain actions such as adding or acquiring a resource that must be Thread safe.

6.188.1 Detailed Description

Handles APR initialization and termination.

6.188.2 Constructor & Destructor Documentation

6.188.2.1 **decaf::internal::DecafRuntime::DecafRuntime ()**

Initializes the APR Runtime for a library.

6.188.2.2 **virtual decaf::internal::DecafRuntime::~~DecafRuntime ()** [virtual]

Terminates the APR Runtime for a library.

6.188.3 Member Function Documentation

6.188.3.1 **decaf::util::concurrent::Mutex* decaf::internal::DecafRuntime::getGlobalLock ()**

Gets a pointer to the Decaf Runtime's Global Lock object, this can be used by Decaf APIs to synchronize around certain actions such as adding or acquiring a resource that must be Thread safe.

The pointer returned is owned by the Decaf runtime and should not be deleted or copied by the caller.

Returns

a pointer to the Decaf Runtime's global Lock instance.

6.188.3.2 apr_pool_t* decaf::internal::DecafRuntime::getGlobalPool () const

Grants access to the Global APR Pool instance that should be used when creating new threads.

The documentation for this class was generated from the following file:

- src/main/decaf/internal/**DecafRuntime.h**

6.189 activemq::threads::DedicatedTaskRunner Class Reference

```
#include <src/main/activemq/threads/DedicatedTaskRunner.h>
```

Inheritance diagram for activemq::threads::DedicatedTaskRunner:

Public Member Functions

- **DedicatedTaskRunner** (**Task** *task)
- virtual ~**DedicatedTaskRunner** ()
- virtual void **shutdown** (unsigned int timeout)
Shutdown after a timeout, does not guarantee that the task's iterate method has completed and the thread halted.
- virtual void **shutdown** ()
*Shutdown once the task has finished and the **TaskRunner** (p. 2677)'s thread has exited.*
- virtual void **wakeup** ()
*Signal the **TaskRunner** (p. 2677) to wakeup and execute another iteration cycle on the task, the **Task** (p. 2676) instance will be run until its iterate method has returned false indicating it is done.*

Protected Member Functions

- virtual void **run** ()
Run method - called by the Thread class in the context of the thread.

6.189.1 Constructor & Destructor Documentation

6.189.1.1 activemq::threads::DedicatedTaskRunner::DedicatedTaskRunner (**Task** * task)

6.189.1.2 virtual activemq::threads::DedicatedTaskRunner::~~DedicatedTaskRunner () [virtual]

6.189.2 Member Function Documentation

6.189.2.1 `virtual void activemq::threads::DedicatedTaskRunner::run ()`
`[protected, virtual]`

Run method - called by the Thread class in the context of the thread.

Implements **decaf::lang::Runnable** (p. 2312).

6.189.2.2 `virtual void activemq::threads::DedicatedTaskRunner::shutdown (`
`unsigned int timeout) [virtual]`

Shutdown after a timeout, does not guarantee that the task's iterate method has completed and the thread halted.

Parameters

<i>timeout</i>	- Time in Milliseconds to wait for the task to stop.
----------------	--

Implements **activemq::threads::TaskRunner** (p. 2677).

6.189.2.3 `virtual void activemq::threads::DedicatedTaskRunner::shutdown ()`
`[virtual]`

Shutdown once the task has finished and the **TaskRunner** (p. 2677)'s thread has exited.

Implements **activemq::threads::TaskRunner** (p. 2678).

6.189.2.4 `virtual void activemq::threads::DedicatedTaskRunner::wakeup ()`
`[virtual]`

Signal the **TaskRunner** (p. 2677) to wakeup and execute another iteration cycle on the task, the **Task** (p. 2676) instance will be run until its iterate method has returned false indicating it is done.

Implements **activemq::threads::TaskRunner** (p. 2678).

The documentation for this class was generated from the following file:

- `src/main/activemq/threads/DedicatedTaskRunner.h`

6.190 `activemq::core::policies::DefaultPrefetchPolicy` Class - Reference

```
#include <src/main/activemq/core/policies/DefaultPrefetch-
Policy.h>
```

Inheritance diagram for activemq::core::policies::DefaultPrefetchPolicy:

Public Member Functions

- **DefaultPrefetchPolicy** ()
- virtual **~DefaultPrefetchPolicy** ()
- virtual void **setDurableTopicPrefetch** (int value)
Sets the amount of prefetched messages for a Durable Topic.
- virtual int **getDurableTopicPrefetch** () const
Gets the amount of messages to prefetch for a Durable Topic.
- virtual void **setQueuePrefetch** (int value)
Sets the amount of prefetched messages for a Queue.
- virtual int **getQueuePrefetch** () const
Gets the amount of messages to prefetch for a Queue.
- virtual void **setQueueBrowserPrefetch** (int value)
Sets the amount of prefetched messages for a Queue Browser.
- virtual int **getQueueBrowserPrefetch** () const
Gets the amount of messages to prefetch for a Queue Browser.
- virtual void **setTopicPrefetch** (int value)
Sets the amount of prefetched messages for a Topic.
- virtual int **getTopicPrefetch** () const
Gets the amount of messages to prefetch for a Topic.
- virtual int **getMaxPrefetchLimit** (int value) const
Given a requested value for a new prefetch limit, compare it against some max prefetch value and return either the requested value or the maximum allowable value for prefetch.
- virtual **PrefetchPolicy** * **clone** () const
Clone the Policy and return a new pointer to that clone.

Static Public Attributes

- static int **MAX_PREFETCH_SIZE**
- static int **DEFAULT_DURABLE_TOPIC_PREFETCH**
- static int **DEFAULT_QUEUE_PREFETCH**
- static int **DEFAULT_QUEUE_BROWSER_PREFETCH**
- static int **DEFAULT_TOPIC_PREFETCH**

6.190.1 Constructor & Destructor Documentation

6.190.1.1 `activemq::core::policies::DefaultPrefetchPolicy::DefaultPrefetchPolicy ()`

6.190.1.2 `virtual activemq::core::policies::DefaultPrefetchPolicy::~~DefaultPrefetchPolicy () [virtual]`

6.190.2 Member Function Documentation

6.190.2.1 `virtual PrefetchPolicy* activemq::core::policies::DefaultPrefetchPolicy::clone () const [virtual]`

Clone the Policy and return a new pointer to that clone.

Returns

pointer to a new **PrefetchPolicy** (p. 2110) instance that is a clone of this one.

Implements **activemq::core::PrefetchPolicy** (p. 2111).

6.190.2.2 `virtual int activemq::core::policies::DefaultPrefetchPolicy::getDurableTopicPrefetch () const [inline, virtual]`

Gets the amount of messages to prefetch for a Durable Topic.

Returns

value of the number of messages to prefetch.

Implements **activemq::core::PrefetchPolicy** (p. 2112).

6.190.2.3 `virtual int activemq::core::policies::DefaultPrefetchPolicy::getMaxPrefetchLimit (int value) const [inline, virtual]`

Given a requested value for a new prefetch limit, compare it against some max prefetch value and return either the requested value or the maximum allowable value for prefetch.

Returns

the allowable value for a prefetch limit, either requested or the max.

Implements **activemq::core::PrefetchPolicy** (p. 2112).

6.190.2.4 `virtual int activemq::core::policies::DefaultPrefetchPolicy::getQueueBrowserPrefetch () const [inline, virtual]`

Gets the amount of messages to prefetch for a Queue Browser.

Returns

value of the number of messages to prefetch.

Implements **activemq::core::PrefetchPolicy** (p. 2112).

6.190.2.5 `virtual int activemq::core::policies::DefaultPrefetchPolicy::getQueuePrefetch () const [inline, virtual]`

Gets the amount of messages to prefetch for a Queue.

Returns

value of the number of messages to prefetch.

Implements **activemq::core::PrefetchPolicy** (p. 2113).

6.190.2.6 `virtual int activemq::core::policies::DefaultPrefetchPolicy::getTopicPrefetch () const [inline, virtual]`

Gets the amount of messages to prefetch for a Topic.

Returns

value of the number of messages to prefetch.

Implements **activemq::core::PrefetchPolicy** (p. 2113).

6.190.2.7 `virtual void activemq::core::policies::DefaultPrefetchPolicy::setDurableTopicPrefetch (int value) [inline, virtual]`

Sets the amount of prefetched messages for a Durable Topic.

Parameters

<i>value</i>	The number of messages to prefetch.
--------------	-------------------------------------

Implements **activemq::core::PrefetchPolicy** (p. 2113).

6.190.2.8 `virtual void activemq::core::policies::DefaultPrefetchPolicy::setQueueBrowserPrefetch (int value) [inline, virtual]`

Sets the amount of prefetched messages for a Queue Browser.

Parameters

<i>value</i>	The number of messages to prefetch.
--------------	-------------------------------------

Implements **activemq::core::PrefetchPolicy** (p. 2113).

6.190.2.9 `virtual void activemq::core::policies::DefaultPrefetchPolicy::setQueuePrefetch (int value) [inline, virtual]`

Sets the amount of prefetched messages for a Queue.

Parameters

<i>value</i>	The number of messages to prefetch.
--------------	-------------------------------------

Implements **activemq::core::PrefetchPolicy** (p. 2114).

6.190.2.10 `virtual void activemq::core::policies::DefaultPrefetchPolicy::setTopicPrefetch (int value) [inline, virtual]`

Sets the amount of prefetched messages for a Topic.

Parameters

<i>value</i>	The number of messages to prefetch.
--------------	-------------------------------------

Implements **activemq::core::PrefetchPolicy** (p. 2114).

6.190.3 Field Documentation

6.190.3.1 `int activemq::core::policies::DefaultPrefetchPolicy::DEFAULT_DURABLE_TOPIC_PREFETCH [static]`

6.190.3.2 `int activemq::core::policies::DefaultPrefetchPolicy::DEFAULT_QUEUE_BROWSER_PREFETCH [static]`

6.190.3.3 `int activemq::core::policies::DefaultPrefetchPolicy::DEFAULT_QUEUE_PREFETCH [static]`

6.190.3.4 `int activemq::core::policies::DefaultPrefetchPolicy::DEFAULT_TOPIC_PREFETCH` [static]

6.190.3.5 `int activemq::core::policies::DefaultPrefetchPolicy::MAX_PREFETCH_SIZE` [static]

The documentation for this class was generated from the following file:

- `src/main/activemq/core/policies/DefaultPrefetchPolicy.h`

6.191 `activemq::core::policies::DefaultRedeliveryPolicy` Class - Reference

```
#include <src/main/activemq/core/policies/DefaultRedeliveryPolicy.h>
```

Inheritance diagram for `activemq::core::policies::DefaultRedeliveryPolicy`:

Public Member Functions

- **`DefaultRedeliveryPolicy`** ()
- virtual **`~DefaultRedeliveryPolicy`** ()
- virtual double **`getBackOffMultiplier`** () const
- virtual void **`setBackOffMultiplier`** (double value)
Sets the Back-Off Multiplier for Message Redelivery.
- virtual short **`getCollisionAvoidancePercent`** () const
- virtual void **`setCollisionAvoidancePercent`** (short value)
- virtual long long **`getInitialRedeliveryDelay`** () const
Gets the initial time that redelivery of messages is delayed.
- virtual void **`setInitialRedeliveryDelay`** (long long value)
Sets the initial time that redelivery will be delayed.
- virtual long long **`getRedeliveryDelay`** () const
Gets the time that redelivery of messages is delayed.
- virtual void **`setRedeliveryDelay`** (long long value)
Sets the time that redelivery will be delayed.
- virtual int **`getMaximumRedeliveries`** () const
Gets the Maximum number of allowed redeliveries for a message before it will be discarded by the consumer.
- virtual void **`setMaximumRedeliveries`** (int value)
Sets the Maximum allowable redeliveries for a Message.
- virtual bool **`isUseCollisionAvoidance`** () const
- virtual void **`setUseCollisionAvoidance`** (bool value)

- virtual bool **isUseExponentialBackOff** () const
- virtual void **setUseExponentialBackOff** (bool value)
- virtual long long **getNextRedeliveryDelay** (long long previousDelay)
Given the last used redelivery delay calculate the next value of the delay based on the settings in this Policy instance.
- virtual **RedeliveryPolicy** * **clone** () const
Create a copy of this Policy and return it.

6.191.1 Constructor & Destructor Documentation

6.191.1.1 **activemq::core::policies::DefaultRedeliveryPolicy::DefaultRedeliveryPolicy** ()

6.191.1.2 virtual **activemq::core::policies::DefaultRedeliveryPolicy::~~DefaultRedeliveryPolicy** () [virtual]

6.191.2 Member Function Documentation

6.191.2.1 virtual **RedeliveryPolicy*** **activemq::core::policies::DefaultRedeliveryPolicy::clone** () const [virtual]

Create a copy of this Policy and return it.

Returns

pointer to a new **RedeliveryPolicy** (p. 2250) that is a copy of this one.

Implements **activemq::core::RedeliveryPolicy** (p. 2252).

6.191.2.2 virtual double **activemq::core::policies::DefaultRedeliveryPolicy::getBackOffMultiplier** () const [inline, virtual]

Returns

The value of the Back-Off Multiplier for Message Redelivery.

Implements **activemq::core::RedeliveryPolicy** (p. 2252).

6.191.2.3 virtual short **activemq::core::policies::DefaultRedeliveryPolicy::getCollisionAvoidancePercent** () const [virtual]

Returns

the currently set Collision Avoidance percentage.

Implements **activemq::core::RedeliveryPolicy** (p. 2253).

6.191.2.4 `virtual long long activemq::core::policies::DefaultRedelivery-Policy::getInitialRedeliveryDelay () const [inline, virtual]`

Gets the initial time that redelivery of messages is delayed.

Returns

the time in milliseconds that redelivery is delayed initially.

Implements **activemq::core::RedeliveryPolicy** (p. 2253).

6.191.2.5 `virtual int activemq::core::policies::DefaultRedelivery-Policy::getMaximumRedeliveries () const [inline, virtual]`

Gets the Maximum number of allowed redeliveries for a message before it will be discarded by the consumer.

Returns

maximum allowed redeliveries for a message.

Implements **activemq::core::RedeliveryPolicy** (p. 2253).

6.191.2.6 `virtual long long activemq::core::policies::DefaultRedelivery-Policy::getNextRedeliveryDelay (long long previousDelay) [virtual]`

Given the last used redelivery delay calculate the next value of the delay based on the settings in this Policy instance.

Parameters

<i>previous-Delay</i>	The last delay that was used between message redeliveries.
-----------------------	--

Returns

the new delay to use before attempting another redelivery.

Implements **activemq::core::RedeliveryPolicy** (p. 2253).

6.191.2.7 `virtual long long activemq::core::policies::DefaultRedelivery-Policy::getRedeliveryDelay () const [inline, virtual]`

Gets the time that redelivery of messages is delayed.

Returns

the time in milliseconds that redelivery is delayed.

Implements **activemq::core::RedeliveryPolicy** (p. 2254).

6.191.2.8 `virtual bool activemq::core::policies::DefaultRedeliveryPolicy::isUseCollisionAvoidance () const [inline, virtual]`

Returns

whether or not collision avoidance is enabled for this Policy.

Implements **activemq::core::RedeliveryPolicy** (p. 2254).

6.191.2.9 `virtual bool activemq::core::policies::DefaultRedeliveryPolicy::isUseExponentialBackOff () const [inline, virtual]`

Returns

whether or not the exponential back off option is enabled.

Implements **activemq::core::RedeliveryPolicy** (p. 2254).

6.191.2.10 `virtual void activemq::core::policies::DefaultRedeliveryPolicy::setBackOffMultiplier (double value) [inline, virtual]`

Sets the Back-Off Multiplier for Message Redelivery.

Parameters

<i>value</i>	The new value for the back-off multiplier.
--------------	--

Implements **activemq::core::RedeliveryPolicy** (p. 2254).

6.191.2.11 `virtual void activemq::core::policies::DefaultRedeliveryPolicy::setCollisionAvoidancePercent (short value) [virtual]`

Parameters

<i>value</i>	The collision avoidance percentage setting.
--------------	---

Implements **activemq::core::RedeliveryPolicy** (p. 2255).

6.191.2.12 `virtual void activemq::core::policies::DefaultRedeliveryPolicy::setInitialRedeliveryDelay (long long value) [inline, virtual]`

Sets the initial time that redelivery will be delayed.

Parameters

<i>value</i>	Time in Milliseconds to wait before starting redelivery.
--------------	--

Implements **activemq::core::RedeliveryPolicy** (p. 2255).

6.191.2.13 `virtual void activemq::core::policies::DefaultRedeliveryPolicy::setMaximumRedeliveries (int maximumRedeliveries) [inline, virtual]`

Sets the Maximum allowable redeliveries for a Message.

Parameters

<i>maximum-Redeliveries</i>	The maximum number of times that a message will be redelivered.
-----------------------------	---

Implements **activemq::core::RedeliveryPolicy** (p. 2255).

6.191.2.14 `virtual void activemq::core::policies::DefaultRedeliveryPolicy::setRedeliveryDelay (long long value) [inline, virtual]`

Sets the time that redelivery will be delayed.

Parameters

<i>value</i>	Time in Milliseconds to wait before the next redelivery.
--------------	--

Implements **activemq::core::RedeliveryPolicy** (p. 2255).

6.191.2.15 `virtual void activemq::core::policies::DefaultRedeliveryPolicy::setUseCollisionAvoidance (bool value) [inline, virtual]`

Parameters

<i>value</i>	Enable or Disable collision avoidance for this Policy.
--------------	--

Implements **activemq::core::RedeliveryPolicy** (p. 2255).

6.191.2.16 `virtual void activemq::core::policies::DefaultRedeliveryPolicy::setUseExponentialBackOff (bool value) [inline, virtual]`

Parameters

<i>value</i>	Enable or Disable the exponential back off multiplier option.
--------------	---

Implements `activemq::core::RedeliveryPolicy` (p. 2256).

The documentation for this class was generated from the following file:

- `src/main/activemq/core/policies/DefaultRedeliveryPolicy.h`

6.192 `decaf::internal::net::DefaultServerSocketFactory` Class - Reference

Default implementation of the Decaf ServerSocketFactory, creates ServerSocket objects with supplied options.

```
#include <src/main/decaf/internal/net/DefaultServerSocketFactory.h>
```

Inheritance diagram for `decaf::internal::net::DefaultServerSocketFactory`:

Public Member Functions

- `DefaultServerSocketFactory ()`
- `virtual ~DefaultServerSocketFactory ()`
- `virtual decaf::net::ServerSocket * createServerSocket ()`

Create a new **ServerSocket** (p. 2342) that is unbound.

The **ServerSocket** (p. 2342) will have been configured with the defaults from the factory.

Returns

new ServerSocket (p. 2342) pointer that is owned by the caller.

Exceptions

IOException	if the ServerSocket (p. 2342) cannot be created for some reason.
-------------	---

- `virtual decaf::net::ServerSocket * createServerSocket (int port)`

Create a new **ServerSocket** (p. 2342) that is bound to the given port.

The **ServerSocket** (p. 2342) will have been configured with the defaults from the factory.

Parameters

port	The port to bind the ServerSocket (p. 2342) to.
------	--

Returns

new **ServerSocket** (p. 2342) pointer that is owned by the caller.

Exceptions

IOException	if the ServerSocket (p. 2342) cannot be created for some reason.
-------------	---

- virtual **decaf::net::ServerSocket * createServerSocket** (int port, int backlog)

Create a new **ServerSocket** (p. 2342) that is bound to the given port.

The **ServerSocket** (p. 2342) will have been configured with the defaults from the factory. The **ServerSocket** (p. 2342) will use the specified connection backlog setting.

Parameters

port	The port to bind the ServerSocket (p. 2342) to.
backlog	The number of pending connect request the ServerSocket (p. 2342) can queue.

Returns

new **ServerSocket** (p. 2342) pointer that is owned by the caller.

Exceptions

IOException	if the ServerSocket (p. 2342) cannot be created for some reason.
-------------	---

- virtual **decaf::net::ServerSocket * createServerSocket** (int port, int backlog, const **decaf::net::InetAddress *address**)

Create a new **ServerSocket** (p. 2342) that is bound to the given port.

The **ServerSocket** (p. 2342) will have been configured with the defaults from the factory. The **ServerSocket** (p. 2342) will bind to the specified interface on the local host, and accept connections only on that interface. If the address parameter is NULL than the **ServerSocket** (p. 2342) will listen on all interfaces.

Parameters

port	The port to bind the ServerSocket (p. 2342) to.
backlog	The number of pending connect request the ServerSocket (p. 2342) can queue.
address	The address of the interface on the local machine to bind to.

Returns

new **ServerSocket** (p. 2342) pointer that is owned by the caller.

Exceptions

IOException	if the ServerSocket (p. 2342) cannot be created for some reason.
-------------	---

6.192.1 Detailed Description

Default implementation of the Decaf ServerSocketFactory, creates ServerSocket objects with supplied options.

Since

1.0

6.192.2 Constructor & Destructor Documentation

6.192.2.1 `decaf::internal::net::DefaultServerSocketFactory::DefaultServerSocketFactory ()`

6.192.2.2 `virtual decaf::internal::net::DefaultServerSocketFactory::~~DefaultServerSocketFactory () [virtual]`

6.192.3 Member Function Documentation

6.192.3.1 `virtual decaf::net::ServerSocket* decaf::internal::net::DefaultServerSocketFactory::createServerSocket () [virtual]`

Create a new **ServerSocket** (p. 2342) that is unbound.

The **ServerSocket** (p. 2342) will have been configured with the defaults from the factory.

Returns

new **ServerSocket** (p. 2342) pointer that is owned by the caller.

Exceptions

<i>IOException</i>	if the ServerSocket (p. 2342) cannot be created for some reason.
--------------------	---

Reimplemented from **decaf::net::ServerSocketFactory** (p. 2353).

6.192.3.2 `virtual decaf::net::ServerSocket* decaf::internal::net::DefaultServerSocketFactory::createServerSocket (int port) [virtual]`

Create a new **ServerSocket** (p. 2342) that is bound to the given port.

The **ServerSocket** (p. 2342) will have been configured with the defaults from the factory.

Parameters

<i>port</i>	The port to bind the ServerSocket (p. 2342) to.
-------------	--

Returns

new **ServerSocket** (p. 2342) pointer that is owned by the caller.

6.192 decaf::internal::net::DefaultServerSocketFactory Class Reference 1161

Exceptions

<i>IOException</i>	if the ServerSocket (p. 2342) cannot be created for some reason.
--------------------	---

Implements **decaf::net::ServerSocketFactory** (p. 2353).

6.192.3.3 virtual **decaf::net::ServerSocket*** **decaf::internal::net::DefaultServerSocketFactory::createServerSocket** (int *port*, int *backlog*)
[virtual]

Create a new **ServerSocket** (p. 2342) that is bound to the given port.

The **ServerSocket** (p. 2342) will have been configured with the defaults from the factory.
The **ServerSocket** (p. 2342) will use the specified connection backlog setting.

Parameters

<i>port</i>	The port to bind the ServerSocket (p. 2342) to.
<i>backlog</i>	The number of pending connect request the ServerSocket (p. 2342) can queue.

Returns

new **ServerSocket** (p. 2342) pointer that is owned by the caller.

Exceptions

<i>IOException</i>	if the ServerSocket (p. 2342) cannot be created for some reason.
--------------------	---

Implements **decaf::net::ServerSocketFactory** (p. 2354).

6.192.3.4 virtual **decaf::net::ServerSocket*** **decaf::internal::net::DefaultServerSocketFactory::createServerSocket** (int *port*, int *backlog*, const **decaf::net::InetAddress*** *address*) [virtual]

Create a new **ServerSocket** (p. 2342) that is bound to the given port.

The **ServerSocket** (p. 2342) will have been configured with the defaults from the factory.
The **ServerSocket** (p. 2342) will bind to the specified interface on the local host, and accept connections only on that interface. If the address parameter is NULL than the **ServerSocket** (p. 2342) will listen on all interfaces.

Parameters

<i>port</i>	The port to bind the ServerSocket (p. 2342) to.
<i>backlog</i>	The number of pending connect request the ServerSocket (p. 2342) can queue.
<i>address</i>	The address of the interface on the local machine to bind to.

Returns

new **ServerSocket** (p. 2342) pointer that is owned by the caller.

Exceptions

<i>IOException</i>	if the ServerSocket (p. 2342) cannot be created for some reason.
--------------------	---

Implements **decaf::net::ServerSocketFactory** (p. 2354).

The documentation for this class was generated from the following file:

- src/main/decaf/internal/net/**DefaultServerSocketFactory.h**

6.193 decaf::internal::net::DefaultSocketFactory Class Reference

SocketFactory implementation that is used to create Sockets.

```
#include <src/main/decaf/internal/net/DefaultSocketFactory.h>
```

Inheritance diagram for decaf::internal::net::DefaultSocketFactory:

Public Member Functions

- **DefaultSocketFactory** ()
- virtual **~DefaultSocketFactory** ()
- virtual **decaf::net::Socket * createSocket** ()

*Creates an unconnected **Socket** (p. 2452) object.*

Returns

*a new **Socket** (p. 2452) object, caller must free this object when done.*

Exceptions

<i>IOException</i>	if the Socket (p. 2452) cannot be created.
--------------------	---

- virtual **decaf::net::Socket * createSocket** (const **decaf::net::InetAddress** *host, int port)

*Creates a new **Socket** (p. 2452) object and connects it to the specified remote host and port using the configuration of this **SocketFactory** (p. 2473).*

Parameters

host	<i>The host to connect the socket to.</i>
port	<i>The port on the remote host to connect to.</i>

Returns

a new **Socket** (p. 2452) object, caller must free this object when done.

Exceptions

IOException	if an I/O error occurs while creating the Socket (p. 2452) object.
UnknownHostException (p. 2816)	if the host name is not known.

- virtual **decaf::net::Socket * createSocket** (const **decaf::net::InetAddress** *host, int port, const **decaf::net::InetAddress** *ifAddress, int localPort)

Creates a new **Socket** (p. 2452) object and connects it to the specified remote host and port using the configuration of this **SocketFactory** (p. 2473).

The **Socket** (p. 2452) will be bound to the specified local address and port.

Parameters

host	The host to connect the socket to.
port	The port on the remote host to connect to.
ifAddress	The address on the local machine to bind the Socket (p. 2452) to.
localPort	The local port to bind the Socket (p. 2452) to.

Returns

a new **Socket** (p. 2452) object, caller must free this object when done.

Exceptions

IOException	if an I/O error occurs while creating the Socket (p. 2452) object.
UnknownHostException (p. 2816)	if the host name is not known.

- virtual **decaf::net::Socket * createSocket** (const std::string &name, int port)

Creates a new **Socket** (p. 2452) object and connects it to the specified remote host and port using the configuration of this **SocketFactory** (p. 2473).

Parameters

host	The host name or IP address to connect the socket to.
port	The port on the remote host to connect to.

Returns

a new **Socket** (p. 2452) object, caller must free this object when done.

Exceptions

IOException	if an I/O error occurs while creating the Socket (p. 2452) object.
UnknownHostException (p. 2816)	if the host name is not known.

- virtual **decaf::net::Socket * createSocket** (const std::string &name, int port, const **decaf::net::InetAddress** *ifAddress, int localPort)

Creates a new **Socket** (p. 2452) object and connects it to the specified remote host and port using the configuration of this **SocketFactory** (p. 2473).

Parameters

host	The host name or IP address to connect the socket to.
port	The port on the remote host to connect to.
ifAddress	The address on the local machine to bind the Socket (p. 2452) to.
localPort	The local port to bind the Socket (p. 2452) to.

Returns

a new **Socket** (p. 2452) object, caller must free this object when done.

Exceptions

IOException	if an I/O error occurs while creating the Socket (p. 2452) object.
UnknownHostException (p. 2816)	if the host name is not known.

6.193.1 Detailed Description

SocketFactory implementation that is used to create Sockets.

Since

1.0

6.193.2 Constructor & Destructor Documentation

6.193.2.1 `decaf::internal::net::DefaultSocketFactory::DefaultSocketFactory ()`

6.193.2.2 `virtual decaf::internal::net::DefaultSocketFactory::~~DefaultSocketFactory () [virtual]`

6.193.3 Member Function Documentation

6.193.3.1 `virtual decaf::net::Socket* decaf::internal::net::DefaultSocketFactory::createSocket () [virtual]`

Creates an unconnected **Socket** (p. 2452) object.

Returns

a new **Socket** (p. 2452) object, caller must free this object when done.

Exceptions

IOException	if the Socket (p. 2452) cannot be created.
-------------	---

Reimplemented from `decaf::net::SocketFactory` (p. 2475).

6.193.3.2 virtual **decaf::net::Socket*** **decaf::internal::net::DefaultSocketFactory::createSocket** (const **decaf::net::InetAddress** * *host*, int *port*)
[virtual]

Creates a new **Socket** (p. 2452) object and connects it to the specified remote host and port using the configuration of this **SocketFactory** (p. 2473).

Parameters

<i>host</i>	The host to connect the socket to.
<i>port</i>	The port on the remote host to connect to.

Returns

a new **Socket** (p. 2452) object, caller must free this object when done.

Exceptions

<i>IOException</i>	if an I/O error occurs while creating the Socket (p. 2452) object.
UnknownHostException (p. 2816)	if the host name is not known.

Implements **decaf::net::SocketFactory** (p. 2475).

6.193.3.3 virtual **decaf::net::Socket*** **decaf::internal::net::DefaultSocketFactory::createSocket** (const **decaf::net::InetAddress** * *host*, int *port*, const **decaf::net::InetAddress** * *ifAddress*, int *localPort*) [virtual]

Creates a new **Socket** (p. 2452) object and connects it to the specified remote host and port using the configuration of this **SocketFactory** (p. 2473).

The **Socket** (p. 2452) will be bound to the specified local address and port.

Parameters

<i>host</i>	The host to connect the socket to.
<i>port</i>	The port on the remote host to connect to.
<i>ifAddress</i>	The address on the local machine to bind the Socket (p. 2452) to.
<i>localPort</i>	The local port to bind the Socket (p. 2452) to.

Returns

a new **Socket** (p. 2452) object, caller must free this object when done.

Exceptions

<i>IOException</i>	if an I/O error occurs while creating the Socket (p. 2452) object.
--------------------	---

UnknownHostException (p. 2816)	if the host name is not known.
--	--------------------------------

Implements **decaf::net::SocketFactory** (p. 2476).

6.193.3.4 **virtual decaf::net::Socket* decaf::internal::net::DefaultSocketFactory::createSocket (const std::string & name, int port)**
[virtual]

Creates a new **Socket** (p. 2452) object and connects it to the specified remote host and port using the configuration of this **SocketFactory** (p. 2473).

Parameters

<i>host</i>	The host name or IP address to connect the socket to.
<i>port</i>	The port on the remote host to connect to.

Returns

a new **Socket** (p. 2452) object, caller must free this object when done.

Exceptions

<i>IOException</i>	if an I/O error occurs while creating the Socket (p. 2452) object.
UnknownHostException (p. 2816)	if the host name is not known.

Implements **decaf::net::SocketFactory** (p. 2476).

6.193.3.5 **virtual decaf::net::Socket* decaf::internal::net::DefaultSocketFactory::createSocket (const std::string & name, int port, const decaf::net::InetAddress * ifAddress, int localPort)** [virtual]

Creates a new **Socket** (p. 2452) object and connects it to the specified remote host and port using the configuration of this **SocketFactory** (p. 2473).

Parameters

<i>host</i>	The host name or IP address to connect the socket to.
<i>port</i>	The port on the remote host to connect to.
<i>ifAddress</i>	The address on the local machine to bind the Socket (p. 2452) to.
<i>localPort</i>	The local port to bind the Socket (p. 2452) to.

Returns

a new **Socket** (p. 2452) object, caller must free this object when done.

Exceptions

<i>IOException</i>	if an I/O error occurs while creating the Socket (p. 2452) object.
<i>UnknownHostException</i> (p. 2816)	if the host name is not known.

Implements **decaf::net::SocketFactory** (p. 2477).

The documentation for this class was generated from the following file:

- `src/main/decaf/internal/net/DefaultSocketFactory.h`

6.194 `decaf::internal::net::ssl::DefaultSSLContext` Class Reference

Default SSLContext manager for the Decaf library.

```
#include <src/main/decaf/internal/net/ssl/DefaultSSLContext.h>
```

Public Member Functions

- virtual `~DefaultSSLContext ()`

Static Public Member Functions

- static `decaf::net::ssl::SSLContext * getContext ()`

Protected Member Functions

- `DefaultSSLContext ()`

6.194.1 Detailed Description

Default SSLContext manager for the Decaf library.

If the user doesn't supply or specify the SSLContext that they wish to use then we load the Decaf library's default SSLContext using whatever SSL provider is enabled an preferred.

Since

1.0

6.194.2 Constructor & Destructor Documentation

6.194.2.1 `decaf::internal::net::ssl::DefaultSSLContext::DefaultSSLContext ()`
[protected]

6.194.2.2 `virtual decaf::internal::net::ssl::DefaultSSLContext::~~DefaultSSLContext ()` [virtual]

6.194.3 Member Function Documentation

6.194.3.1 `static decaf::net::ssl::SSLContext* decaf::internal::net::ssl::DefaultSSLContext::getContext ()`
[static]

The documentation for this class was generated from the following file:

- `src/main/decaf/internal/net/ssl/DefaultSSLContext.h`

6.195 decaf::internal::net::ssl::DefaultSSLServerSocketFactory - Class Reference

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

```
#include <src/main/decaf/internal/net/ssl/DefaultSSL-  
ServerSocketFactory.h>
```

Inheritance diagram for `decaf::internal::net::ssl::DefaultSSLServerSocketFactory`:

Public Member Functions

- **DefaultSSLServerSocketFactory** (const std::string &errorMessage)
- virtual **~DefaultSSLServerSocketFactory** ()
- virtual **decaf::net::ServerSocket * createServerSocket** ()

Create a new **ServerSocket** (p. 2342) that is unbound.

The **ServerSocket** (p. 2342) will have been configured with the defaults from the factory.

Returns

new **ServerSocket** (p. 2342) pointer that is owned by the caller.

Exceptions

IOException	if the ServerSocket (p. 2342) cannot be created for some reason.
-------------	---

- virtual **decaf::net::ServerSocket * createServerSocket** (int port)

6.195 `decaf::internal::net::ssl::DefaultSSLServerSocketFactory` Class Reference

Create a new **ServerSocket** (p. 2342) that is bound to the given port.

The **ServerSocket** (p. 2342) will have been configured with the defaults from the factory.

Parameters

port	The port to bind the ServerSocket (p. 2342) to.
------	--

Returns

new **ServerSocket** (p. 2342) pointer that is owned by the caller.

Exceptions

IOException	if the ServerSocket (p. 2342) cannot be created for some reason.
-------------	---

- virtual **decaf::net::ServerSocket * createServerSocket** (int port, int backlog)

Create a new **ServerSocket** (p. 2342) that is bound to the given port.

The **ServerSocket** (p. 2342) will have been configured with the defaults from the factory. The **ServerSocket** (p. 2342) will use the specified connection backlog setting.

Parameters

port	The port to bind the ServerSocket (p. 2342) to.
backlog	The number of pending connect request the ServerSocket (p. 2342) can queue.

Returns

new **ServerSocket** (p. 2342) pointer that is owned by the caller.

Exceptions

IOException	if the ServerSocket (p. 2342) cannot be created for some reason.
-------------	---

- virtual **decaf::net::ServerSocket * createServerSocket** (int port, int backlog, const **decaf::net::InetAddress *address**)

Create a new **ServerSocket** (p. 2342) that is bound to the given port.

The **ServerSocket** (p. 2342) will have been configured with the defaults from the factory. The **ServerSocket** (p. 2342) will bind to the specified interface on the local host, and accept connections only on that interface. If the address parameter is NULL than the **ServerSocket** (p. 2342) will listen on all interfaces.

Parameters

port	The port to bind the ServerSocket (p. 2342) to.
backlog	The number of pending connect request the ServerSocket (p. 2342) can queue.
address	The address of the interface on the local machine to bind to.

Returns

new **ServerSocket** (p. 2342) pointer that is owned by the caller.

Exceptions

IOException	if the ServerSocket (p. 2342) cannot be created for some reason.
-------------	---

- virtual **std::vector< std::string > getDefaultCipherSuites** ()

Returns the list of cipher suites which are enabled by default.

Unless a different list is enabled, handshaking on an SSL connection will use one of these cipher suites. The minimum quality of service for these defaults requires confidentiality protection and server authentication (that is, no anonymous cipher suites).

Returns

an STL vector containing the list of cipher suites enabled by default.

See also

getSupportedCipherSuites() (p. 2512)

- virtual std::vector< std::string > **getSupportedCipherSuites** ()

Returns the names of the cipher suites which could be enabled for use on an SSL connection.

Normally, only a subset of these will actually be enabled by default, since this list may include cipher suites which do not meet quality of service requirements for those defaults. Such cipher suites are useful in specialized applications.

Returns

an STL vector containing the list of supported cipher suites.

See also

getDefaultCipherSuites() (p. 2512)

6.195.1 Detailed Description

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

Since

1.0

6.195.2 Constructor & Destructor Documentation

6.195.2.1 **decaf::internal::net::ssl::DefaultSSLServerSocketFactory::DefaultSSLServerSocketFactory (const std::string & errorMessage)**

6.195.2.2 **virtual decaf::internal::net::ssl::DefaultSSLServerSocketFactory::~DefaultSSLServerSocketFactory ()**
[virtual]

6.195.3 Member Function Documentation

6.195.3.1 **virtual decaf::net::ServerSocket* decaf::internal::net::ssl::DefaultSSLServerSocketFactory::createServerSocket ()**
[virtual]

Create a new **ServerSocket** (p. 2342) that is unbound.

The **ServerSocket** (p. 2342) will have been configured with the defaults from the factory.

6.195 `decaf::internal::net::ssl::DefaultSSLServerSocketFactory` Class Reference

Returns

new **ServerSocket** (p. 2342) pointer that is owned by the caller.

Exceptions

<i>IOException</i>	if the ServerSocket (p. 2342) cannot be created for some reason.
--------------------	---

Reimplemented from **decaf::net::ServerSocketFactory** (p. 2353).

6.195.3.2 `virtual decaf::net::ServerSocket* decaf::internal::net::ssl::DefaultSSLServerSocketFactory::createServerSocket (int port)`
[virtual]

Create a new **ServerSocket** (p. 2342) that is bound to the given port.

The **ServerSocket** (p. 2342) will have been configured with the defaults from the factory.

Parameters

<i>port</i>	The port to bind the ServerSocket (p. 2342) to.
-------------	--

Returns

new **ServerSocket** (p. 2342) pointer that is owned by the caller.

Exceptions

<i>IOException</i>	if the ServerSocket (p. 2342) cannot be created for some reason.
--------------------	---

Implements **decaf::net::ServerSocketFactory** (p. 2353).

6.195.3.3 `virtual decaf::net::ServerSocket* decaf::internal::net::ssl::DefaultSSLServerSocketFactory::createServerSocket (int port, int backlog)`
[virtual]

Create a new **ServerSocket** (p. 2342) that is bound to the given port.

The **ServerSocket** (p. 2342) will have been configured with the defaults from the factory.

The **ServerSocket** (p. 2342) will use the specified connection backlog setting.

Parameters

<i>port</i>	The port to bind the ServerSocket (p. 2342) to.
<i>backlog</i>	The number of pending connect request the ServerSocket (p. 2342) can queue.

Returns

new **ServerSocket** (p. 2342) pointer that is owned by the caller.

Exceptions

<i>IOException</i>	if the ServerSocket (p. 2342) cannot be created for some reason.
--------------------	---

Implements **decaf::net::ServerSocketFactory** (p. 2354).

6.195.3.4 `virtual decaf::net::ServerSocket* decaf::internal::net::ssl::DefaultSSL-
ServerSocketFactory::createServerSocket (int port, int backlog, const
decaf::net::InetAddress * address) [virtual]`

Create a new **ServerSocket** (p. 2342) that is bound to the given port.

The **ServerSocket** (p. 2342) will have been configured with the defaults from the factory. The **ServerSocket** (p. 2342) will bind to the specified interface on the local host, and accept connections only on that interface. If the address parameter is NULL than the **ServerSocket** (p. 2342) will listen on all interfaces.

Parameters

<i>port</i>	The port to bind the ServerSocket (p. 2342) to.
<i>backlog</i>	The number of pending connect request the ServerSocket (p. 2342) can queue.
<i>address</i>	The address of the interface on the local machine to bind to.

Returns

new **ServerSocket** (p. 2342) pointer that is owned by the caller.

Exceptions

<i>IOException</i>	if the ServerSocket (p. 2342) cannot be created for some reason.
--------------------	---

Implements **decaf::net::ServerSocketFactory** (p. 2354).

6.195.3.5 `virtual std::vector<std::string> decaf::internal::net::ssl::Default-
SSLServerSocketFactory::getDefaultCipherSuites ()
[virtual]`

Returns the list of cipher suites which are enabled by default.

Unless a different list is enabled, handshaking on an SSL connection will use one of these cipher suites. The minimum quality of service for these defaults requires confidentiality protection and server authentication (that is, no anonymous cipher suites).

6.196 decaf::internal::net::ssl::DefaultSSLSocketFactory Class Reference 1173

Returns

an STL vector containing the list of cipher suites enabled by default.

See also

getSupportedCipherSuites() (p. 2512)

Implements **decaf::net::ssl::SSLServerSocketFactory** (p. 2512).

```
6.195.3.6 virtual std::vector<std::string> decaf::internal::net::ssl::Default-  
SSLServerSocketFactory::getSupportedCipherSuites ( )  
[virtual]
```

Returns the names of the cipher suites which could be enabled for use on an SSL connection.

Normally, only a subset of these will actually be enabled by default, since this list may include cipher suites which do not meet quality of service requirements for those defaults. Such cipher suites are useful in specialized applications.

Returns

an STL vector containing the list of supported cipher suites.

See also

getDefaultCipherSuites() (p. 2512)

Implements **decaf::net::ssl::SSLServerSocketFactory** (p. 2512).

The documentation for this class was generated from the following file:

- src/main/decaf/internal/net/ssl/**DefaultSSLServerSocketFactory.h**

6.196 decaf::internal::net::ssl::DefaultSSLSocketFactory Class - Reference

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

```
#include <src/main/decaf/internal/net/ssl/DefaultSSL-  
SocketFactory.h>
```

Inheritance diagram for decaf::internal::net::ssl::DefaultSSLSocketFactory:

Public Member Functions

- **DefaultSSLSocketFactory** (const std::string &errorMessage)
- virtual ~**DefaultSSLSocketFactory** ()
- virtual **decaf::net::Socket** * **createSocket** ()

*Creates an unconnected **Socket** (p. 2452) object.*

Returns

*a new **Socket** (p. 2452) object, caller must free this object when done.*

Exceptions

IOException	<i>if the Socket (p. 2452) cannot be created.</i>
-------------	--

- virtual **decaf::net::Socket** * **createSocket** (const **decaf::net::InetAddress** *host, int port)

*Creates a new **Socket** (p. 2452) object and connects it to the specified remote host and port using the configuration of this **SocketFactory** (p. 2473).*

Parameters

host	<i>The host to connect the socket to.</i>
port	<i>The port on the remote host to connect to.</i>

Returns

*a new **Socket** (p. 2452) object, caller must free this object when done.*

Exceptions

IOException	<i>if an I/O error occurs while creating the Socket (p. 2452) object.</i>
UnknownHostException (p. 2816)	<i>if the host name is not known.</i>

- virtual **decaf::net::Socket** * **createSocket** (const **decaf::net::InetAddress** *host, int port, const **decaf::net::InetAddress** *ifAddress, int localPort)

*Creates a new **Socket** (p. 2452) object and connects it to the specified remote host and port using the configuration of this **SocketFactory** (p. 2473).*

*The **Socket** (p. 2452) will be bound to the specified local address and port.*

Parameters

host	<i>The host to connect the socket to.</i>
port	<i>The port on the remote host to connect to.</i>
ifAddress	<i>The address on the local machine to bind the Socket (p. 2452) to.</i>
localPort	<i>The local port to bind the Socket (p. 2452) to.</i>

Returns

*a new **Socket** (p. 2452) object, caller must free this object when done.*

Exceptions

IOException	<i>if an I/O error occurs while creating the Socket (p. 2452) object.</i>
UnknownHostException (p. 2816)	<i>if the host name is not known.</i>

- virtual **decaf::net::Socket** * **createSocket** (const std::string &name, int port)

*Creates a new **Socket** (p. 2452) object and connects it to the specified remote host*

and port using the configuration of this **SocketFactory** (p. 2473).

Parameters

host	The host name or IP address to connect the socket to.
port	The port on the remote host to connect to.

Returns

a new **Socket** (p. 2452) object, caller must free this object when done.

Exceptions

IOException	if an I/O error occurs while creating the Socket (p. 2452) object.
UnknownHostException (p. 2816)	if the host name is not known.

- virtual **decaf::net::Socket * createSocket** (const std::string &name, int port, const **decaf::net::InetAddress** *ifAddress, int localPort)

Creates a new **Socket** (p. 2452) object and connects it to the specified remote host and port using the configuration of this **SocketFactory** (p. 2473).

Parameters

host	The host name or IP address to connect the socket to.
port	The port on the remote host to connect to.
ifAddress	The address on the local machine to bind the Socket (p. 2452) to.
localPort	The local port to bind the Socket (p. 2452) to.

Returns

a new **Socket** (p. 2452) object, caller must free this object when done.

Exceptions

IOException	if an I/O error occurs while creating the Socket (p. 2452) object.
UnknownHostException (p. 2816)	if the host name is not known.

- virtual std::vector< std::string > **getDefaultCipherSuites** ()

Returns the list of cipher suites which are enabled by default.
Unless a different list is enabled, handshaking on an SSL connection will use one of these cipher suites. The minimum quality of service for these defaults requires confidentiality protection and server authentication (that is, no anonymous cipher suites).

Returns

an STL vector containing the list of cipher suites enabled by default.

See also

getSupportedCipherSuites() (p. 2524)

- virtual std::vector< std::string > **getSupportedCipherSuites** ()

Returns the names of the cipher suites which could be enabled for use on an SSL connection.

Normally, only a subset of these will actually be enabled by default, since this list may include cipher suites which do not meet quality of service requirements for those defaults. Such cipher suites are useful in specialized applications.

Returns

an STL vector containing the list of supported cipher suites.

See also

getDefaultCipherSuites() (p. 2524)

- virtual **decaf::net::Socket** * **createSocket** (**decaf::net::Socket** *socket, std::string host, int port, bool autoClose)

Returns a socket layered over an existing socket connected to the named host, at the given port.

This constructor can be used when tunneling SSL through a proxy or when negotiating the use of SSL over an existing socket. The host and port refer to the logical peer destination. This socket is configured using the socket options established for this factory.

Parameters

socket	The existing socket to layer over.
host	The server host the original Socket (p. 2452) is connected to.
port	The server port the original Socket (p. 2452) is connected to.
autoClose	Should the layered over Socket (p. 2452) be closed when the topmost socket is closed.

Returns

a new **Socket** (p. 2452) instance that wraps the given **Socket** (p. 2452).

Exceptions

IOException	if an I/O exception occurs while performing this operation.
UnknownHostException (p. 2816)	if the host is unknown.

6.196.1 Detailed Description

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

Since

1.0

6.196.2 Constructor & Destructor Documentation

6.196.2.1 **decaf::internal::net::ssl::DefaultSSLSocketFactory::DefaultSSLSocketFactory** (const std::string & *errorMessage*)

6.196.2.2 virtual **decaf::internal::net::ssl::DefaultSSLSocketFactory::~DefaultSSLSocketFactory** () [virtual]

6.196.3 Member Function Documentation

6.196 decaf::internal::net::ssl::DefaultSSLSocketFactory Class Reference 1177

6.196.3.1 virtual **decaf::net::Socket*** **decaf::internal::net::ssl::DefaultSSLSocketFactory::createSocket** ()
[virtual]

Creates an unconnected **Socket** (p. 2452) object.

Returns

a new **Socket** (p. 2452) object, caller must free this object when done.

Exceptions

<i>IOException</i>	if the Socket (p. 2452) cannot be created.
--------------------	---

Reimplemented from **decaf::net::SocketFactory** (p. 2475).

6.196.3.2 virtual **decaf::net::Socket*** **decaf::internal::net::ssl::DefaultSSLSocketFactory::createSocket** (const **decaf::net::InetAddress** * *host*, int *port*)
[virtual]

Creates a new **Socket** (p. 2452) object and connects it to the specified remote host and port using the configuration of this **SocketFactory** (p. 2473).

Parameters

<i>host</i>	The host to connect the socket to.
<i>port</i>	The port on the remote host to connect to.

Returns

a new **Socket** (p. 2452) object, caller must free this object when done.

Exceptions

<i>IOException</i>	if an I/O error occurs while creating the Socket (p. 2452) object.
UnknownHostException (p. 2816)	if the host name is not known.

Implements **decaf::net::SocketFactory** (p. 2475).

6.196.3.3 virtual **decaf::net::Socket*** **decaf::internal::net::ssl::DefaultSSLSocketFactory::createSocket** (const **decaf::net::InetAddress** * *host*, int *port*, const **decaf::net::InetAddress** * *ifAddress*, int *localPort*) [virtual]

Creates a new **Socket** (p. 2452) object and connects it to the specified remote host and port using the configuration of this **SocketFactory** (p. 2473).

The **Socket** (p. 2452) will be bound to the specified local address and port.

Parameters

<i>host</i>	The host to connect the socket to.
<i>port</i>	The port on the remote host to connect to.
<i>ifAddress</i>	The address on the local machine to bind the Socket (p. 2452) to.
<i>localPort</i>	The local port to bind the Socket (p. 2452) to.

Returns

a new **Socket** (p. 2452) object, caller must free this object when done.

Exceptions

<i>IOException</i>	if an I/O error occurs while creating the Socket (p. 2452) object.
UnknownHostException (p. 2816)	if the host name is not known.

Implements **decaf::net::SocketFactory** (p. 2476).

6.196.3.4 virtual **decaf::net::Socket*** **decaf::internal::net::ssl::DefaultSSL-SocketFactory::createSocket** (const std::string & *name*, int *port*)
[virtual]

Creates a new **Socket** (p. 2452) object and connects it to the specified remote host and port using the configuration of this **SocketFactory** (p. 2473).

Parameters

<i>host</i>	The host name or IP address to connect the socket to.
<i>port</i>	The port on the remote host to connect to.

Returns

a new **Socket** (p. 2452) object, caller must free this object when done.

Exceptions

<i>IOException</i>	if an I/O error occurs while creating the Socket (p. 2452) object.
UnknownHostException (p. 2816)	if the host name is not known.

Implements **decaf::net::SocketFactory** (p. 2476).

6.196 decaf::internal::net::ssl::DefaultSSLSocketFactory Class Reference 1179

6.196.3.5 virtual **decaf::net::Socket*** **decaf::internal::net::ssl::DefaultSSL-
SocketFactory::createSocket** (const std::string & *name*, int *port*, const
decaf::net::InetAddress * *ifAddress*, int *localPort*) [virtual]

Creates a new **Socket** (p. 2452) object and connects it to the specified remote host and port using the configuration of this **SocketFactory** (p. 2473).

Parameters

<i>host</i>	The host name or IP address to connect the socket to.
<i>port</i>	The port on the remote host to connect to.
<i>ifAddress</i>	The address on the local machine to bind the Socket (p. 2452) to.
<i>localPort</i>	The local port to bind the Socket (p. 2452) to.

Returns

a new **Socket** (p. 2452) object, caller must free this object when done.

Exceptions

<i>IOException</i>	if an I/O error occurs while creating the Socket (p. 2452) object.
UnknownHost- Exception (p. 2816)	if the host name is not known.

Implements **decaf::net::SocketFactory** (p. 2477).

6.196.3.6 virtual **decaf::net::Socket*** **decaf::internal::net::ssl::DefaultSSLSocket-
Factory::createSocket** (decaf::net::Socket * *socket*, std::string *host*, int
port, bool *autoClose*) [virtual]

Returns a socket layered over an existing socket connected to the named host, at the given port.

This constructor can be used when tunneling SSL through a proxy or when negotiating the use of SSL over an existing socket. The host and port refer to the logical peer destination. This socket is configured using the socket options established for this factory.

Parameters

<i>socket</i>	The existing socket to layer over.
<i>host</i>	The server host the original Socket (p. 2452) is connected to.
<i>port</i>	The server port the original Socket (p. 2452) is connected to.
<i>autoClose</i>	Should the layered over Socket (p. 2452) be closed when the topmost socket is closed.

Returns

a new **Socket** (p. 2452) instance that wraps the given **Socket** (p. 2452).

Exceptions

<i>IOException</i>	if an I/O exception occurs while performing this operation.
UnknownHostException (p. 2816)	if the host is unknown.

Implements **decaf::net::ssl::SSLSocketFactory** (p. 2523).

```
6.196.3.7  virtual std::vector<std::string> decaf::internal::net::ssl:-
            DefaultSSLSocketFactory::getDefaultCipherSuites ( )
            [virtual]
```

Returns the list of cipher suites which are enabled by default.

Unless a different list is enabled, handshaking on an SSL connection will use one of these cipher suites. The minimum quality of service for these defaults requires confidentiality protection and server authentication (that is, no anonymous cipher suites).

Returns

an STL vector containing the list of cipher suites enabled by default.

See also

getSupportedCipherSuites() (p. 2524)

Implements **decaf::net::ssl::SSLSocketFactory** (p. 2524).

```
6.196.3.8  virtual std::vector<std::string> decaf::internal::net::ssl:-
            DefaultSSLSocketFactory::getSupportedCipherSuites ( )
            [virtual]
```

Returns the names of the cipher suites which could be enabled for use on an SSL connection.

Normally, only a subset of these will actually be enabled by default, since this list may include cipher suites which do not meet quality of service requirements for those defaults. Such cipher suites are useful in specialized applications.

Returns

an STL vector containing the list of supported cipher suites.

See also

getDefaultCipherSuites() (p. 2524)

Implements **decaf::net::ssl::SSLSocketFactory** (p. 2524).

The documentation for this class was generated from the following file:

- src/main/decaf/internal/net/ssl/**DefaultSSLSocketFactory.h**

6.197 activemq::transport::DefaultTransportListener Class - Reference

A Utility class that create empty implementations for the **TransportListener** (p. 2810) interface so that a subclass only needs to override the one's its interested.

```
#include <src/main/activemq/transport/DefaultTransport-  
Listener.h>
```

Inheritance diagram for activemq::transport::DefaultTransportListener:

Public Member Functions

- virtual **~DefaultTransportListener** ()
- virtual void **onCommand** (const **Pointer**< **Command** > &command AMQCPP_UNUSED)
Event handler for the receipt of a command.
- virtual void **onException** (const **decaf::lang::Exception** &ex AMQCPP_UNUSED)
Event handler for an exception from a command transport.
- virtual void **transportInterrupted** ()
The transport has suffered an interruption from which it hopes to recover.
- virtual void **transportResumed** ()
The transport has resumed after an interruption.

6.197.1 Detailed Description

A Utility class that create empty implementations for the **TransportListener** (p. 2810) interface so that a subclass only needs to override the one's its interested.

6.197.2 Constructor & Destructor Documentation

6.197.2.1 `virtual activemq::transport::DefaultTransportListener-
::~DefaultTransportListener () [inline,
virtual]`

6.197.3 Member Function Documentation

6.197.3.1 `virtual void activemq::transport::DefaultTransportListener::onCommand (
const Pointer< Command > &command AMQCPP_UNUSED) [inline,
virtual]`

Event handler for the receipt of a command.

The transport passes off all received commands to its listeners, the listener then owns the Object. If there is no registered listener the **Transport** (p. 2790) deletes the command upon receipt.

Parameters

<i>command</i>	the received command object.
----------------	------------------------------

6.197.3.2 `virtual void activemq::transport::DefaultTransportListener::onException (const decaf::lang::Exception &ex AMQCPP_UNUSED) [inline,
virtual]`

Event handler for an exception from a command transport.

Parameters

<i>ex</i>	The exception.
-----------	----------------

6.197.3.3 `virtual void activemq::transport::DefaultTransport-
Listener::transportInterrupted () [inline,
virtual]`

The transport has suffered an interruption from which it hopes to recover.

Implements **activemq::transport::TransportListener** (p. 2812).

6.197.3.4 `virtual void activemq::transport::DefaultTransport-
Listener::transportResumed () [inline,
virtual]`

The transport has resumed after an interruption.

Implements **activemq::transport::TransportListener** (p. 2812).

The documentation for this class was generated from the following file:

- `src/main/activemq/transport/DefaultTransportListener.h`

6.198 decaf::util::zip::Deflater Class Reference

This class compresses data using the *DEFLATE* algorithm (see [specification](#)).

```
#include <src/main/decaf/util/zip/Deflater.h>
```

Public Member Functions

- **Deflater** (int level, bool nowrap=false)
Creates a new compressor using the specified compression level.
- **Deflater** ()
Creates a new compressor with the default compression level.
- virtual ~**Deflater** ()
- void **setInput** (const unsigned char *buffer, int size, int offset, int length)
Sets input data for compression.
- void **setInput** (const std::vector< unsigned char > &buffer, int offset, int length)
Sets input data for compression.
- void **setInput** (const std::vector< unsigned char > &buffer)
Sets input data for compression.
- void **setDictionary** (const unsigned char *buffer, int size, int offset, int length)
Sets preset dictionary for compression.
- void **setDictionary** (const std::vector< unsigned char > &buffer, int offset, int length)
Sets preset dictionary for compression.
- void **setDictionary** (const std::vector< unsigned char > &buffer)
Sets preset dictionary for compression.
- void **setStrategy** (int strategy)
Sets the compression strategy to the specified value.
- void **setLevel** (int level)
Sets the compression level to the specified value.
- bool **needsInput** () const
- void **finish** ()
When called, indicates that compression should end with the current contents of the input buffer.
- bool **finished** () const
- int **deflate** (unsigned char *buffer, int size, int offset, int length)
Fills specified buffer with compressed data.
- int **deflate** (std::vector< unsigned char > &buffer, int offset, int length)
Fills specified buffer with compressed data.
- int **deflate** (std::vector< unsigned char > &buffer)
Fills specified buffer with compressed data.
- long long **getAdler** () const
- long long **getBytesRead** () const
- long long **getBytesWritten** () const

- void **reset** ()
Resets deflater so that a new set of input data can be processed.
- void **end** ()
Closes the compressor and discards any unprocessed input.

Static Public Attributes

- static const int **BEST_SPEED**
Compression level for fastest compression.
- static const int **BEST_COMPRESSION**
Compression level for best compression.
- static const int **DEFAULT_COMPRESSION**
Default compression level.
- static const int **DEFLATED**
Compression method for the deflate algorithm (the only one currently supported).
- static const int **NO_COMPRESSION**
Compression level for no compression.
- static const int **FILTERED**
Compression strategy best used for data consisting mostly of small values with a somewhat random distribution.
- static const int **HUFFMAN_ONLY**
Compression strategy for Huffman coding only.
- static const int **DEFAULT_STRATEGY**
Default compression strategy.

6.198.1 Detailed Description

This class compresses data using the *DEFLATE* algorithm (see *specification*).

Basically this class is part of the API to the stream based ZLIB compression library and is used as such by **DeflaterOutputStream** (p. 1189) and its descendants.

The typical usage of a **Deflater** (p. 1180) instance outside this package consists of a specific call to one of its constructors before being passed to an instance of **DeflaterOutputStream** (p. 1189).

See also

DeflaterOutputStream (p. 1189)
Inflater (p. 1448)

Since

1.0

6.198.2 Constructor & Destructor Documentation

6.198.2.1 decaf::util::zip::Deflater::Deflater (int *level*, bool *nowrap* = false)

Creates a new compressor using the specified compression level.

If 'nowrap' is true then the ZLIB header and checksum fields will not be used in order to support the compression format used in both GZIP and PKZIP.

Parameters

<i>level</i>	The compression level to use (0-9).
<i>nowrap</i>	If true uses GZip compatible compression (defaults to false).

6.198.2.2 decaf::util::zip::Deflater::Deflater ()

Creates a new compressor with the default compression level.

Compressed data will be generated in ZLIB format.

6.198.2.3 virtual decaf::util::zip::Deflater::~~Deflater () [virtual]

6.198.3 Member Function Documentation

6.198.3.1 int decaf::util::zip::Deflater::deflate (unsigned char * *buffer*, int *size*, int *offset*, int *length*)

Fills specified buffer with compressed data.

Returns actual number of bytes of compressed data. A return value of 0 indicates that **needsInput()** (p. 1185) should be called in order to determine if more input data is required.

Parameters

<i>buffer</i>	The Buffer to write the compressed data to.
<i>size</i>	The size of the passed buffer.
<i>offset</i>	The position in the Buffer to start writing at.
<i>length</i>	The maximum number of byte of data to write.

Returns

the actual number of bytes of compressed data.

Exceptions

<i>NullPointerException</i>	if buffer is NULL.
<i>IndexOutOfBoundsException</i>	if the offset + length > size of the buffer.
<i>IllegalStateException</i>	if in the end state.

6.198.3.2 `int decaf::util::zip::Deflater::deflate (std::vector< unsigned char > & buffer,
int offset, int length)`

Fills specified buffer with compressed data.

Returns actual number of bytes of compressed data. A return value of 0 indicates that **needsInput()** (p. 1185) should be called in order to determine if more input data is required.

Parameters

<i>buffer</i>	The Buffer to write the compressed data to.
<i>offset</i>	The position in the Buffer to start writing at.
<i>length</i>	The maximum number of byte of data to write.

Returns

the actual number of bytes of compressed data.

Exceptions

<i>IndexOutOfBoundsException</i>	if the offset + length > size of the buffer.
<i>IllegalStateException</i>	if in the end state.

6.198.3.3 `int decaf::util::zip::Deflater::deflate (std::vector< unsigned char > & buffer)`

Fills specified buffer with compressed data.

Returns actual number of bytes of compressed data. A return value of 0 indicates that **needsInput()** (p. 1185) should be called in order to determine if more input data is required.

Parameters

<i>buffer</i>	The Buffer to write the compressed data to.
---------------	---

Returns

the actual number of bytes of compressed data.

Exceptions

<i>IllegalStateException</i>	if in the end state.
------------------------------	----------------------

6.198.3.4 void decaf::util::zip::Deflater::end ()

Closes the compressor and discards any unprocessed input.

This method should be called when the compressor is no longer being used, but will also be called automatically by the destructor. Once this method is called, the behavior of the **Deflater** (p. 1180) object is undefined.

6.198.3.5 void decaf::util::zip::Deflater::finish ()

When called, indicates that compression should end with the current contents of the input buffer.

6.198.3.6 bool decaf::util::zip::Deflater::finished () const**Returns**

true if the end of the compressed data output stream has been reached.

6.198.3.7 long long decaf::util::zip::Deflater::getAdler () const**Returns**

the ADLER-32 value of the uncompressed data.

Exceptions

<i>IllegalStateException</i>	if in the end state.
------------------------------	----------------------

6.198.3.8 long long decaf::util::zip::Deflater::getBytesRead () const**Returns**

the total number of uncompressed bytes input so far.

Exceptions

<i>IllegalStateException</i>	if in the end state.
------------------------------	----------------------

6.198.3.9 long long decaf::util::zip::Deflater::getBytesWritten () const**Returns**

the total number of compressed bytes output so far.

Exceptions

<i>IllegalStateException</i>	if in the end state.
------------------------------	----------------------

6.198.3.10 **bool decaf::util::zip::Deflater::needsInput () const**

Returns

true if the input data buffer is empty and **setInput()** (p. 1186) should be called in order to provide more input

6.198.3.11 **void decaf::util::zip::Deflater::reset ()**

Resets deflater so that a new set of input data can be processed.

Keeps current compression level and strategy settings.

Exceptions

<i>IllegalStateException</i>	if in the end state.
------------------------------	----------------------

6.198.3.12 **void decaf::util::zip::Deflater::setDictionary (const unsigned char * *buffer*, int *size*, int *offset*, int *length*)**

Sets preset dictionary for compression.

A preset dictionary is used when the history buffer can be predetermined. When the data is later uncompressed with **Inflater.inflate()** (p. 1451), **Inflater.getAdler()** (p. 1450) can be called in order to get the Adler-32 value of the dictionary required for decompression.

Parameters

<i>buffer</i>	The buffer containing the preset dictionary.
<i>size</i>	The size of the passed dictionary buffer.
<i>offset</i>	The position in the Buffer to start reading from.
<i>length</i>	The number of bytes to read from the input buffer.

Exceptions

<i>NullPointerException</i>	if buffer is NULL.
<i>IndexOutOfBoundsException</i>	if the offset + length > size of the buffer.
<i>IllegalStateException</i>	if in the end state.

6.198.3.13 void `decaf::util::zip::Deflater::setDictionary` (const std::vector< unsigned char > & *buffer*, int *offset*, int *length*)

Sets preset dictionary for compression.

A preset dictionary is used when the history buffer can be predetermined. When the data is later uncompressed with **Inflater.inflate()** (p. 1451), **Inflater.getAdler()** (p. 1450) can be called in order to get the Adler-32 value of the dictionary required for decompression.

Parameters

<i>buffer</i>	The buffer containing the preset dictionary.
<i>offset</i>	The position in the Buffer to start reading from.
<i>length</i>	The number of bytes to read from the input buffer.

Exceptions

<i>IndexOutOfBoundsException</i>	if the offset + length > size of the buffer.
<i>IllegalStateException</i>	if in the end state.

6.198.3.14 void `decaf::util::zip::Deflater::setDictionary` (const std::vector< unsigned char > & *buffer*)

Sets preset dictionary for compression.

A preset dictionary is used when the history buffer can be predetermined. When the data is later uncompressed with **Inflater.inflate()** (p. 1451), **Inflater.getAdler()** (p. 1450) can be called in order to get the Adler-32 value of the dictionary required for decompression.

Parameters

<i>buffer</i>	The buffer containing the preset dictionary.
---------------	--

Exceptions

<i>IllegalStateException</i>	if in the end state.
------------------------------	----------------------

6.198.3.15 void `decaf::util::zip::Deflater::setInput` (const unsigned char * *buffer*, int *size*, int *offset*, int *length*)

Sets input data for compression.

This should be called whenever **needsInput()** (p. 1185) returns true indicating that more input data is required.

Parameters

<i>buffer</i>	The Buffer to read in for compression.
<i>size</i>	The size in bytes of the buffer passed.
<i>offset</i>	The position in the Buffer to start reading from.
<i>length</i>	The number of bytes to read from the input buffer.

Exceptions

<i>NullPointerException</i>	if buffer is NULL.
<i>IndexOutOfBoundsException</i>	if the offset + length > size of the buffer.
<i>IllegalStateException</i>	if in the end state.

6.198.3.16 `void decaf::util::zip::Deflater::setInput (const std::vector< unsigned char > & buffer, int offset, int length)`

Sets input data for compression.

This should be called whenever **needsInput()** (p. 1185) returns true indicating that more input data is required.

Parameters

<i>buffer</i>	The Buffer to read in for compression.
<i>offset</i>	The position in the Buffer to start reading from.
<i>length</i>	The number of bytes to read from the input buffer.

Exceptions

<i>IndexOutOfBoundsException</i>	if the offset + length > size of the buffer.
<i>IllegalStateException</i>	if in the end state.

6.198.3.17 `void decaf::util::zip::Deflater::setInput (const std::vector< unsigned char > & buffer)`

Sets input data for compression.

This should be called whenever **needsInput()** (p. 1185) returns true indicating that more input data is required.

Parameters

<i>buffer</i>	The Buffer to read in for compression.
---------------	--

Exceptions

<i>IllegalStateException</i>	if in the end state.
------------------------------	----------------------

6.198.3.18 void decaf::util::zip::Deflater::setLevel (int *level*)

Sets the compression level to the specified value.

Parameters

<i>level</i>	The new Compression level to use.
--------------	-----------------------------------

Exceptions

<i>IllegalArgument-Exception</i>	if the level value is invalid.
<i>IllegalStateException</i>	if in the end state.

6.198.3.19 void decaf::util::zip::Deflater::setStrategy (int *strategy*)

Sets the compression strategy to the specified value.

Parameters

<i>strategy</i>	The new Compression strategy to use.
-----------------	--------------------------------------

Exceptions

<i>IllegalArgument-Exception</i>	if the strategy value is invalid.
<i>IllegalStateException</i>	if in the end state.

6.198.4 Field Documentation

6.198.4.1 const int decaf::util::zip::Deflater::BEST_COMPRESSION [static]

Compression level for best compression.

6.198.4.2 const int decaf::util::zip::Deflater::BEST_SPEED [static]

Compression level for fastest compression.

6.198.4.3 `const int decaf::util::zip::Deflater::DEFAULT_COMPRESSION`
[static]

Default compression level.

6.198.4.4 `const int decaf::util::zip::Deflater::DEFAULT_STRATEGY` [static]

Default compression strategy.

6.198.4.5 `const int decaf::util::zip::Deflater::DEFLATED` [static]

Compression method for the deflate algorithm (the only one currently supported).

6.198.4.6 `const int decaf::util::zip::Deflater::FILTERED` [static]

Compression strategy best used for data consisting mostly of small values with a somewhat random distribution.

Forces more Huffman coding and less string matching.

6.198.4.7 `const int decaf::util::zip::Deflater::HUFFMAN_ONLY` [static]

Compression strategy for Huffman coding only.

6.198.4.8 `const int decaf::util::zip::Deflater::NO_COMPRESSION` [static]

Compression level for no compression.

The documentation for this class was generated from the following file:

- `src/main/decaf/util/zip/Deflater.h`

6.199 decaf::util::zip::DeflaterOutputStream Class Reference

Provides a `FilterOutputStream` instance that compresses the data before writing it to the wrapped `OutputStream`.

```
#include <src/main/decaf/util/zip/DeflaterOutputStream.h>
```

Inheritance diagram for `decaf::util::zip::DeflaterOutputStream`:

Public Member Functions

- **DeflaterOutputStream** (decaf::io::OutputStream *outputStream, bool own=false)

*Creates a new DeflateOutputStream with a Default **Deflater** (p. 1180) and buffer size.*

- **DeflaterOutputStream** (decaf::io::OutputStream *outputStream, Deflater *deflater, bool own=false, bool ownDeflater=false)

*Creates a new DeflateOutputStream with a user supplied **Deflater** (p. 1180) and a default buffer size.*

- **DeflaterOutputStream** (decaf::io::OutputStream *outputStream, Deflater *deflater, int bufferSize, bool own=false, bool ownDeflater=false)

*Creates a new DeflateOutputStream with a user supplied **Deflater** (p. 1180) and specified buffer size.*

- virtual ~**DeflaterOutputStream** ()
- virtual void **finish** ()

Finishes writing any remaining data to the wrapped OutputStream but does not close it upon completion.

- virtual void **close** ()

*Closes this object and deallocates the appropriate resources.
The object is generally no longer usable after calling close.*

Exceptions

IOException (p. 1545)	<i>if an error occurs while closing.</i>
------------------------------	--

The default implementation of this method does nothing.

*The close method of **FilterOutputStream** (p. 1341) calls its flush method, and then calls the close method of its underlying output stream.*

Protected Member Functions

- virtual void **doWriteByte** (unsigned char value)
- virtual void **doWriteArrayBounded** (const unsigned char *buffer, int size, int offset, int length)
- virtual void **deflate** ()

Writes a buffers worth of compressed data to the wrapped OutputStream.

Protected Attributes

- **Deflater** * **deflater**

*The **Deflater** (p. 1180) for this stream.*

- std::vector< unsigned char > **buf**

The Buffer to use for.

- bool **ownDeflater**
- bool **isDone**

Static Protected Attributes

- static const std::size_t **DEFAULT_BUFFER_SIZE**

6.199.1 Detailed Description

Provides a FilterOutputStream instance that compresses the data before writing it to the wrapped OutputStream.

Since

1.0

6.199.2 Constructor & Destructor Documentation

6.199.2.1 `decaf::util::zip::DeflaterOutputStream::DeflaterOutputStream (decaf::io::OutputStream * outputStream, bool own = false)`

Creates a new DeflateOutputStream with a Default **Deflater** (p. 1180) and buffer size.

Parameters

<i>output-Stream</i>	The OutputStream instance to wrap.
<i>own</i>	Should this filter take ownership of the OutputStream pointer (default is false).

6.199.2.2 `decaf::util::zip::DeflaterOutputStream::DeflaterOutputStream (decaf::io::OutputStream * outputStream, Deflater * deflater, bool own = false, bool ownDeflater = false)`

Creates a new DeflateOutputStream with a user supplied **Deflater** (p. 1180) and a default buffer size.

When the user supplied a **Deflater** (p. 1180) instance the DeflaterOutputpotStream does not take ownership of the **Deflater** (p. 1180) pointer unless the ownDeflater parameter is set to true, the caller is still responsible for deleting the **Deflater** (p. 1180) when own-Deflater is false.

Parameters

<i>output-Stream</i>	The OutputStream instance to wrap.
<i>deflater</i>	The user supplied Deflater (p. 1180) to use for compression. (
<i>own</i>	Should this filter take ownership of the OutputStream pointer (default is false).
<i>ownDeflater</i>	Should the filter take ownership of the passed Deflater (p. 1180) object (default is false).

Exceptions

<i>NullPointerException</i>	if the Deflater (p. 1180) given is NULL.
-----------------------------	---

6.199.2.3 `decaf::util::zip::DeflaterOutputStream::DeflaterOutputStream (decaf::io::OutputStream * outputStream, Deflater * deflater, int bufferSize, bool own = false, bool ownDeflater = false)`

Creates a new DeflateOutputStream with a user supplied **Deflater** (p. 1180) and specified buffer size.

When the user supplied a **Deflater** (p. 1180) instance the DeflaterOutputpotStream does not take ownership of the **Deflater** (p. 1180) pointer unless the ownDeflater parameter is set to true, otherwise the caller is still responsible for deleting the **Deflater** (p. 1180).

Parameters

<i>output-Stream</i>	The OutputStream instance to wrap.
<i>deflater</i>	The user supplied Deflater (p. 1180) to use for compression.
<i>bufferSize</i>	The size of the input buffer.
<i>own</i>	Should this filter take ownership of the OutputStream pointer (default is false).
<i>ownDeflater</i>	Should the filter take ownership of the passed Deflater (p. 1180) object (default is false).

Exceptions

<i>NullPointerException</i>	if the Deflater (p. 1180) given is NULL.
<i>IllegalArgument-Exception</i>	if bufferSize is 0.

6.199.2.4 `virtual decaf::util::zip::DeflaterOutputStream::~~DeflaterOutputStream ()` [virtual]

6.199.3 Member Function Documentation

6.199.3.1 `virtual void decaf::util::zip::DeflaterOutputStream::close ()` [virtual]

Closes this object and deallocates the appropriate resources.

The object is generally no longer usable after calling close.

Exceptions

<i>IOException</i> (p. 1545)	if an error occurs while closing.
------------------------------	-----------------------------------

The default implementation of this method does nothing.

The close method of **FilterOutputStream** (p. 1341) calls its flush method, and then calls the close method of its underlying output stream.

Finishes writing any remaining data to the OutputStream then closes the stream.

Reimplemented from **decaf::io::FilterOutputStream** (p. 1342).

6.199.3.2 `virtual void decaf::util::zip::DeflaterOutputStream::deflate ()`
[protected, virtual]

Writes a buffers worth of compressed data to the wrapped OutputStream.

6.199.3.3 `virtual void decaf::util::zip::DeflaterOutputStream::doWriteArrayBounded (const unsigned char * buffer, int size, int offset, int length)` [protected, virtual]

Reimplemented from **decaf::io::FilterOutputStream** (p. 1343).

6.199.3.4 `virtual void decaf::util::zip::DeflaterOutputStream::doWriteByte (unsigned char value)` [protected, virtual]

Reimplemented from **decaf::io::FilterOutputStream** (p. 1343).

6.199.3.5 `virtual void decaf::util::zip::DeflaterOutputStream::finish ()`
[virtual]

Finishes writing any remaining data to the wrapped OutputStream but does not close it upon completion.

Exceptions

<i>IOException</i>	if an I/O error occurs.
--------------------	-------------------------

6.199.4 Field Documentation

6.199.4.1 `std::vector<unsigned char> decaf::util::zip::DeflaterOutputStream::buf`
[protected]

The Buffer to use for.

6.199.4.2 `const std::size_t decaf::util::zip::DeflaterOutputStream::DEFAULT_BUFFER_SIZE` [static, protected]

6.199.4.3 **Deflater*** `decaf::util::zip::DeflaterOutputStream::deflater`
`[protected]`

The **Deflater** (p. 1180) for this stream.

6.199.4.4 **bool** `decaf::util::zip::DeflaterOutputStream::isDone` `[protected]`

6.199.4.5 **bool** `decaf::util::zip::DeflaterOutputStream::ownDeflater`
`[protected]`

The documentation for this class was generated from the following file:

- `src/main/decaf/util/zip/DeflaterOutputStream.h`

6.200 decaf::util::concurrent::Delayed Class Reference

A mix-in style interface for marking objects that should be acted upon after a given delay.

```
#include <src/main/decaf/util/concurrent/Delayed.h>
```

Inheritance diagram for `decaf::util::concurrent::Delayed`:

Public Member Functions

- virtual **~Delayed** ()
- virtual long long **getDelay** (const **TimeUnit** &unit)=0

Returns the remaining delay associated with this object, in the given time unit.

6.200.1 Detailed Description

A mix-in style interface for marking objects that should be acted upon after a given delay.

An implementation of this interface must define a `Comparable` methods that provides an ordering consistent with its `getDelay` method.

6.200.2 Constructor & Destructor Documentation

6.200.2.1 **virtual** `decaf::util::concurrent::Delayed::~Delayed ()` `[inline, virtual]`

6.200.3 Member Function Documentation

6.200.3.1 `virtual long long decaf::util::concurrent::Delayed::getDelay (const TimeUnit & unit) [pure virtual]`

Returns the remaining delay associated with this object, in the given time unit.

Parameters

<i>unit</i>	The time unit
-------------	---------------

Returns

the remaining delay; zero or negative values indicate that the delay has already elapsed

The documentation for this class was generated from the following file:

- `src/main/decaf/util/concurrent/Delayed.h`

6.201 cms::DeliveryMode Class Reference

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

```
#include <src/main/cms/DeliveryMode.h>
```

Public Types

- enum **DELIVERY_MODE** { **PERSISTENT** = 0, **NON_PERSISTENT** = 1 }

*Enumeration values for **Message** (p. 1839) Delivery Mode.*

Public Member Functions

- virtual `~DeliveryMode ()`

6.201.1 Detailed Description

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

When a client sends a **cms::Message** (p. 1839) it can mark the **Message** (p. 1839) as either Persistent or Non-Persistent. If the client feels that the **Message** (p. 1839) cannot be lost in transit it should mark it as Persistent, otherwise if it is allowable for a **Message** (p. 1839) to occasionally be lost it can mark it as Non-Persistent. This allows the Provider to balance make tradeoffs between balance and **Message** (p. 1839) throughput.

The **DeliveryMode** (p. 1195) covers only the transport of the **Message** (p. 1839) for sending client to its destination and doesn't apply to the receiving **Message** (p. 1839) consumer. The receiving Consumer can drop **Message** (p. 1839)'s based on configuration such as memory limits or **Message** (p. 1839) filtering.

A message is guaranteed to be delivered once and only once by a CMS provider if the delivery mode of the message is PERSISTENT and the configuration of the **Message** (p. 1839) consumer allows for it.

Since

1.0

6.201.2 Member Enumeration Documentation

6.201.2.1 enum cms::DeliveryMode::DELIVERY_MODE

Enumeration values for **Message** (p. 1839) Delivery Mode.

Enumerator:

PERSISTENT

NON_PERSISTENT

6.201.3 Constructor & Destructor Documentation

6.201.3.1 virtual cms::DeliveryMode::~~DeliveryMode () [virtual]

The documentation for this class was generated from the following file:

- src/main/cms/**DeliveryMode.h**

6.202 decaf::util::Deque< E > Class Template Reference

Defines a 'Double ended **Queue** (p. 2222)' interface that allows for insertion and removal of elements from both ends.

```
#include <src/main/decaf/util/Deque.h>
```

Inheritance diagram for decaf::util::Deque< E >:

Public Member Functions

- virtual **~Deque** ()
- virtual void **addFirst** (const E &element)=0

*Inserts an element onto the front of the **Deque** (p. 1196) if possible without violating the implementations capacity restrictions.*

- virtual void **addLast** (const E &element)=0

*Inserts an element onto the end of the **Deque** (p. 1196) if possible without violating the implementations capacity restrictions.*

- virtual bool **offerFirst** (const E &element)=0

*This method attempts to insert the given element into the **Deque** (p. 1196) at the front end.*

- virtual bool **offerLast** (const E &element)=0

*This method attempts to insert the given element into the **Deque** (p. 1196) at the end.*

- virtual E **removeFirst** ()=0

*Removes the topmost element from the **Deque** (p. 1196) and returns it.*

- virtual E **removeLast** ()=0

*Removes the last element from the **Deque** (p. 1196) and returns it.*

- virtual bool **pollFirst** (E &element)=0

*Removes the first element from the **Deque** (p. 1196) assigns it to the element reference passed.*

- virtual bool **pollLast** (E &element)=0

*Removes the last element from the **Deque** (p. 1196) assigns it to the element reference passed.*

- virtual E & **getFirst** ()=0

*Attempts to fetch a reference to the first element in the **Deque** (p. 1196).*

- virtual const E & **getFirst** () const =0

- virtual E & **getLast** ()=0

*Attempts to fetch a reference to the last element in the **Deque** (p. 1196).*

- virtual const E & **getLast** () const =0

- virtual bool **peekFirst** (E &value) const =0

*Retrieves the first element contained in this **Deque** (p. 1196) and assigns its value to the reference value passed the value however is not removed from the **Deque** (p. 1196).*

- virtual bool **peekLast** (E &value) const =0

*Retrieves the last element contained in this **Deque** (p. 1196) and assigns its value to the reference value passed the value however is not removed from the **Deque** (p. 1196).*

- virtual bool **removeFirstOccurrence** (const E &value)=0

*Removes the first occurrence of the specified element from this **Deque** (p. 1196).*

- virtual bool **removeLastOccurrence** (const E &value)=0

*Removes the last occurrence of the specified element from this **Deque** (p. 1196).*

- virtual void **push** (const E &element)=0

Pushes an element onto the stack represented by this deque (in other words, at the head of this deque) if it is possible to do so immediately without violating capacity restrictions, otherwise it throwing an `IllegalStateException` if no space is currently available.

- virtual E **pop** ()=0

*Treats this **Deque** (p. 1196) as a stack and attempts to pop an element off the top.*

- virtual **Iterator**< E > * **descendingIterator** ()=0

Provides an **Iterator** (p. 1559) over this **Collection** (p. 851) that traverses the element in reverse order.

- virtual **Iterator**< E > * **descendingIterator** () const =0

6.202.1 Detailed Description

template<typename E>class decaf::util::Deque< E >

Defines a 'Double ended **Queue** (p. 2222)' interface that allows for insertion and removal of elements from both ends.

Generally there is no limit on the number of elements that can be placed into a **Deque** (p. 1196).

Unlike a **List** (p. 1658) the **Deque** (p. 1196) doesn't provide index element based access, however methods are provided to grant access to interior elements.

Since

1.0

6.202.2 Constructor & Destructor Documentation

6.202.2.1 template<typename E> virtual decaf::util::Deque< E >::~~Deque ()
[inline, virtual]

6.202.3 Member Function Documentation

6.202.3.1 template<typename E> virtual void decaf::util::Deque< E >::addFirst (const E & *element*) [pure virtual]

Inserts an element onto the front of the **Deque** (p. 1196) if possible without violating the implementations capacity restrictions.

For a capacity restricted **Deque** (p. 1196) it is preferable to call offerFirst instead.

Parameters

<i>element</i>	The element to be placed at the front of the Deque (p. 1196).
----------------	--

Exceptions

<i>IllegalStateException</i>	if the element cannot be added at this time due to capacity restrictions
<i>NullPointerException</i>	if the specified element is NULL and this deque is a Collection (p. 851) of pointers and does not permit null elements.
<i>IllegalArgumentException</i>	if some property of the specified element prevents it from being added to this deque.

Implemented in `decaf::util::LinkedList< E >` (p. 1643), `decaf::util::LinkedList< Pointer< Transport > >` (p. 1643), `decaf::util::LinkedList< cms::MessageConsumer * >` (p. 1643), `decaf::util::LinkedList< CompositeTask * >` (p. 1643), `decaf::util::LinkedList< URI >` (p. 1643), `decaf::util::LinkedList< Pointer< MessageDispatch > >` (p. 1643), `decaf::util::LinkedList< Pointer< DestinationInfo > >` (p. 1643), `decaf::util::LinkedList< PrimitiveValueNode >` (p. 1643), `decaf::util::LinkedList< Pointer< Command > >` (p. 1643), `decaf::util::LinkedList< Pointer< BackupTransport > >` (p. 1643), `decaf::util::LinkedList< cms::MessageProducer * >` (p. 1643), `decaf::util::LinkedList< cms::Destination * >` (p. 1643), `decaf::util::LinkedList< cms::Session * >` (p. 1643), and `decaf::util::LinkedList< cms::Connection * >` (p. 1643).

6.202.3.2 `template<typename E> virtual void decaf::util::Deque< E >::addLast (const E & element) [pure virtual]`

Inserts an element onto the end of the **Deque** (p. 1196) if possible without violating the implementations capacity restrictions.

For a capacity restricted **Deque** (p. 1196) it is preferable to call `offerLast` instead.

Parameters

<i>element</i>	The element to be placed at the end of the Deque (p. 1196).
----------------	--

Exceptions

<i>IllegalStateException</i>	if the element cannot be added at this time due to capacity restrictions
<i>NullPointerException</i>	if the specified element is NULL and this deque is a Collection (p. 851) of pointers and does not permit null elements.
<i>IllegalArgumentException</i>	if some property of the specified element prevents it from being added to this deque.

Implemented in `decaf::util::LinkedList< E >` (p. 1643), `decaf::util::LinkedList< Pointer< Transport > >` (p. 1643), `decaf::util::LinkedList< cms::MessageConsumer * >` (p. 1643), `decaf::util::LinkedList< CompositeTask * >` (p. 1643), `decaf::util::LinkedList< URI >` (p. 1643), `decaf::util::LinkedList< Pointer< MessageDispatch > >` (p. 1643), `decaf::util::LinkedList< Pointer< DestinationInfo > >` (p. 1643), `decaf::util::LinkedList< PrimitiveValueNode >` (p. 1643), `decaf::util::LinkedList< Pointer< Command > >` (p. 1643), `decaf::util::LinkedList< Pointer< BackupTransport > >` (p. 1643), `decaf::util::LinkedList< cms::MessageProducer * >` (p. 1643), `decaf::util::LinkedList< cms::Destination * >` (p. 1643), `decaf::util::LinkedList< cms::Session * >` (p. 1643), and `decaf::util::LinkedList< cms::Connection * >` (p. 1643).

6.202.3.3 `template<typename E> virtual Iterator<E>* decaf::util::Deque< E
>::descendingIterator () [pure virtual]`

Provides an **Iterator** (p. 1559) over this **Collection** (p. 851) that traverses the element in reverse order.

Returns

a new **Iterator** (p. 1559) instance that moves from last to first.

Implemented in `decaf::util::LinkedList< E >` (p. 1645), `decaf::util::LinkedList< Pointer< Transport > >` (p. 1645), `decaf::util::LinkedList< cms::MessageConsumer * >` (p. 1645), `decaf::util::LinkedList< CompositeTask * >` (p. 1645), `decaf::util::LinkedList< URI >` (p. 1645), `decaf::util::LinkedList< Pointer< MessageDispatch > >` (p. 1645), `decaf::util::LinkedList< Pointer< DestinationInfo > >` (p. 1645), `decaf::util::LinkedList< PrimitiveValueNode >` (p. 1645), `decaf::util::LinkedList< Pointer< Command > >` (p. 1645), `decaf::util::LinkedList< Pointer< BackupTransport > >` (p. 1645), `decaf::util::LinkedList< cms::MessageProducer * >` (p. 1645), `decaf::util::LinkedList< cms::Destination * >` (p. 1645), `decaf::util::LinkedList< cms::Session * >` (p. 1645), and `decaf::util::LinkedList< cms::Connection * >` (p. 1645).

6.202.3.4 `template<typename E> virtual Iterator<E>* decaf::util::Deque< E
>::descendingIterator () const [pure virtual]`

Implemented in `decaf::util::LinkedList< E >` (p. 1645), `decaf::util::LinkedList< Pointer< Transport > >` (p. 1645), `decaf::util::LinkedList< cms::MessageConsumer * >` (p. 1645), `decaf::util::LinkedList< CompositeTask * >` (p. 1645), `decaf::util::LinkedList< URI >` (p. 1645), `decaf::util::LinkedList< Pointer< MessageDispatch > >` (p. 1645), `decaf::util::LinkedList< Pointer< DestinationInfo > >` (p. 1645), `decaf::util::LinkedList< PrimitiveValueNode >` (p. 1645), `decaf::util::LinkedList< Pointer< Command > >` (p. 1645), `decaf::util::LinkedList< Pointer< BackupTransport > >` (p. 1645), `decaf::util::LinkedList< cms::MessageProducer * >` (p. 1645), `decaf::util::LinkedList< cms::Destination * >` (p. 1645), `decaf::util::LinkedList< cms::Session * >` (p. 1645), and `decaf::util::LinkedList< cms::Connection * >` (p. 1645).

6.202.3.5 `template<typename E> virtual E& decaf::util::Deque< E >::getFirst ()
[pure virtual]`

Attempts to fetch a reference to the first element in the **Deque** (p. 1196).

This method does not remove the element from the **Deque** (p. 1196) but simply returns a reference to it.

Returns

reference to the first element in the **Deque** (p. 1196).

Exceptions

<i>NoSuchElementException</i> (p. 1984)	if the Deque (p. 1196) is empty.
---	---

Implemented in **decaf::util::LinkedList< E >** (p. 1646), **decaf::util::LinkedList< Pointer< Transport > >** (p. 1646), **decaf::util::LinkedList< cms::MessageConsumer * >** (p. 1646), **decaf::util::LinkedList< CompositeTask * >** (p. 1646), **decaf::util::LinkedList< URI >** (p. 1646), **decaf::util::LinkedList< Pointer< MessageDispatch > >** (p. 1646), **decaf::util::LinkedList< Pointer< DestinationInfo > >** (p. 1646), **decaf::util::LinkedList< PrimitiveValueNode >** (p. 1646), **decaf::util::LinkedList< Pointer< Command > >** (p. 1646), **decaf::util::LinkedList< Pointer< BackupTransport > >** (p. 1646), **decaf::util::LinkedList< cms::MessageProducer * >** (p. 1646), **decaf::util::LinkedList< cms::Destination * >** (p. 1646), **decaf::util::LinkedList< cms::Session * >** (p. 1646), and **decaf::util::LinkedList< cms::Connection * >** (p. 1646).

6.202.3.6 `template<typename E> virtual const E& decaf::util::Deque< E >::getFirst ()
const [pure virtual]`

Implemented in **decaf::util::LinkedList< E >** (p. 1647), **decaf::util::LinkedList< Pointer< Transport > >** (p. 1647), **decaf::util::LinkedList< cms::MessageConsumer * >** (p. 1647), **decaf::util::LinkedList< CompositeTask * >** (p. 1647), **decaf::util::LinkedList< URI >** (p. 1647), **decaf::util::LinkedList< Pointer< MessageDispatch > >** (p. 1647), **decaf::util::LinkedList< Pointer< DestinationInfo > >** (p. 1647), **decaf::util::LinkedList< PrimitiveValueNode >** (p. 1647), **decaf::util::LinkedList< Pointer< Command > >** (p. 1647), **decaf::util::LinkedList< Pointer< BackupTransport > >** (p. 1647), **decaf::util::LinkedList< cms::MessageProducer * >** (p. 1647), **decaf::util::LinkedList< cms::Destination * >** (p. 1647), **decaf::util::LinkedList< cms::Session * >** (p. 1647), and **decaf::util::LinkedList< cms::Connection * >** (p. 1647).

6.202.3.7 `template<typename E> virtual E& decaf::util::Deque< E >::getLast ()
[pure virtual]`

Attempts to fetch a reference to the last element in the **Deque** (p. 1196).

This method does not remove the element from the **Deque** (p. 1196) but simply returns a reference to it.

Returns

reference to the last element in the **Deque** (p. 1196).

Exceptions

<i>NoSuchElementException</i> (p. 1984)	if the Deque (p. 1196) is empty.
---	---

Implemented in `decaf::util::LinkedList< E >` (p. 1647), `decaf::util::LinkedList< Pointer< Transport > >` (p. 1647), `decaf::util::LinkedList< cms::MessageConsumer * >` (p. 1647), `decaf::util::LinkedList< CompositeTask * >` (p. 1647), `decaf::util::LinkedList< URI >` (p. 1647), `decaf::util::LinkedList< Pointer< MessageDispatch > >` (p. 1647), `decaf::util::LinkedList< Pointer< DestinationInfo > >` (p. 1647), `decaf::util::LinkedList< PrimitiveValueNode >` (p. 1647), `decaf::util::LinkedList< Pointer< Command > >` (p. 1647), `decaf::util::LinkedList< Pointer< BackupTransport > >` (p. 1647), `decaf::util::LinkedList< cms::MessageProducer * >` (p. 1647), `decaf::util::LinkedList< cms::Destination * >` (p. 1647), `decaf::util::LinkedList< cms::Session * >` (p. 1647), and `decaf::util::LinkedList< cms::Connection * >` (p. 1647).

6.202.3.8 `template<typename E> virtual const E& decaf::util::Deque< E >::getLast ()`
`const [pure virtual]`

Implemented in `decaf::util::LinkedList< E >` (p. 1647), `decaf::util::LinkedList< Pointer< Transport > >` (p. 1647), `decaf::util::LinkedList< cms::MessageConsumer * >` (p. 1647), `decaf::util::LinkedList< CompositeTask * >` (p. 1647), `decaf::util::LinkedList< URI >` (p. 1647), `decaf::util::LinkedList< Pointer< MessageDispatch > >` (p. 1647), `decaf::util::LinkedList< Pointer< DestinationInfo > >` (p. 1647), `decaf::util::LinkedList< PrimitiveValueNode >` (p. 1647), `decaf::util::LinkedList< Pointer< Command > >` (p. 1647), `decaf::util::LinkedList< Pointer< BackupTransport > >` (p. 1647), `decaf::util::LinkedList< cms::MessageProducer * >` (p. 1647), `decaf::util::LinkedList< cms::Destination * >` (p. 1647), `decaf::util::LinkedList< cms::Session * >` (p. 1647), and `decaf::util::LinkedList< cms::Connection * >` (p. 1647).

6.202.3.9 `template<typename E> virtual bool decaf::util::Deque< E >::offerFirst (const E & element)` `[pure virtual]`

This method attempts to insert the given element into the **Deque** (p. 1196) at the front end.

Unlike the `addFirst` method that throws an exception if it cannot insert the element due to capacity restrictions etc this method returns false if it cannot insert the element.

Parameters

<i>element</i>	The element to add to this Deque (p. 1196).
----------------	--

Returns

true if the element was added, false otherwise.

Exceptions

<i>NullPointerException</i>	if the specified element is NULL and this deque is a Collection (p. 851) of pointers and does not permit null elements.
<i>IllegalArgumentException</i>	if some property of the specified element prevents it from being added to this deque.

Implemented in `decaf::util::LinkedList< E >` (p. 1650), `decaf::util::LinkedList< Pointer< Transport > >` (p. 1650), `decaf::util::LinkedList< cms::MessageConsumer * >` (p. 1650), `decaf::util::LinkedList< CompositeTask * >` (p. 1650), `decaf::util::LinkedList< URI >` (p. 1650), `decaf::util::LinkedList< Pointer< MessageDispatch > >` (p. 1650), `decaf::util::LinkedList< Pointer< DestinationInfo > >` (p. 1650), `decaf::util::LinkedList< PrimitiveValueNode >` (p. 1650), `decaf::util::LinkedList< Pointer< Command > >` (p. 1650), `decaf::util::LinkedList< Pointer< BackupTransport > >` (p. 1650), `decaf::util::LinkedList< cms::MessageProducer * >` (p. 1650), `decaf::util::LinkedList< cms::Destination * >` (p. 1650), `decaf::util::LinkedList< cms::Session * >` (p. 1650), and `decaf::util::LinkedList< cms::Connection * >` (p. 1650).

6.202.3.10 `template<typename E> virtual bool decaf::util::Deque< E >::offerLast (const E & element)` [pure virtual]

This method attempts to insert the given element into the **Deque** (p. 1196) at the end.

Unlike the `addLast` method that throws an exception if it cannot insert the element due to capacity restrictions etc this method returns false if it cannot insert the element.

Parameters

<i>element</i>	The element to add to this Deque (p. 1196).
----------------	--

Returns

true if the element was added, false otherwise.

Exceptions

<i>NullPointerException</i>	if the specified element is NULL and this deque is a Collection (p. 851) of pointers and does not permit null elements.
<i>IllegalArgumentException</i>	if some property of the specified element prevents it from being added to this deque.

Implemented in `decaf::util::LinkedList< E >` (p. 1650), `decaf::util::LinkedList< Pointer< Transport > >` (p. 1650), `decaf::util::LinkedList< cms::MessageConsumer * >` (p. 1650), `decaf::util::LinkedList< CompositeTask * >` (p. 1650), `decaf::util::LinkedList< URI >` (p. 1650), `decaf::util::LinkedList< Pointer< MessageDispatch > >` (p. 1650), `decaf::util::LinkedList< Pointer< DestinationInfo > >` (p. 1650), `decaf::util::LinkedList< PrimitiveValueNode >` (p. 1650), `decaf::util::LinkedList< Pointer< Command > >` (p. 1650), `decaf::util::LinkedList< Pointer< BackupTransport > >` (p. 1650), `decaf::util::LinkedList< cms::MessageProducer * >` (p. 1650), `decaf::util::LinkedList< cms::Destination * >` (p. 1650), `decaf::util::LinkedList< cms::Session * >` (p. 1650), and `decaf::util::LinkedList< cms::Connection * >` (p. 1650).

6.202.3.11 `template<typename E> virtual bool decaf::util::Deque< E >::peekFirst (E & value) const [pure virtual]`

Retrieves the first element contained in this **Deque** (p. 1196) and assigns its value to the reference value passed the value however is not removed from the **Deque** (p. 1196).

If this call is successful it returns true. Unlike `getFirst` this method does not throw an exception if the **Deque** (p. 1196) is empty.

Returns

true if an element was assigned to the reference passed, false otherwise.

Implemented in `decaf::util::LinkedList< E >` (p. 1651), `decaf::util::LinkedList< Pointer< Transport > >` (p. 1651), `decaf::util::LinkedList< cms::MessageConsumer * >` (p. 1651), `decaf::util::LinkedList< CompositeTask * >` (p. 1651), `decaf::util::LinkedList< URI >` (p. 1651), `decaf::util::LinkedList< Pointer< MessageDispatch > >` (p. 1651), `decaf::util::LinkedList< Pointer< DestinationInfo > >` (p. 1651), `decaf::util::LinkedList< PrimitiveValueNode >` (p. 1651), `decaf::util::LinkedList< Pointer< Command > >` (p. 1651), `decaf::util::LinkedList< Pointer< BackupTransport > >` (p. 1651), `decaf::util::LinkedList< cms::MessageProducer * >` (p. 1651), `decaf::util::LinkedList< cms::Destination * >` (p. 1651), `decaf::util::LinkedList< cms::Session * >` (p. 1651), and `decaf::util::LinkedList< cms::Connection * >` (p. 1651).

6.202.3.12 `template<typename E> virtual bool decaf::util::Deque< E >::peekLast (E & value) const [pure virtual]`

Retrieves the last element contained in this **Deque** (p. 1196) and assigns its value to the reference value passed the value however is not removed from the **Deque** (p. 1196).

If this call is successful it returns true. Unlike `getLast` this method does not throw an exception if the **Deque** (p. 1196) is empty.

Returns

true if an element was assigned to the reference passed, false otherwise.

Implemented in `decaf::util::LinkedList< E >` (p. 1652), `decaf::util::LinkedList< Pointer< Transport > >` (p. 1652), `decaf::util::LinkedList< cms::MessageConsumer * >` (p. 1652), `decaf::util::LinkedList< CompositeTask * >` (p. 1652), `decaf::util::LinkedList< URI >` (p. 1652), `decaf::util::LinkedList< Pointer< MessageDispatch > >` (p. 1652), `decaf::util::LinkedList< Pointer< DestinationInfo > >` (p. 1652), `decaf::util::LinkedList< PrimitiveValueNode >` (p. 1652), `decaf::util::LinkedList< Pointer< Command > >` (p. 1652), `decaf::util::LinkedList< Pointer< BackupTransport > >` (p. 1652), `decaf::util::LinkedList< cms::MessageProducer * >` (p. 1652), `decaf::util::LinkedList< cms::Destination * >` (p. 1652), `decaf::util::LinkedList< cms::Session * >` (p. 1652), and `decaf::util::LinkedList< cms::Connection * >` (p. 1652).

6.202.3.13 `template<typename E> virtual bool decaf::util::Deque< E >::pollFirst (E & element)` [pure virtual]

Removes the first element from the **Deque** (p. 1196) assigns it to the element reference passed.

Parameters

<i>element</i>	Reference to an variable that can be assigned the value of the head of this Deque (p. 1196).
----------------	---

Returns

true if an element was available to remove, false otherwise.

Implemented in **decaf::util::LinkedList< E >** (p. 1652), **decaf::util::LinkedList< Pointer< Transport > >** (p. 1652), **decaf::util::LinkedList< cms::MessageConsumer * >** (p. 1652), **decaf::util::LinkedList< CompositeTask * >** (p. 1652), **decaf::util::LinkedList< URI >** (p. 1652), **decaf::util::LinkedList< Pointer< MessageDispatch > >** (p. 1652), **decaf::util::LinkedList< Pointer< DestinationInfo > >** (p. 1652), **decaf::util::LinkedList< PrimitiveValueNode >** (p. 1652), **decaf::util::LinkedList< Pointer< Command > >** (p. 1652), **decaf::util::LinkedList< Pointer< BackupTransport > >** (p. 1652), **decaf::util::LinkedList< cms::MessageProducer * >** (p. 1652), **decaf::util::LinkedList< cms::Destination * >** (p. 1652), **decaf::util::LinkedList< cms::Session * >** (p. 1652), and **decaf::util::LinkedList< cms::Connection * >** (p. 1652).

6.202.3.14 `template<typename E> virtual bool decaf::util::Deque< E >::pollLast (E & element)` [pure virtual]

Removes the last element from the **Deque** (p. 1196) assigns it to the element reference passed.

Parameters

<i>element</i>	Reference to an variable that can be assigned the value of the tail of this Deque (p. 1196).
----------------	---

Returns

true if an element was available to remove, false otherwise.

Implemented in **decaf::util::LinkedList< E >** (p. 1653), **decaf::util::LinkedList< Pointer< Transport > >** (p. 1653), **decaf::util::LinkedList< cms::MessageConsumer * >** (p. 1653), **decaf::util::LinkedList< CompositeTask * >** (p. 1653), **decaf::util::LinkedList< URI >** (p. 1653), **decaf::util::LinkedList< Pointer< MessageDispatch > >** (p. 1653), **decaf::util::LinkedList< Pointer< DestinationInfo > >** (p. 1653), **decaf::util::LinkedList< PrimitiveValueNode >** (p. 1653), **decaf::util::LinkedList< Pointer< Command > >** (p. 1653), **decaf::util::LinkedList< Pointer< BackupTransport > >** (p. 1653), **decaf::util::LinkedList< cms::MessageProducer * >** (p. 1653), **decaf::util::LinkedList< cms::Destination * >** (p. 1653), **decaf::util::LinkedList< cms::Session * >** (p. 1653), and **decaf::util::LinkedList< cms::Connection * >** (p. 1653).

List< **Pointer**< **BackupTransport** > > (p. 1653), **decaf::util::LinkedList**< **cms::MessageProducer** * > (p. 1653), **decaf::util::LinkedList**< **cms::Destination** * > (p. 1653), **decaf::util::LinkedList**< **cms::Session** * > (p. 1653), and **decaf::util::LinkedList**< **cms::Connection** * > (p. 1653).

6.202.3.15 `template<typename E> virtual E decaf::util::Deque< E >::pop ()` [pure virtual]

Treats this **Deque** (p. 1196) as a stack and attempts to pop an element off the top.

If there's no element to pop then a `NoSuchElementException` is thrown, otherwise the top element is removed and assigned to the reference passed.

This operation performs the same basic function as the `removeFirst` method.

Returns

the element at the front of this deque which would be the top of a stack.

Exceptions

<i>NoSuchElementException</i> (p. 1984)	if there is nothing on the top of the stack.
---	--

Implemented in **decaf::util::LinkedList**< **E** > (p. 1653), **decaf::util::LinkedList**< **Pointer**< **Transport** > > (p. 1653), **decaf::util::LinkedList**< **cms::MessageConsumer** * > (p. 1653), **decaf::util::LinkedList**< **CompositeTask** * > (p. 1653), **decaf::util::LinkedList**< **URI** > (p. 1653), **decaf::util::LinkedList**< **Pointer**< **MessageDispatch** > > (p. 1653), **decaf::util::LinkedList**< **Pointer**< **DestinationInfo** > > (p. 1653), **decaf::util::LinkedList**< **PrimitiveValueNode** > (p. 1653), **decaf::util::LinkedList**< **Pointer**< **Command** > > (p. 1653), **decaf::util::LinkedList**< **Pointer**< **BackupTransport** > > (p. 1653), **decaf::util::LinkedList**< **cms::MessageProducer** * > (p. 1653), **decaf::util::LinkedList**< **cms::Destination** * > (p. 1653), **decaf::util::LinkedList**< **cms::Session** * > (p. 1653), and **decaf::util::LinkedList**< **cms::Connection** * > (p. 1653).

6.202.3.16 `template<typename E> virtual void decaf::util::Deque< E >::push (const E & element)` [pure virtual]

Pushes an element onto the stack represented by this deque (in other words, at the head of this deque) if it is possible to do so immediately without violating capacity restrictions, otherwise it throwing an `IllegalStateException` if no space is currently available.

This method performs the same basic operation as the `addFirst` method.

Parameters

<i>element</i>	The element to be pushed onto the Deque (p. 1196).
----------------	---

Exceptions

<i>IllegalStateException</i>	if the element cannot be added at this time due to capacity restrictions
<i>NullPointerException</i>	if the specified element is NULL and this deque is a Collection (p. 851) of pointers and does not permit null elements.
<i>IllegalArgument-Exception</i>	if some property of the specified element prevents it from being added to this deque.

Implemented in **decaf::util::LinkedList< E >** (p. 1654), **decaf::util::LinkedList< Pointer< Transport > >** (p. 1654), **decaf::util::LinkedList< cms::MessageConsumer * >** (p. 1654), **decaf::util::LinkedList< CompositeTask * >** (p. 1654), **decaf::util::LinkedList< URI >** (p. 1654), **decaf::util::LinkedList< Pointer< MessageDispatch > >** (p. 1654), **decaf::util::LinkedList< Pointer< DestinationInfo > >** (p. 1654), **decaf::util::LinkedList< PrimitiveValueNode >** (p. 1654), **decaf::util::LinkedList< Pointer< Command > >** (p. 1654), **decaf::util::LinkedList< Pointer< BackupTransport > >** (p. 1654), **decaf::util::LinkedList< cms::MessageProducer * >** (p. 1654), **decaf::util::LinkedList< cms::Destination * >** (p. 1654), **decaf::util::LinkedList< cms::Session * >** (p. 1654), and **decaf::util::LinkedList< cms::Connection * >** (p. 1654).

6.202.3.17 `template<typename E> virtual E decaf::util::Deque< E >::removeFirst ()`
`[pure virtual]`

Removes the topmost element from the **Deque** (p. 1196) and returns it.

Unlike the pollFirst method this method throws a NoSuchElementException if the **Deque** (p. 1196) is empty.

Returns

the element at the Head of the **Deque** (p. 1196).

Exceptions

<i>NoSuchElementException</i> (p. 1984)	if the Deque (p. 1196) is empty.
--	---

Implemented in **decaf::util::LinkedList< E >** (p. 1655), **decaf::util::LinkedList< Pointer< Transport > >** (p. 1655), **decaf::util::LinkedList< cms::MessageConsumer * >** (p. 1655), **decaf::util::LinkedList< CompositeTask * >** (p. 1655), **decaf::util::LinkedList< URI >** (p. 1655), **decaf::util::LinkedList< Pointer< MessageDispatch > >** (p. 1655), **decaf::util::LinkedList< Pointer< DestinationInfo > >** (p. 1655), **decaf::util::LinkedList< PrimitiveValueNode >** (p. 1655), **decaf::util::LinkedList< Pointer< Command > >** (p. 1655), **decaf::util::LinkedList< Pointer< BackupTransport > >** (p. 1655), **decaf::util::LinkedList< cms::MessageProducer * >** (p. 1655), **decaf::util::LinkedList< cms::Destination * >** (p. 1655), **decaf::util::LinkedList< cms::Session * >** (p. 1655), and **decaf::util::-**

LinkedList< **cms::Connection** * > (p. 1655).

6.202.3.18 `template<typename E> virtual bool decaf::util::Deque< E
>::removeFirstOccurrence (const E & value) [pure virtual]`

Removes the first occurrence of the specified element from this **Deque** (p. 1196).

If there is no matching element then the **Deque** (p. 1196) is left unchanged.

Parameters

<i>value</i>	The value to be removed from this Deque (p. 1196).
--------------	---

Returns

true if the **Deque** (p. 1196) was modified as a result of this operation.

Exceptions

<i>NullPointerException</i>	if the specified element is NULL and this deque is a Collection (p. 851) of pointers and does not permit null elements.
-----------------------------	--

Implemented in **decaf::util::LinkedList**< **E** > (p. 1656), **decaf::util::LinkedList**< **Pointer**< **Transport** > > (p. 1656), **decaf::util::LinkedList**< **cms::MessageConsumer** * > (p. 1656), **decaf::util::LinkedList**< **CompositeTask** * > (p. 1656), **decaf::util::LinkedList**< **URI** > (p. 1656), **decaf::util::LinkedList**< **Pointer**< **MessageDispatch** > > (p. 1656), **decaf::util::LinkedList**< **Pointer**< **DestinationInfo** > > (p. 1656), **decaf::util::LinkedList**< **PrimitiveValueNode** > (p. 1656), **decaf::util::LinkedList**< **Pointer**< **Command** > > (p. 1656), **decaf::util::LinkedList**< **Pointer**< **BackupTransport** > > (p. 1656), **decaf::util::LinkedList**< **cms::MessageProducer** * > (p. 1656), **decaf::util::LinkedList**< **cms::Destination** * > (p. 1656), **decaf::util::LinkedList**< **cms::Session** * > (p. 1656), and **decaf::util::LinkedList**< **cms::Connection** * > (p. 1656).

6.202.3.19 `template<typename E> virtual E decaf::util::Deque< E >::removeLast ()
[pure virtual]`

Removes the last element from the **Deque** (p. 1196) and returns it.

Unlike the pollLast method this method throws a NoSuchElementException if the **Deque** (p. 1196) is empty.

Returns

the element at the Tail of the **Deque** (p. 1196).

Exceptions

<i>NoSuchElementException</i> (p. 1984)	if the Deque (p. 1196) is empty.
---	---

Implemented in **decaf::util::LinkedList< E >** (p. 1656), **decaf::util::LinkedList< Pointer< Transport > >** (p. 1656), **decaf::util::LinkedList< cms::MessageConsumer * >** (p. 1656), **decaf::util::LinkedList< CompositeTask * >** (p. 1656), **decaf::util::LinkedList< URI >** (p. 1656), **decaf::util::LinkedList< Pointer< MessageDispatch > >** (p. 1656), **decaf::util::LinkedList< Pointer< DestinationInfo > >** (p. 1656), **decaf::util::LinkedList< PrimitiveValueNode >** (p. 1656), **decaf::util::LinkedList< Pointer< Command > >** (p. 1656), **decaf::util::LinkedList< Pointer< BackupTransport > >** (p. 1656), **decaf::util::LinkedList< cms::MessageProducer * >** (p. 1656), **decaf::util::LinkedList< cms::Destination * >** (p. 1656), **decaf::util::LinkedList< cms::Session * >** (p. 1656), and **decaf::util::LinkedList< cms::Connection * >** (p. 1656).

6.202.3.20 `template<typename E> virtual bool decaf::util::Deque< E >::removeLastOccurrence (const E & value) [pure virtual]`

Removes the last occurrence of the specified element from this **Deque** (p. 1196).

If there is no matching element then the **Deque** (p. 1196) is left unchanged.

Parameters

<i>value</i>	The value to be removed from this Deque (p. 1196).
--------------	---

Returns

true if the **Deque** (p. 1196) was modified as a result of this operation.

Exceptions

<i>NullPointerException</i>	if the specified element is NULL and this deque is a Collection (p. 851) of pointers and does not permit null elements.
------------------------------------	--

Implemented in **decaf::util::LinkedList< E >** (p. 1657), **decaf::util::LinkedList< Pointer< Transport > >** (p. 1657), **decaf::util::LinkedList< cms::MessageConsumer * >** (p. 1657), **decaf::util::LinkedList< CompositeTask * >** (p. 1657), **decaf::util::LinkedList< URI >** (p. 1657), **decaf::util::LinkedList< Pointer< MessageDispatch > >** (p. 1657), **decaf::util::LinkedList< Pointer< DestinationInfo > >** (p. 1657), **decaf::util::LinkedList< PrimitiveValueNode >** (p. 1657), **decaf::util::LinkedList< Pointer< Command > >** (p. 1657), **decaf::util::LinkedList< Pointer< BackupTransport > >** (p. 1657), **decaf::util::LinkedList< cms::MessageProducer * >** (p. 1657), **decaf::util::LinkedList< cms::Destination * >** (p. 1657), **decaf::util::LinkedList< cms::Session * >** (p. 1657), and **decaf::util::LinkedList< cms::Connection * >** (p. 1657).

The documentation for this class was generated from the following file:

- src/main/decaf/util/**Deque.h**

6.203 cms::Destination Class Reference

A **Destination** (p. 1210) object encapsulates a provider-specific address.

```
#include <src/main/cms/Destination.h>
```

Inheritance diagram for cms::Destination:

Public Types

- enum **DestinationType** { **TOPIC**, **QUEUE**, **TEMPORARY_TOPIC**, **TEMPORARY_QUEUE** }

Public Member Functions

- virtual **~Destination** () throw ()
- virtual **DestinationType** **getDestinationType** () const =0
*Retrieve the **Destination** (p. 1210) Type for this **Destination** (p. 1210).*
- virtual **cms::Destination** * **clone** () const =0
Creates a new instance of this destination type that is a copy of this one, and returns it.
- virtual void **copy** (const **cms::Destination** &source)=0
*Copies the contents of the given **Destination** (p. 1210) object to this one.*
- virtual bool **equals** (const **cms::Destination** &other) const =0
*Compares two **Destination** (p. 1210) instances to determine if they represent the same logic **Destination** (p. 1210).*
- virtual const **CMSProperties** & **getCMSProperties** () const =0
Retrieve any properties that might be part of the destination that was specified.

6.203.1 Detailed Description

A **Destination** (p. 1210) object encapsulates a provider-specific address.

There is no standard definition of a **Destination** (p. 1210) address, each provider can provide its own definition and there can be configuration data attached to the **Destination** (p. 1210) address.

All CMS **Destination** (p. 1210) objects support concurrent use.

Since

1.0

6.203.2 Member Enumeration Documentation

6.203.2.1 enum cms::Destination::DestinationType

Enumerator:

TOPIC

QUEUE

TEMPORARY_TOPIC

TEMPORARY_QUEUE

6.203.3 Constructor & Destructor Documentation

6.203.3.1 virtual cms::Destination::~~Destination () throw () [virtual]

6.203.4 Member Function Documentation

6.203.4.1 virtual cms::Destination* cms::Destination::clone () const [pure virtual]

Creates a new instance of this destination type that is a copy of this one, and returns it.

Returns

cloned copy of this object

Implemented in **activemq::commands::ActiveMQQueue** (p. 322), **activemq::commands::ActiveMQTopic** (p. 415), **activemq::commands::ActiveMQTempQueue** (p. 388), and **activemq::commands::ActiveMQTempTopic** (p. 397).

Referenced by **activemq::commands::ActiveMQMessageTemplate< cms::ObjectMessage >::setCMSDestination()**, and **activemq::commands::ActiveMQMessageTemplate< cms::ObjectMessage >::setCMSReplyTo()**.

6.203.4.2 virtual void cms::Destination::copy (const cms::Destination & source) [pure virtual]

Copies the contents of the given **Destination** (p. 1210) object to this one.

Parameters

<i>source</i>	The source Destination (p. 1210) object.
---------------	---

Implemented in `activemq::commands::ActiveMQQueue` (p. 323), `activemq::commands::ActiveMQTopic` (p. 416), `activemq::commands::ActiveMQTempQueue` (p. 388), and `activemq::commands::ActiveMQTempTopic` (p. 397).

6.203.4.3 `virtual bool cms::Destination::equals (const cms::Destination & other) const [pure virtual]`

Compares two **Destination** (p. 1210) instances to determine if they represent the same logic **Destination** (p. 1210).

Parameters

<i>other</i>	The other destination to compare this one to.
--------------	---

Returns

true if the two destinations are the same.

Implemented in `activemq::commands::ActiveMQQueue` (p. 323), `activemq::commands::ActiveMQTopic` (p. 417), `activemq::commands::ActiveMQTempQueue` (p. 389), and `activemq::commands::ActiveMQTempTopic` (p. 399).

6.203.4.4 `virtual const CMSProperties& cms::Destination::getCMSProperties () const [pure virtual]`

Retrieve any properties that might be part of the destination that was specified.

This is a deviation from the JMS spec but necessary due to C++ restrictions.

Returns

A {const} reference to a **CMSProperties** (p. 830) object.

Implemented in `activemq::commands::ActiveMQQueue` (p. 324), `activemq::commands::ActiveMQTopic` (p. 417), `activemq::commands::ActiveMQTempQueue` (p. 390), and `activemq::commands::ActiveMQTempTopic` (p. 399).

6.203.4.5 `virtual DestinationType cms::Destination::getDestinationType () const [pure virtual]`

Retrieve the **Destination** (p. 1210) Type for this **Destination** (p. 1210).

Returns

The **Destination** (p. 1210) Type

Implemented in `activemq::commands::ActiveMQQueue` (p. 325), `activemq::commands::ActiveMQTopic` (p. 418), `activemq::commands::ActiveMQTempQueue` (p. 390), and `activemq::commands::ActiveMQTempTopic` (p. 400).

The documentation for this class was generated from the following file:

- src/main/cms/**Destination.h**

6.204 activemq::commands::ActiveMQDestination::DestinationFilter Struct Reference

```
#include <src/main/activemq/commands/ActiveMQDestination.h>
```

Static Public Attributes

- static const std::string **ANY_CHILD**
- static const std::string **ANY_DESCENDENT**

6.204.1 Field Documentation

6.204.1.1 const std::string activemq::commands::ActiveMQDestination::DestinationFilter::ANY_CHILD [static]

6.204.1.2 const std::string activemq::commands::ActiveMQDestination::DestinationFilter::ANY_DESCENDENT [static]

The documentation for this struct was generated from the following file:

- src/main/activemq/commands/**ActiveMQDestination.h**

6.205 activemq::commands::DestinationInfo Class Reference

```
#include <src/main/activemq/commands/DestinationInfo.h>
```

Inheritance diagram for activemq::commands::DestinationInfo:

Public Member Functions

- **DestinationInfo** ()
- virtual **~DestinationInfo** ()
- virtual unsigned char **getDataStructureType** () const
*Get the **DataStructure** (p. 1133) Type as defined in CommandTypes.h.*
- virtual **DestinationInfo** * **cloneDataStructure** () const
Clone this object and return a new instance that the caller now owns, this will be an exact copy of this one.

- virtual void **copyDataStructure** (const **DataStructure** *src)

Copy the contents of the passed object into this objects members, overwriting any existing data.
- virtual std::string **toString** () const

*Returns a string containing the information for this **DataStructure** (p. 1133) such as its type and value of its elements.*
- virtual bool **equals** (const **DataStructure** *value) const

*Compares the **DataStructure** (p. 1133) passed in to this one, and returns if they are equivalent.*
- virtual const **Pointer** < **ConnectionId** > & **getConnectionId** () const
- virtual **Pointer**< **ConnectionId** > & **getConnectionId** ()
- virtual void **setConnectionId** (const **Pointer**< **ConnectionId** > &connectionId)
- virtual const **Pointer** < **ActiveMQDestination** > & **getDestination** () const
- virtual **Pointer** < **ActiveMQDestination** > & **getDestination** ()
- virtual void **setDestination** (const **Pointer**< **ActiveMQDestination** > &destination)
- virtual unsigned char **getOperationType** () const
- virtual void **setOperationType** (unsigned char operationType)
- virtual long long **getTimeout** () const
- virtual void **setTimeout** (long long timeout)
- virtual const std::vector < **decaf::lang::Pointer** < **BrokerId** > > & **getBrokerPath** () const
- virtual std::vector < **decaf::lang::Pointer** < **BrokerId** > > & **getBrokerPath** ()
- virtual void **setBrokerPath** (const std::vector< **decaf::lang::Pointer**< **BrokerId** > > &brokerPath)
- virtual **Pointer**< **Command** > **visit** (activemq::state::CommandVisitor *visitor)

Allows a Visitor to visit this command and return a response to the command based on the command type being visited.

Static Public Attributes

- static const unsigned char **ID_DESTINATIONINFO** = 8

Protected Attributes

- **Pointer**< **ConnectionId** > **connectionId**
- **Pointer**< **ActiveMQDestination** > **destination**
- unsigned char **operationType**
- long long **timeout**
- std::vector < **decaf::lang::Pointer** < **BrokerId** > > **brokerPath**

6.205.1 Constructor & Destructor Documentation

6.205.1.1 `activemq::commands::DestinationInfo::DestinationInfo ()`

6.205.1.2 `virtual activemq::commands::DestinationInfo::~~DestinationInfo ()`
[virtual]

6.205.2 Member Function Documentation

6.205.2.1 `virtual DestinationInfo* activemq::commands::DestinationInfo::clone-
DataStructure () const` [virtual]

Clone this object and return a new instance that the caller now owns, this will be an exact copy of this one.

Returns

new copy of this object.

Implements `activemq::commands::DataStructure` (p. 1133).

6.205.2.2 `virtual void activemq::commands::DestinationInfo::copyDataStructure (`
`const DataStructure * src)` [virtual]

Copy the contents of the passed object into this objects members, overwriting any existing data.

Returns

src - Source Object

Reimplemented from `activemq::commands::BaseCommand` (p. 493).

6.205.2.3 `virtual bool activemq::commands::DestinationInfo::equals (const`
`DataStructure * value) const` [virtual]

Compares the `DataStructure` (p. 1133) passed in to this one, and returns if they are equivalent.

Equivalent here means that they are of the same type, and that each element of the objects are the same.

Returns

true if `DataStructure` (p. 1133)'s are Equal.

Reimplemented from `activemq::commands::BaseCommand` (p. 494).

- 6.205.2.4 `virtual const std::vector< decaf::lang::Pointer<BrokerId> >& activemq::commands::DestinationInfo::getBrokerPath () const`
[virtual]
- 6.205.2.5 `virtual std::vector< decaf::lang::Pointer<BrokerId> >& activemq::commands::DestinationInfo::getBrokerPath ()`
[virtual]
- 6.205.2.6 `virtual const Pointer<ConnectionId>& activemq::commands::DestinationInfo::getConnectionId () const`
[virtual]
- 6.205.2.7 `virtual Pointer<ConnectionId>& activemq::commands::DestinationInfo::getConnectionId ()` [virtual]
- 6.205.2.8 `virtual unsigned char activemq::commands::DestinationInfo::getData-StructureType () const` [virtual]

Get the **DataStructure** (p. 1133) Type as defined in CommandTypes.h.

Returns

The type of the data structure

Implements **activemq::commands::DataStructure** (p. 1137).

- 6.205.2.9 `virtual const Pointer<ActiveMQDestination>& activemq::commands::DestinationInfo::getDestination () const`
[virtual]

Referenced by `activemq::state::ConnectionState::removeTempDestination()`.

- 6.205.2.10 `virtual Pointer<ActiveMQDestination>& activemq::commands::DestinationInfo::getDestination ()`
[virtual]
- 6.205.2.11 `virtual unsigned char activemq::commands::DestinationInfo::get-OperationType () const` [virtual]
- 6.205.2.12 `virtual long long activemq::commands::DestinationInfo::getTimeout () const` [virtual]
- 6.205.2.13 `virtual void activemq::commands::DestinationInfo::setBrokerPath (const std::vector< decaf::lang::Pointer< BrokerId > > & brokerPath)`
[virtual]
- 6.205.2.14 `virtual void activemq::commands::DestinationInfo::setConnectionId (const Pointer< ConnectionId > & connectionId)` [virtual]

6.205.2.15 `virtual void activemq::commands::DestinationInfo::setDestination (const Pointer< ActiveMQDestination > & destination) [virtual]`

6.205.2.16 `virtual void activemq::commands::DestinationInfo::setOperationType (unsigned char operationType) [virtual]`

6.205.2.17 `virtual void activemq::commands::DestinationInfo::setTimeout (long long timeout) [virtual]`

6.205.2.18 `virtual std::string activemq::commands::DestinationInfo::toString () const [virtual]`

Returns a string containing the information for this **DataStructure** (p. 1133) such as its type and value of its elements.

Returns

formatted string useful for debugging.

Reimplemented from **activemq::commands::BaseCommand** (p. 498).

6.205.2.19 `virtual Pointer<Command> activemq::commands::DestinationInfo::visit (activemq::state::CommandVisitor * visitor) [virtual]`

Allows a Visitor to visit this command and return a response to the command based on the command type being visited.

The command will call the proper processXXX method in the visitor.

Returns

a **Response** (p. 2298) to the visitor being called or NULL if no response.

Implements **activemq::commands::Command** (p. 871).

6.205.3 Field Documentation

6.205.3.1 `std::vector< decaf::lang::Pointer<BrokerId> > activemq::commands::DestinationInfo::brokerPath [protected]`

6.205.3.2 `Pointer<ConnectionId> activemq::commands::DestinationInfo::connectionId [protected]`

6.205.3.3 `Pointer<ActiveMQDestination> activemq::commands::DestinationInfo::destination [protected]`

6.205.3.4 `const unsigned char activemq::commands::DestinationInfo::ID_DESTINATIONINFO = 8 [static]`

6.206

activemq::wireformat::openwire::marshal::generated::DestinationInfoMarshaller
Class Reference 1221

6.205.3.5 unsigned char **activemq::commands::DestinationInfo::operationType**
[protected]

6.205.3.6 long long **activemq::commands::DestinationInfo::timeout**
[protected]

The documentation for this class was generated from the following file:

- src/main/activemq/commands/**DestinationInfo.h**

6.206 **activemq::wireformat::openwire::marshal::generated::-** **DestinationInfoMarshaller Class Reference**

Marshaling code for Open Wire Format for **DestinationInfoMarshaller** (p. 1218).

```
#include <src/main/activemq/wireformat/openwire/marshal/generated/-  
DestinationInfoMarshaller.h>
```

Inheritance diagram for **activemq::wireformat::openwire::marshal::generated::-**
DestinationInfoMarshaller:

Public Member Functions

- **DestinationInfoMarshaller** ()
- virtual ~**DestinationInfoMarshaller** ()
- virtual **commands::DataStructure** * **createObject** () const
Creates a new instance of the class that this class is a marshaling director for.
- virtual unsigned char **getDataStructureType** () const
Gets the DataStructureType that this class marshals/unmarshals.
- virtual void **tightUnmarshal** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataInputStream** *dataIn, **utils::BooleanStream** *bs)
Tight Un-marhsal to the given stream.
- virtual int **tightMarshal1** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **utils::BooleanStream** *bs)
Tight Marshal to the given stream.
- virtual void **tightMarshal2** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataOutputStream** *dataOut, **utils::BooleanStream** *bs)
Tight Marshal to the given stream.
- virtual void **looseUnmarshal** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataInputStream** *dataIn)
Loose Un-marhsal to the given stream.

- virtual void **looseMarshal** (**OpenWireFormat** *wireFormat, **commands::Data-Structure** *dataStructure, **decaf::io::DataOutputStream** *dataOut)

Tight Marhsal to the given stream.

6.206.1 Detailed Description

Marshaling code for Open Wire Format for **DestinationInfoMarshaller** (p. 1218).

NOTE!: This file is auto generated - do not modify! if you need to make a change, please see the Java Classes in the activemq-openwire-generator module

6.206.2 Constructor & Destructor Documentation

6.206.2.1 **activemq::wireformat::openwire::marshal::generated::DestinationInfoMarshaller::DestinationInfoMarshaller** ()
[inline]

6.206.2.2 **virtual activemq::wireformat::openwire::marshal::generated::DestinationInfoMarshaller::~~DestinationInfoMarshaller** () [inline, virtual]

6.206.3 Member Function Documentation

6.206.3.1 **virtual commands::DataStructure* activemq::wireformat::openwire::marshal::generated::DestinationInfoMarshaller::createObject** () const
[virtual]

Creates a new instance of the class that this class is a marshaling director for.

Returns

newly allocated Command

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1120).

6.206.3.2 **virtual unsigned char activemq::wireformat::openwire::marshal::generated::DestinationInfoMarshaller::getDataStructureType** () const
[virtual]

Gets the DataStructureType that this class marshals/unmarshals.

Returns

byte Id of this classes DataStructureType

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1122).

6.206

activemq::wireformat::openwire::marshal::generated::DestinationInfoMarshaller
Class Reference 1223

6.206.3.3 virtual void **activemq::wireformat::openwire::marshal::generated::DestinationInfoMarshaller::looseMarshal** (**OpenWireFormat** * *format*,
commands::DataStructure * *command*, **decaf::io::DataOutputStream** * *ds*
) [virtual]

Tight Marhsal to the given stream.

Parameters

<i>format</i>	- The OpenWireFormat properties
<i>command</i>	- the object to Marshal
<i>ds</i>	- DataOutputStream to marshal to

Exceptions

<i>IOException</i>	if an error occurs.
--------------------	---------------------

Reimplemented from **activemq::wireformat::openwire::marshal::generated::Base-CommandMarshaller** (p. 500).

6.206.3.4 virtual void **activemq::wireformat::openwire::marshal::generated::DestinationInfoMarshaller::looseUnmarshal** (**OpenWireFormat** * *format*,
commands::DataStructure * *command*, **decaf::io::DataInputStream** * *dis*)
[virtual]

Loose Un-marhsal to the given stream.

Parameters

<i>format</i>	- The OpenWireFormat properties
<i>command</i>	- the object to Un-Marshal
<i>dis</i>	- the DataInputStream to Un-Marshal from

Exceptions

<i>IOException</i>	if an error occurs.
--------------------	---------------------

Reimplemented from **activemq::wireformat::openwire::marshal::generated::Base-CommandMarshaller** (p. 501).

6.206.3.5 virtual int **activemq::wireformat::openwire::marshal::generated::DestinationInfoMarshaller::tightMarshal1** (**OpenWireFormat** * *format*,
commands::DataStructure * *command*, **utils::BooleanStream** * *bs*)
[virtual]

Tight Marhsal to the given stream.

Parameters

<i>format</i>	The OpenwireFormat properties
<i>command</i>	The object to Marshal
<i>bs</i>	The boolean stream to marshal to.

Exceptions

<i>IOException</i>	if an error occurs.
--------------------	---------------------

Reimplemented from **activemq::wireformat::openwire::marshal::generated::BaseCommandMarshaller** (p. 503).

6.206.3.6 virtual void activemq::wireformat::openwire::marshal::generated::DestinationInfoMarshaller::tightMarshal2 (OpenWireFormat * *format*, commands::DataStructure * *command*, decaf::io::DataOutputStream * *ds*, utils::BooleanStream * *bs*) [virtual]

Tight Marshal to the given stream.

Parameters

<i>format</i>	- The OpenwireFormat properties
<i>command</i>	- the object to Marshal
<i>ds</i>	- the DataOutputStream to Marshal to
<i>bs</i>	- boolean stream to marshal to.

Exceptions

<i>IOException</i>	if an error occurs.
--------------------	---------------------

Reimplemented from **activemq::wireformat::openwire::marshal::generated::BaseCommandMarshaller** (p. 504).

6.206.3.7 virtual void activemq::wireformat::openwire::marshal::generated::DestinationInfoMarshaller::tightUnmarshal (OpenWireFormat * *format*, commands::DataStructure * *command*, decaf::io::DataInputStream * *dis*, utils::BooleanStream * *bs*) [virtual]

Tight Un-marshal to the given stream.

Parameters

<i>format</i>	- The OpenwireFormat properties
<i>command</i>	- the object to Un-Marshal
<i>dis</i>	- the DataInputStream to Un-Marshal from
<i>bs</i>	- boolean stream to unmarshal from.

Exceptions

<i>IOException</i>	if an error occurs.
--------------------	---------------------

Reimplemented from **activemq::wireformat::openwire::marshal::generated::BaseCommandMarshaller** (p. 505).

The documentation for this class was generated from the following file:

- src/main/activemq/wireformat/openwire/marshal/generated/**DestinationInfoMarshaller.h**

6.207 activemq::cmsutil::DestinationResolver Class Reference

Resolves a CMS destination name to a *Destination*.

```
#include <src/main/activemq/cmsutil/DestinationResolver.h>
```

Inheritance diagram for activemq::cmsutil::DestinationResolver:

Public Member Functions

- virtual **~DestinationResolver** () throw ()
- virtual void **init** (**ResourceLifecycleManager** *mgr)=0
Initializes this destination resolver for use.
- virtual void **destroy** ()=0
Destroys any allocated resources.
- virtual **cms::Destination** * **resolveDestinationName** (**cms::Session** *session, const std::string &destName, bool pubSubDomain)=0
Resolves the given name to a destination.

6.207.1 Detailed Description

Resolves a CMS destination name to a *Destination*.

6.207.2 Constructor & Destructor Documentation

6.207.2.1 virtual **activemq::cmsutil::DestinationResolver::~DestinationResolver** () throw () [inline, virtual]

6.207.3 Member Function Documentation

6.207.3.1 `virtual void activemq::cmsutil::DestinationResolver::destroy () [pure virtual]`

Destroys any allocated resources.

Implemented in **activemq::cmsutil::DynamicDestinationResolver** (p. 1273).

6.207.3.2 `virtual void activemq::cmsutil::DestinationResolver::init (ResourceLifecycleManager * mgr) [pure virtual]`

Initializes this destination resolver for use.

Ensures that any previously allocated resources are first destroyed (e.g. calls **destroy()** (p. 1223)).

Parameters

<i>mgr</i>	the resource lifecycle manager.
------------	---------------------------------

Implemented in **activemq::cmsutil::DynamicDestinationResolver** (p. 1273).

6.207.3.3 `virtual cms::Destination* activemq::cmsutil::DestinationResolver::resolveDestinationName (cms::Session * session, const std::string & destName, bool pubSubDomain) [pure virtual]`

Resolves the given name to a destination.

If `pubSubDomain` is true, a topic will be returned, otherwise a queue will be returned.

Parameters

<i>session</i>	the session for which to retrieve resolve the destination.
<i>destName</i>	the name to be resolved.
<i>pubSubDomain</i>	If true, the name will be resolved to a Topic, otherwise a Queue.

Returns

the resolved destination

Exceptions

<i>cms::CMS-Exception</i> (p. 826)	if resolution failed.
--	-----------------------

Implemented in **activemq::cmsutil::DynamicDestinationResolver** (p. 1273).

The documentation for this class was generated from the following file:

6.208 decaf::util::concurrent::ThreadPoolExecutor::CallerRunsPolicy::DiscardOldestPolicy Class

Reference

1227

- [src/main/activemq/cmsutil/DestinationResolver.h](#)

6.208 decaf::util::concurrent::ThreadPoolExecutor::CallerRunsPolicy::DiscardOldestPolicy Class Reference

Handler policy for tasks that are rejected upon a call to **ThreadPoolExecutor::execute** (p. 2724) this class always destroys the oldest unexecuted task in the **Queue** (p. 2222) and then attempts to execute the rejected task using the passed in executor.

```
#include <src/main/decaf/util/concurrent/ThreadPoolExecutor.h>
```

Inheritance diagram for decaf::util::concurrent::ThreadPoolExecutor::CallerRunsPolicy:-
:DiscardOldestPolicy:

Public Member Functions

- **DiscardOldestPolicy** ()
- virtual **~DiscardOldestPolicy** ()
- virtual void **rejectedExecution** (decaf::lang::Runnable *task, **ThreadPoolExecutor** *executor)

*Method that may be invoked by a **ThreadPoolExecutor** (p. 2715) when **execute** (p. 2724) cannot accept a task.*

6.208.1 Detailed Description

Handler policy for tasks that are rejected upon a call to **ThreadPoolExecutor::execute** (p. 2724) this class always destroys the oldest unexecuted task in the **Queue** (p. 2222) and then attempts to execute the rejected task using the passed in executor.

Since

1.0

6.208.2 Constructor & Destructor Documentation

6.208.2.1 **decaf::util::concurrent::ThreadPoolExecutor::CallerRunsPolicy::DiscardOldestPolicy::DiscardOldestPolicy** ()
[inline]

6.208.2.2 **virtual decaf::util::concurrent::ThreadPoolExecutor::CallerRunsPolicy::DiscardOldestPolicy::~~DiscardOldestPolicy** () [inline, virtual]

6.208.3 Member Function Documentation

6.208.3.1 `virtual void decaf::util::concurrent::ThreadPoolExecutor::CallerRunsPolicy::DiscardOldestPolicy::rejectedExecution (decaf::lang::Runnable * r, ThreadPoolExecutor * executor) [inline, virtual]`

Method that may be invoked by a **ThreadPoolExecutor** (p.2715) when **execute** (p.2724) cannot accept a task.

This may occur when no more threads or queue slots are available because their bounds would be exceeded, or upon shutdown of the **Executor** (p.1297).

In the absence of other alternatives, the method may throw an **RejectedExecutionException** (p.2263), which will be propagated to the caller of **execute** (p.2724).

Parameters

<i>r</i>	The pointer to the runnable task requested to be executed.
<i>executor</i>	The pointer to the executor attempting to execute this task.

Exceptions

RejectedExecutionException (p.2263)	if there is no remedy.
---	------------------------

Implements **decaf::util::concurrent::RejectedExecutionHandler** (p.2267).

References `decaf::util::concurrent::ThreadPoolExecutor::execute()`, `decaf::util::concurrent::ThreadPoolExecutor::getQueue()`, and `decaf::util::concurrent::ThreadPoolExecutor::isShutdown()`.

The documentation for this class was generated from the following file:

- `src/main/decaf/util/concurrent/ThreadPoolExecutor.h`

6.209 decaf::util::concurrent::ThreadPoolExecutor::CallerRunsPolicy::DiscardPolicy Class Reference

Handler policy for tasks that are rejected upon a call to **ThreadPoolExecutor::execute** (p.2724) this class always destroys the rejected task and returns quietly.

```
#include <src/main/decaf/util/concurrent/ThreadPoolExecutor.h>
```

Inheritance diagram for `decaf::util::concurrent::ThreadPoolExecutor::CallerRunsPolicy::DiscardPolicy`:

Public Member Functions

- **DiscardPolicy** ()
- virtual **~DiscardPolicy** ()
- virtual void **rejectedExecution** (decaf::lang::Runnable *task, ThreadPool-Executor *executer DECAF_UNUSED)

6.209.1 Detailed Description

Handler policy for tasks that are rejected upon a call to **ThreadPoolExecutor::execute** (p. 2724) this class always destroys the rejected task and returns quietly.

Since

1.0

6.209.2 Constructor & Destructor Documentation

6.209.2.1 **decaf::util::concurrent::ThreadPoolExecutor::CallerRunsPolicy::DiscardPolicy::DiscardPolicy** ()
[inline]

6.209.2.2 **virtual decaf::util::concurrent::ThreadPoolExecutor::CallerRunsPolicy::DiscardPolicy::~~DiscardPolicy** () [inline, virtual]

6.209.3 Member Function Documentation

6.209.3.1 **virtual void decaf::util::concurrent::ThreadPoolExecutor::CallerRunsPolicy::DiscardPolicy::rejectedExecution** (decaf::lang::Runnable * task, ThreadPoolExecutor *executer DECAF_UNUSED) [inline, virtual]

The documentation for this class was generated from the following file:

- src/main/decaf/util/concurrent/**ThreadPoolExecutor.h**

6.210 activemq::commands::DiscoveryEvent Class Reference

```
#include <src/main/activemq/commands/DiscoveryEvent.h>
```

Inheritance diagram for activemq::commands::DiscoveryEvent:

Public Member Functions

- **DiscoveryEvent** ()
- virtual **~DiscoveryEvent** ()
- virtual unsigned char **getDataStructureType** () const
*Get the **DataStructure** (p. 1133) Type as defined in CommandTypes.h.*
- virtual **DiscoveryEvent * cloneDataStructure** () const
Clone this object and return a new instance that the caller now owns, this will be an exact copy of this one.
- virtual void **copyDataStructure** (const **DataStructure** *src)
Copy the contents of the passed object into this objects members, overwriting any existing data.
- virtual std::string **toString** () const
*Returns a string containing the information for this **DataStructure** (p. 1133) such as its type and value of its elements.*
- virtual bool **equals** (const **DataStructure** *value) const
*Compares the **DataStructure** (p. 1133) passed in to this one, and returns if they are equivalent.*
- virtual const std::string & **getServiceName** () const
- virtual std::string & **getServiceName** ()
- virtual void **setServiceName** (const std::string &serviceName)
- virtual const std::string & **getBrokerName** () const
- virtual std::string & **getBrokerName** ()
- virtual void **setBrokerName** (const std::string &brokerName)

Static Public Attributes

- static const unsigned char **ID_DISCOVERYEVENT** = 40

Protected Attributes

- std::string **serviceName**
- std::string **brokerName**

6.210.1 Constructor & Destructor Documentation

6.210.1.1 **activemq::commands::DiscoveryEvent::DiscoveryEvent** ()

6.210.1.2 **virtual activemq::commands::DiscoveryEvent::~~DiscoveryEvent** ()
 [virtual]

6.210.2 Member Function Documentation

6.210.2.1 `virtual DiscoveryEvent* activemq::commands::DiscoveryEvent::cloneDataStructure () const`
[virtual]

Clone this object and return a new instance that the caller now owns, this will be an exact copy of this one.

Returns

new copy of this object.

Implements **activemq::commands::DataStructure** (p. 1133).

6.210.2.2 `virtual void activemq::commands::DiscoveryEvent::copyDataStructure (const DataStructure * src)` [virtual]

Copy the contents of the passed object into this objects members, overwriting any existing data.

Returns

src - Source Object

Implements **activemq::commands::DataStructure** (p. 1134).

6.210.2.3 `virtual bool activemq::commands::DiscoveryEvent::equals (const DataStructure * value) const` [virtual]

Compares the **DataStructure** (p. 1133) passed in to this one, and returns if they are equivalent.

Equivalent here means that they are of the same type, and that each element of the objects are the same.

Returns

true if **DataStructure** (p. 1133)'s are Equal.

Implements **activemq::commands::DataStructure** (p. 1135).

6.210.2.4 `virtual const std::string& activemq::commands::DiscoveryEvent::getBrokerName () const` [virtual]

6.210.2.5 `virtual std::string& activemq::commands::DiscoveryEvent::getBrokerName ()` [virtual]

6.210.2.6 `virtual unsigned char activemq::commands::DiscoveryEvent::getDataStructureType () const` [virtual]

Get the **DataStructure** (p. 1133) Type as defined in CommandTypes.h.

Returns

The type of the data structure

Implements **activemq::commands::DataStructure** (p. 1137).

6.210.2.7 `virtual const std::string& activemq::commands::DiscoveryEvent::getServiceName () const` [virtual]

6.210.2.8 `virtual std::string& activemq::commands::DiscoveryEvent::getServiceName ()` [virtual]

6.210.2.9 `virtual void activemq::commands::DiscoveryEvent::setBrokerName (const std::string & brokerName)` [virtual]

6.210.2.10 `virtual void activemq::commands::DiscoveryEvent::setServiceName (const std::string & serviceName)` [virtual]

6.210.2.11 `virtual std::string activemq::commands::DiscoveryEvent::toString () const` [virtual]

Returns a string containing the information for this **DataStructure** (p. 1133) such as its type and value of its elements.

Returns

formatted string useful for debugging.

Reimplemented from **activemq::commands::BaseDataStructure** (p. 531).

6.210.3 Field Documentation

6.210.3.1 `std::string activemq::commands::DiscoveryEvent::brokerName` [protected]

6.210.3.2 `const unsigned char activemq::commands::DiscoveryEvent::ID_DISCOVERYEVENT = 40` [static]

6.210.3.3 `std::string activemq::commands::DiscoveryEvent::serviceName` [protected]

The documentation for this class was generated from the following file:

- `src/main/activemq/commands/DiscoveryEvent.h`

6.211

activemq::wireformat::openwire::marshal::generated::DiscoveryEventMarshaller

Class Reference

1233

6.211 ~~activemq::wireformat::openwire::marshal::generated::-~~

DiscoveryEventMarshaller Class Reference

Marshaling code for Open Wire Format for **DiscoveryEventMarshaller** (p. 1230).

```
#include <src/main/activemq/wireformat/openwire/marshal/generated/-  
DiscoveryEventMarshaller.h>
```

Inheritance diagram for activemq::wireformat::openwire::marshal::generated::DiscoveryEventMarshaller:

Public Member Functions

- **DiscoveryEventMarshaller** ()
- virtual ~**DiscoveryEventMarshaller** ()
- virtual **commands::DataStructure** * **createObject** () const
Creates a new instance of the class that this class is a marshaling director for.
- virtual unsigned char **getDataStructureType** () const
Gets the DataStructureType that this class marshals/unmarshals.
- virtual void **tightUnmarshal** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataInputStream** *dataIn, **utils::BooleanStream** *bs)
Tight Un-marhsal to the given stream.
- virtual int **tightMarshal1** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **utils::BooleanStream** *bs)
Tight Marshal to the given stream.
- virtual void **tightMarshal2** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataOutputStream** *dataOut, **utils::BooleanStream** *bs)
Tight Marshal to the given stream.
- virtual void **looseUnmarshal** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataInputStream** *dataIn)
Loose Un-marhsal to the given stream.
- virtual void **looseMarshal** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataOutputStream** *dataOut)
Tight Marshal to the given stream.

6.211.1 Detailed Description

Marshaling code for Open Wire Format for **DiscoveryEventMarshaller** (p. 1230).

NOTE!: This file is auto generated - do not modify! if you need to make a change, please see the Java Classes in the activemq-openwire-generator module

6.211.2 Constructor & Destructor Documentation

6.211.2.1 **activemq::wireformat::openwire::marshal::generated::DiscoveryEventMarshaller::DiscoveryEventMarshaller ()**
[inline]

6.211.2.2 **virtual activemq::wireformat::openwire::marshal::generated::DiscoveryEventMarshaller::~~DiscoveryEventMarshaller ()** [inline, virtual]

6.211.3 Member Function Documentation

6.211.3.1 **virtual commands::DataStructure* activemq::wireformat::openwire::marshal::generated::DiscoveryEventMarshaller::createObject () const**
[virtual]

Creates a new instance of the class that this class is a marshaling director for.

Returns

newly allocated Command

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1120).

6.211.3.2 **virtual unsigned char activemq::wireformat::openwire::marshal::generated::DiscoveryEventMarshaller::getDataStructureType () const**
[virtual]

Gets the DataStructureType that this class marshals/unmarshals.

Returns

byte Id of this classes DataStructureType

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1122).

6.211.3.3 **virtual void activemq::wireformat::openwire::marshal::generated::DiscoveryEventMarshaller::looseMarshal (OpenWireFormat * *format*, commands::DataStructure * *command*, decaf::io::DataOutputStream * *ds*)** [virtual]

Tight Marshal to the given stream.

Parameters

<i>format</i>	- The OpenWireFormat properties
<i>command</i>	- the object to Marshal
<i>ds</i>	- DataOutputStream to marshal to

6.211

activemq::wireformat::openwire::marshal::generated::DiscoveryEventMarshaller

Class Reference

1235

Exceptions

<i>IOException</i>	if an error occurs.
--------------------	---------------------

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller**
(p. 1123).

6.211.3.4 `virtual void activemq::wireformat::openwire::marshal::generated::-
DiscoveryEventMarshaller::looseUnmarshal (OpenWireFormat * format,
commands::DataStructure * command, decaf::io::DataInputStream * dis)
[virtual]`

Loose Un-marhsal to the given stream.

Parameters

<i>format</i>	- The OpenWireFormat properties
<i>command</i>	- the object to Un-Marshal
<i>dis</i>	- the DataInputStream to Un-Marshal from

Exceptions

<i>IOException</i>	if an error occurs.
--------------------	---------------------

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller**
(p. 1125).

6.211.3.5 `virtual int activemq::wireformat::openwire::marshal::generated::-
DiscoveryEventMarshaller::tightMarshal1 (OpenWireFormat * format,
commands::DataStructure * command, utils::BooleanStream * bs)
[virtual]`

Tight Marhsal to the given stream.

Parameters

<i>format</i>	The OpenWireFormat properties
<i>command</i>	The object to Marshal
<i>bs</i>	The boolean stream to marshal to.

Exceptions

<i>IOException</i>	if an error occurs.
--------------------	---------------------

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller**
(p. 1127).

6.211.3.6 `virtual void activemq::wireformat::openwire::marshal::generated::DiscoveryEventMarshaller::tightMarshal2 (OpenWireFormat * format, commands::DataStructure * command, decaf::io::DataOutputStream * ds, utils::BooleanStream * bs) [virtual]`

Tight Marhsal to the given stream.

Parameters

<i>format</i>	- The OpenWireFormat properties
<i>command</i>	- the object to Marshal
<i>ds</i>	- the DataOutputStream to Marshal to
<i>bs</i>	- boolean stream to marshal to.

Exceptions

<i>IOException</i>	if an error occurs.
--------------------	---------------------

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1129).

6.211.3.7 `virtual void activemq::wireformat::openwire::marshal::generated::DiscoveryEventMarshaller::tightUnmarshal (OpenWireFormat * format, commands::DataStructure * command, decaf::io::DataInputStream * dis, utils::BooleanStream * bs) [virtual]`

Tight Un-marhsal to the given stream.

Parameters

<i>format</i>	- The OpenWireFormat properties
<i>command</i>	- the object to Un-Marshal
<i>dis</i>	- the DataInputStream to Un-Marshal from
<i>bs</i>	- boolean stream to unmarshal from.

Exceptions

<i>IOException</i>	if an error occurs.
--------------------	---------------------

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1131).

The documentation for this class was generated from the following file:

- `src/main/activemq/wireformat/openwire/marshal/generated/DiscoveryEvent-Marshaller.h`

6.212 `activemq::core::DispatchData` Class Reference

Simple POCO that contains the information necessary to route a message to a specified consumer.

```
#include <src/main/activemq/core/DispatchData.h>
```

Public Member Functions

- `DispatchData ()`
- `DispatchData (const decaf::lang::Pointer< commands::ConsumerId > &consumer, const decaf::lang::Pointer< commands::Message > &message)`
- `const decaf::lang::Pointer < commands::ConsumerId > &getConsumerId ()`
- `const decaf::lang::Pointer < commands::Message > &getMessage ()`

6.212.1 Detailed Description

Simple POCO that contains the information necessary to route a message to a specified consumer.

6.212.2 Constructor & Destructor Documentation

6.212.2.1 `activemq::core::DispatchData::DispatchData ()` `[inline]`

6.212.2.2 `activemq::core::DispatchData::DispatchData (const decaf::lang::Pointer< commands::ConsumerId > &consumer, const decaf::lang::Pointer< commands::Message > &message)` `[inline]`

6.212.3 Member Function Documentation

6.212.3.1 `const decaf::lang::Pointer<commands::ConsumerId>&activemq::core::DispatchData::getConsumerId ()` `[inline]`

6.212.3.2 `const decaf::lang::Pointer<commands::Message>&activemq::core::DispatchData::getMessage ()` `[inline]`

The documentation for this class was generated from the following file:

- `src/main/activemq/core/DispatchData.h`

6.213 `activemq::core::Dispatcher` Class Reference

Interface for an object responsible for dispatching messages to consumers.

```
#include <src/main/activemq/core/Dispatcher.h>
```

Inheritance diagram for `activemq::core::Dispatcher`:

Public Member Functions

- virtual `~Dispatcher()`
- virtual void **dispatch** (const **Pointer**< **MessageDispatch** > &message)=0
Dispatches a message to a particular consumer.

6.213.1 Detailed Description

Interface for an object responsible for dispatching messages to consumers.

6.213.2 Constructor & Destructor Documentation

6.213.2.1 `virtual activemq::core::Dispatcher::~~Dispatcher ()` [inline, virtual]

6.213.3 Member Function Documentation

6.213.3.1 `virtual void activemq::core::Dispatcher::dispatch (const Pointer< MessageDispatch > & message)` [pure virtual]

Dispatches a message to a particular consumer.

Parameters

<i>message</i>	The message to be dispatched to a waiting consumer.
----------------	---

Implemented in `activemq::core::ActiveMQSession` (p. 346), and `activemq::core::ActiveMQConsumer` (p. 238).

The documentation for this class was generated from the following file:

- `src/main/activemq/core/Dispatcher.h`

6.214 decaf::lang::Double Class Reference

```
#include <src/main/decaf/lang/Double.h>
```

Inheritance diagram for `decaf::lang::Double`:

Public Member Functions

- **Double** (double value)
*Constructs a new instance of a **Double** (p. 1235) object and assigns it the given value.*
- **Double** (const std::string &value)
*Constructs a new **Double** (p. 1235) and attempts to convert the given string to a double value, assigning it to the new object is successful or throwing a **NumberFormatException** if the string is not a properly formatted double.*
- virtual ~**Double** ()
- virtual int **compareTo** (const **Double** &d) const
*Compares this **Double** (p. 1235) instance with another.*
- bool **equals** (const **Double** &d) const
- virtual bool **operator==** (const **Double** &d) const
Compares equality between this object and the one passed.
- virtual bool **operator<** (const **Double** &d) const
Compares this object to another and returns true if this object is considered to be less than the one passed.
- virtual int **compareTo** (const double &d) const
*Compares this **Double** (p. 1235) instance with another.*
- bool **equals** (const double &d) const
- virtual bool **operator==** (const double &d) const
Compares equality between this object and the one passed.
- virtual bool **operator<** (const double &d) const
Compares this object to another and returns true if this object is considered to be less than the one passed.
- std::string **toString** () const
- virtual double **doubleValue** () const
Answers the double value which the receiver represents.
- virtual float **floatValue** () const
Answers the float value which the receiver represents.
- virtual unsigned char **byteValue** () const
Answers the byte value which the receiver represents.
- virtual short **shortValue** () const
Answers the short value which the receiver represents.
- virtual int **intValue** () const
Answers the int value which the receiver represents.
- virtual long long **longValue** () const
Answers the long value which the receiver represents.
- bool **isInfinite** () const
- bool **isNaN** () const

Static Public Member Functions

- static int **compare** (double d1, double d2)
Compares the two specified double values.
- static long long **doubleToLongBits** (double value)
Returns a representation of the specified floating-point value according to the IEEE 754 floating-point "double format" bit layout.
- static long long **doubleToRawLongBits** (double value)
Returns a representation of the specified floating-point value according to the IEEE 754 floating-point "double format" bit layout, preserving Not-a-Number (NaN) values.
- static bool **isInfinite** (double value)
- static bool **isNaN** (double value)
- static double **longBitsToDouble** (long long bits)
Returns the double value corresponding to a given bit representation.
- static double **parseDouble** (const std::string value)
*Returns a new double initialized to the value represented by the specified string, as performed by the `valueOf` method of class **Double** (p. 1235).*
- static std::string **toHexString** (double value)
Returns a hexadecimal string representation of the double argument.
- static std::string **toString** (double value)
Returns a string representation of the double argument.
- static **Double valueOf** (double value)
*Returns a **Double** (p. 1235) instance representing the specified double value.*
- static **Double valueOf** (const std::string &value)
*Returns a **Double** (p. 1235) instance that wraps a primitive double which is parsed from the string value passed.*

Static Public Attributes

- static const int **SIZE** = 64
The size in bits of the primitive int type.
- static const double **MAX_VALUE**
The maximum value that the primitive type can hold.
- static const double **MIN_VALUE**
The minimum value that the primitive type can hold.
- static const double **NaN**
*Constant for the Not a **Number** (p. 1992) Value.*
- static const double **POSITIVE_INFINITY**
Constant for Positive Infinity.
- static const double **NEGATIVE_INFINITY**
Constant for Negative Infinity.

6.214.1 Constructor & Destructor Documentation

6.214.1.1 decaf::lang::Double::Double (double *value*)

Constructs a new instance of a **Double** (p. 1235) object and assigns it the given value.

Parameters

<i>value</i>	The primitive type to wrap.
--------------	-----------------------------

6.214.1.2 decaf::lang::Double::Double (const std::string & *value*)

Constructs a new **Double** (p. 1235) and attempts to convert the given string to a double value, assigning it to the new object is successful or throwing a `NumberFormatException` if the string is not a properly formatted double.

Parameters

<i>value</i>	The string to convert to a primitive type to wrap.
--------------	--

Exceptions

<i>NumberFormatException</i>	if the string is not a a valid double.
------------------------------	--

6.214.1.3 virtual decaf::lang::Double::~~Double () [inline, virtual]

6.214.2 Member Function Documentation

6.214.2.1 virtual unsigned char decaf::lang::Double::byteValue () const [inline, virtual]

Answers the byte value which the receiver represents.

Returns

byte the value of the receiver.

Reimplemented from **decaf::lang::Number** (p. 1993).

6.214.2.2 static int decaf::lang::Double::compare (double *d1*, double *d2*) [static]

Compares the two specified double values.

The sign of the integer value returned is the same as that of the integer that would be returned by the call: `new Double(d1).compareTo(new Double(d2))`

Parameters

<i>d1</i>	- the first double to compare
<i>d2</i>	- the second double to compare

Returns

the value 0 if *d1* is numerically equal to *d2*; a value less than 0 if *d1* is numerically less than *d2*; and a value greater than 0 if *d1* is numerically greater than *d2*.

6.214.2.3 `virtual int decaf::lang::Double::compareTo (const Double & d) const`
[virtual]

Compares this **Double** (p. 1235) instance with another.

Parameters

<i>d</i>	- the Double (p. 1235) instance to be compared
----------	---

Returns

zero if this object represents the same integer value as the argument; a positive value if this object represents a value greater than the passed in value, and -1 if this object represents a value less than the passed in value.

Implements **decaf::lang::Comparable**< **Double** > (p. 886).

6.214.2.4 `virtual int decaf::lang::Double::compareTo (const double & d) const`
[virtual]

Compares this **Double** (p. 1235) instance with another.

Parameters

<i>d</i>	- the Double (p. 1235) instance to be compared
----------	---

Returns

zero if this object represents the same integer value as the argument; a positive value if this object represents a value greater than the passed in value, and -1 if this object represents a value less than the passed in value.

Implements **decaf::lang::Comparable**< **double** > (p. 886).

6.214.2.5 `static long long decaf::lang::Double::doubleToLongBits (double value)` `[static]`

Returns a representation of the specified floating-point value according to the IEEE 754 floating-point "double format" bit layout.

Bit 63 (the bit that is selected by the mask 0x8000000000000000L) represents the sign of the floating-point number. Bits 62-52 (the bits that are selected by the mask 0x7ff0000000000000L) represent the exponent. Bits 51-0 (the bits that are selected by the mask 0x000fffffffffffL) represent the significand (sometimes called the mantissa) of the floating-point number.

If the argument is positive infinity, the result is 0x7ff0000000000000L. If the argument is negative infinity, the result is 0xfff0000000000000L. If the argument is NaN, the result is 0x7ff8000000000000L.

In all cases, the result is a long integer that, when given to the `longBitsToDouble(long)` method, will produce a floating-point value the same as the argument to `doubleToLongBits` (except all NaN values are collapsed to a single "canonical" NaN value).

Parameters

<i>value</i>	- double to be converted
--------------	--------------------------

Returns

the long long bits that make up the double

6.214.2.6 `static long long decaf::lang::Double::doubleToRawLongBits (double value)` `[static]`

Returns a representation of the specified floating-point value according to the IEEE 754 floating-point "double format" bit layout, preserving Not-a-Number (NaN) values.

Bit 63 (the bit that is selected by the mask 0x8000000000000000LL) represents the sign of the floating-point number. Bits 62-52 (the bits that are selected by the mask 0x7ff0000000000000L) represent the exponent. Bits 51-0 (the bits that are selected by the mask 0x000fffffffffffL) represent the significand (sometimes called the mantissa) of the floating-point number.

If the argument is positive infinity, the result is 0x7ff0000000000000LL. If the argument is negative infinity, the result is 0xfff0000000000000LL. If the argument is NaN, the result is the long integer representing the actual NaN value. Unlike the `doubleToLongBits` method, `doubleToRawLongBits` does not collapse all the bit patterns encoding a NaN to a single "canonical" NaN value.

In all cases, the result is a long integer that, when given to the `longBitsToDouble(long)` method, will produce a floating-point value the same as the argument to `doubleToRawLongBits`.

Parameters

<i>value</i>	- double to be converted
--------------	--------------------------

Returns

the long long bits that make up the double

6.214.2.7 `virtual double decaf::lang::Double::doubleValue () const` `[inline, virtual]`

Answers the double value which the receiver represents.

Returns

double the value of the receiver.

Implements **decaf::lang::Number** (p. 1993).

6.214.2.8 `bool decaf::lang::Double::equals (const Double & d) const` `[inline, virtual]`

Parameters

<i>d</i>	- the Double (p. 1235) object to compare against.
----------	--

Returns

true if the two **Double** (p. 1235) Objects have the same value.

Implements **decaf::lang::Comparable< Double >** (p. 887).

6.214.2.9 `bool decaf::lang::Double::equals (const double & d) const` `[inline, virtual]`

Parameters

<i>d</i>	- the Double (p. 1235) object to compare against.
----------	--

Returns

true if the two **Double** (p. 1235) Objects have the same value.

Implements **decaf::lang::Comparable< double >** (p. 887).

6.214.2.10 `virtual float decaf::lang::Double::floatValue () const [inline, virtual]`

Answers the float value which the receiver represents.

Returns

float the value of the receiver.

Implements **decaf::lang::Number** (p. 1993).

6.214.2.11 `virtual int decaf::lang::Double::intValue () const [inline, virtual]`

Answers the int value which the receiver represents.

Returns

int the value of the receiver.

Implements **decaf::lang::Number** (p. 1993).

6.214.2.12 `bool decaf::lang::Double::isInfinite () const`

Returns

true if the double is equal to positive infinity.

6.214.2.13 `static bool decaf::lang::Double::isInfinite (double value) [static]`

Parameters

<i>value</i>	- The double to check.
--------------	------------------------

Returns

true if the double is equal to infinity.

6.214.2.14 `bool decaf::lang::Double::isNaN () const`

Returns

true if the double is equal to NaN.

6.214.2.15 `static bool decaf::lang::Double::isNaN (double value) [static]`

Parameters

<i>value</i>	- The double to check.
--------------	------------------------

Returns

true if the double is equal to NaN.

6.214.2.16 `static double decaf::lang::Double::longBitsToDouble (long long bits)`
`[static]`

Returns the double value corresponding to a given bit representation.

The argument is considered to be a representation of a floating-point value according to the IEEE 754 floating-point "double format" bit layout.

If the argument is 0x7ff0000000000000L, the result is positive infinity. If the argument is 0xfff0000000000000L, the result is negative infinity. If the argument is any value in the range 0x7ff0000000000001L through 0x7fffffffffffffffL or in the range 0xfff0000000000001L through 0xfffffffffffffffL, the result is a NaN. No IEEE 754 floating-point operation provided by C++ can distinguish between two NaN values of the same type with different bit patterns. Distinct values of NaN are only distinguishable by use of the **Double.doubleToRawLongBits** (p. 1240) method.

Parameters

<i>bits</i>	- the long long bits to convert to double
-------------	---

Returns

the double converted from the bits

6.214.2.17 `virtual long long decaf::lang::Double::longValue () const` `[inline, virtual]`

Answers the long value which the receiver represents.

Returns

long the value of the receiver.

Implements **decaf::lang::Number** (p. 1994).

6.214.2.18 `virtual bool decaf::lang::Double::operator< (const Double & d) const`
`[inline, virtual]`

Compares this object to another and returns true if this object is considered to be less than the one passed.

This

Parameters

<i>d</i>	- the value to be compared to this one.
----------	---

Returns

true if this object is equal to the one passed.

Implements **decaf::lang::Comparable**< **Double** > (p. 887).

6.214.2.19 `virtual bool decaf::lang::Double::operator< (const double & d) const`
`[inline, virtual]`

Compares this object to another and returns true if this object is considered to be less than the one passed.

This

Parameters

<i>d</i>	- the value to be compared to this one.
----------	---

Returns

true if this object is equal to the one passed.

Implements **decaf::lang::Comparable**< **double** > (p. 887).

6.214.2.20 `virtual bool decaf::lang::Double::operator== (const Double & d) const`
`[inline, virtual]`

Compares equality between this object and the one passed.

Parameters

<i>d</i>	- the value to be compared to this one.
----------	---

Returns

true if this object is equal to the one passed.

Implements **decaf::lang::Comparable**< **Double** > (p. 888).

6.214.2.21 `virtual bool decaf::lang::Double::operator== (const double & d) const`
`[inline, virtual]`

Compares equality between this object and the one passed.

Parameters

<i>d</i>	- the value to be compared to this one.
----------	---

Returns

true if this object is equal to the one passed.

Implements **decaf::lang::Comparable**< **double** > (p. 888).

6.214.2.22 `static double decaf::lang::Double::parseDouble (const std::string value)`
[static]

Returns a new double initialized to the value represented by the specified string, as performed by the `valueOf` method of class **Double** (p. 1235).

Parameters

<i>value</i>	- The string to parse to an double
--------------	------------------------------------

Returns

a double parsed from the passed string

Exceptions

<i>NumberFormatException</i>	
------------------------------	--

6.214.2.23 `virtual short decaf::lang::Double::shortValue () const` [inline, virtual]

Answers the short value which the receiver represents.

Returns

short the value of the receiver.

Reimplemented from **decaf::lang::Number** (p. 1994).

6.214.2.24 `static std::string decaf::lang::Double::toHexString (double value)`
[static]

Returns a hexadecimal string representation of the double argument.

All characters mentioned below are ASCII characters.

* If the argument is NaN, the result is the string "NaN". * Otherwise, the result is a string that represents the sign and magnitude (absolute value) of the argument. If the sign is negative, the first character of the result is '-'; if the sign is positive, no sign character appears in the result. As for the magnitude m: o If m is infinity, it is represented by the string "Infinity"; thus, positive infinity produces the result "Infinity" and negative infinity produces the result "-Infinity". o If m is zero, it is represented by the string "0x0.0p0"; thus, negative zero produces the result "-0x0.0p0" and positive zero produces the result "0x0.0p0". o If m is a double value with a normalized representation, substrings are used to represent the significand and exponent fields. The significand is represented by the characters "0x1." followed by a lowercase hexadecimal representation of the rest of the significand as a fraction. Trailing zeros in the hexadecimal representation are removed unless all the digits are zero, in which case a single zero is used. Next, the exponent is represented by "p" followed by a decimal string of the unbiased exponent as if produced by a call to **Integer.toString** (p. 1513) on the exponent value. o If m is a double value with a subnormal representation, the significand is represented by the characters "0x0." followed by a hexadecimal representation of the rest of the significand as a fraction. Trailing zeros in the hexadecimal representation are removed. Next, the exponent is represented by "p-126". Note that there must be at least one nonzero digit in a subnormal significand.

Parameters

<i>value</i>	- The double to convert to a string
--------------	-------------------------------------

Returns

the Hex formatted double string.

6.214.2.25 std::string decaf::lang::Double::toString () const

Returns

this **Double** (p. 1235) Object as a **String** (p. 2620) Representation

6.214.2.26 static std::string decaf::lang::Double::toString (double value) [static]

Returns a string representation of the double argument.

All characters mentioned below are ASCII characters.

If the argument is NaN, the result is the string "NaN". Otherwise, the result is a string that represents the sign and magnitude (absolute value) of the argument. If the sign is negative, the first character of the result is '-'; if the sign is positive, no sign character appears in the result. As for the magnitude m: o If m is infinity, it is represented by the characters "Infinity"; thus, positive infinity produces the result "Infinity" and negative infinity produces the result "-Infinity". o If m is zero, it is represented by the characters "0.0"; thus, negative zero produces the result "-0.0" and positive zero produces the result "0.0". o If m is greater than or equal to 10⁻³ but less than 10⁷, then it is represented as the integer part of m, in decimal form with no leading zeroes, followed by '.', followed

by one or more decimal digits representing the fractional part of m . o If m is less than 10^{-3} or greater than or equal to 10^7 , then it is represented in so-called "computerized scientific notation." Let n be the unique integer such that $10^n \leq m < 10^{n+1}$; then let a be the mathematically exact quotient of m and 10^n so that $1 \leq a < 10$. The magnitude is then represented as the integer part of a , as a single decimal digit, followed by '.', followed by decimal digits representing the fractional part of a , followed by the letter 'E', followed by a representation of n as a decimal integer, as produced by the method **Integer.toString(int)** (p. 1514).

Parameters

<i>value</i>	- The double to convert to a string
--------------	-------------------------------------

Returns

the formatted double string.

6.214.2.27 `static Double decaf::lang::Double::valueOf (double value)` [static]

Returns a **Double** (p. 1235) instance representing the specified double value.

Parameters

<i>value</i>	- double to wrap
--------------	------------------

Returns

new **Double** (p. 1235) instance wrapping the primitive value

6.214.2.28 `static Double decaf::lang::Double::valueOf (const std::string & value)`
[static]

Returns a **Double** (p. 1235) instance that wraps a primitive double which is parsed from the string value passed.

Parameters

<i>value</i>	- the string to parse
--------------	-----------------------

Returns

a new **Double** (p. 1235) instance wrapping the double parsed from value

Exceptions

<i>NumberFormatException</i>	on error.
------------------------------	-----------

6.214.3 Field Documentation

6.214.3.1 `const double decaf::lang::Double::MAX_VALUE` `[static]`

The maximum value that the primitive type can hold.

6.214.3.2 `const double decaf::lang::Double::MIN_VALUE` `[static]`

The minimum value that the primitive type can hold.

6.214.3.3 `const double decaf::lang::Double::NaN` `[static]`

Constant for the Not a **Number** (p. 1992) Value.

6.214.3.4 `const double decaf::lang::Double::NEGATIVE_INFINITY` `[static]`

Constant for Negative Infinity.

6.214.3.5 `const double decaf::lang::Double::POSITIVE_INFINITY` `[static]`

Constant for Positive Infinity.

6.214.3.6 `const int decaf::lang::Double::SIZE = 64` `[static]`

The size in bits of the primitive int type.

The documentation for this class was generated from the following file:

- `src/main/decaf/lang/Double.h`

6.215 decaf::internal::nio::DoubleArrayBuffer Class Reference

```
#include <src/main/decaf/internal/nio/DoubleArrayBuffer.h>
```

Inheritance diagram for decaf::internal::nio::DoubleArrayBuffer:

Public Member Functions

- **DoubleArrayBuffer** (int **capacity**, bool **readOnly**=false)

*Creates a **DoubleArrayBuffer** (p. 1248) object that has its backing array allocated internally and is then owned and deleted when this object is deleted.*

- **DoubleArrayBuffer** (double *array, int size, int offset, int length, bool read-Only=false)

*Creates a **DoubleArrayBuffer** (p. 1248) object that wraps the given array.*

- **DoubleArrayBuffer** (const decaf::lang::Pointer< **ByteArrayAdapter** > &array, int offset, int length, bool readOnly=false)

Creates a byte buffer that wraps the passed ByteArrayAdapter and start at the given offset.

- **DoubleArrayBuffer** (const **DoubleArrayBuffer** &other)

*Create a **DoubleArrayBuffer** (p. 1248) that mirrors this one, meaning it shares a reference to this buffers ByteArrayAdapter and when changes are made to that data it is reflected in both.*

- virtual ~**DoubleArrayBuffer** ()
- virtual double * array ()

Returns the double array that backs this buffer (optional operation).

Modifications to this buffer's content will cause the returned array's content to be modified, and vice versa.

Invoke the hasArray method before invoking this method in order to ensure that this buffer has an accessible backing array.

Returns

*the array that backs this **Buffer** (p. 582).*

Exceptions

ReadOnlyBuffer-Exception (p. 2244)	<i>if this Buffer (p. 582) is read only.</i>
UnsupportedOperation-Exception	<i>if the underlying store has no array.</i>

- virtual int arrayOffset ()

Returns the offset within this buffer's backing array of the first element of the buffer (optional operation).

Invoke the hasArray method before invoking this method in order to ensure that this buffer has an accessible backing array.

Returns

The offset into the backing array where index zero starts.

Exceptions

ReadOnlyBuffer-Exception (p. 2244)	<i>if this Buffer (p. 582) is read only.</i>
UnsupportedOperation-Exception	<i>if the underlying store has no array.</i>

- virtual **DoubleBuffer** * asReadOnlyBuffer () const

Creates a new, read-only double buffer that shares this buffer's content.

The content of the new buffer will be that of this buffer. Changes to this buffer's content will be visible in the new buffer; the new buffer itself, however, will be read-only and will not allow the shared content to be modified. The two buffers' position, limit, and mark values will be independent.

If this buffer is itself read-only then this method behaves in exactly the same way as the duplicate method.

The new buffer's capacity, limit, position, and mark values will be identical to those of this buffer.

Returns

The new, read-only double buffer which the caller then owns.

- virtual **DoubleBuffer** & **compact** ()

Compacts this buffer.

The bytes between the buffer's current position and its limit, if any, are copied to the beginning of the buffer. That is, the byte at index $p = \text{position}()$ (p. 587) is copied to index zero, the byte at index $p + 1$ is copied to index one, and so forth until the byte at index $\text{limit}()$ (p. 586) - 1 is copied to index $n = \text{limit}()$ (p. 586) - 1 - p . The buffer's position is then set to $n+1$ and its limit is set to its capacity. The mark, if defined, is discarded.

The buffer's position is set to the number of bytes copied, rather than to zero, so that an invocation of this method can be followed immediately by an invocation of another relative put method.

Returns

a reference to this **DoubleBuffer** (p. 1259).

Exceptions

ReadOnlyBuffer-Exception (p. 2244)	if this buffer is read-only.
--	------------------------------

- virtual **DoubleBuffer** * **duplicate** ()

Creates a new double buffer that shares this buffer's content.

The content of the new buffer will be that of this buffer. Changes to this buffer's content will be visible in the new buffer, and vice versa; the two buffers' position, limit, and mark values will be independent.

The new buffer's capacity, limit, position, and mark values will be identical to those of this buffer. The new buffer will be read-only if, and only if, this buffer is read-only.

Returns

a new double **Buffer** (p. 582) which the caller owns.

- virtual double **get** ()

Relative get method.

Reads the value at this buffer's current position, and then increments the position.

Returns

the double at the current position.

Exceptions

BufferUnderflow-Exception (p. 611)	if there no more data to return.
--	----------------------------------

- virtual double **get** (int index) const

Absolute get method.

Reads the value at the given index.

Parameters

index	The index in the Buffer (p. 582) where the double is to be read.
-------	---

Returns

the double that is located at the given index.

Exceptions

IndexOutOfBoundsException	<i>if index is not smaller than the buffer's limit</i>
---------------------------	--

- virtual bool **hasArray** () const

Tells whether or not this buffer is backed by an accessible double array.

If this method returns true then the array and arrayOffset methods may safely be invoked. Subclasses should override this method if they do not have a backing array as this class always returns true.

Returns

true if, and only if, this buffer is backed by an array and is not read-only

- virtual bool **isReadOnly** () const

Tells whether or not this buffer is read-only.

Returns

true if, and only if, this buffer is read-only.

- virtual **DoubleBuffer** & **put** (double value)

Writes the given doubles into this buffer at the current position, and then increments the position.

Parameters

value	<i>The doubles value to be written.</i>
-------	---

Returns

a reference to this buffer.

Exceptions

BufferOverflowException (p. 609)	<i>if this buffer's current position is not smaller than its limit.</i>
ReadOnlyBufferException (p. 2244)	<i>if this buffer is read-only.</i>

- virtual **DoubleBuffer** & **put** (int index, double value)

Writes the given doubles into this buffer at the given index.

Parameters

index	<i>The position in the Buffer (p. 582) to write the data.</i>
value	<i>The doubles to write.</i>

Returns

a reference to this buffer

Exceptions

IndexOutOfBoundsException	<i>if index greater than the buffer's limit minus the size of the type being written, or the index is negative.</i>
ReadOnlyBufferException (p. 2244)	<i>if this buffer is read-only.</i>

- virtual **DoubleBuffer** * **slice** () const

Creates a new **DoubleBuffer** (p. 1259) whose content is a shared subsequence of this buffer's content.

The content of the new buffer will start at this buffer's current position. Changes to this buffer's content will be visible in the new buffer, and vice versa; the two buffers' position, limit, and mark values will be independent.

The new buffer's position will be zero, its capacity and its limit will be the number of bytes remaining in this buffer, and its mark will be undefined. The new buffer will be read-only if, and only if, this buffer is read-only.

Returns

the newly create **DoubleBuffer** (p. 1259) which the caller owns.

Protected Member Functions

- virtual void **setReadOnly** (bool value)

Sets this **DoubleArrayBuffer** (p. 1248) as Read-Only or not Read-Only.

6.215.1 Constructor & Destructor Documentation

6.215.1.1 decaf::internal::nio::DoubleArrayBuffer::DoubleArrayBuffer (int capacity, bool readOnly = false)

Creates a **DoubleArrayBuffer** (p. 1248) object that has its backing array allocated internally and is then owned and deleted when this object is deleted.

The array is initially created with all elements initialized to zero.

Parameters

<i>size</i>	The size of the array, this is the limit we read and write to.
<i>readOnly</i>	Boolean indicating if this buffer should be read-only, default as false.

Exceptions

<i>IllegalArgument-Exception</i>	if the capacity value is negative.
----------------------------------	------------------------------------

6.215.1.2 decaf::internal::nio::DoubleArrayBuffer::DoubleArrayBuffer (double * array, int size, int offset, int length, bool readOnly = false)

Creates a **DoubleArrayBuffer** (p. 1248) object that wraps the given array.

If the own flag is set then it will delete this array when this object is deleted.

Parameters

<i>array</i>	The actual array to wrap.
<i>size</i>	The size of the given array.

<i>offset</i>	The position that is this buffers start position.
<i>length</i>	The limit of how many bytes into the array this Buffer can write.
<i>readOnly</i>	Boolean indicating if this buffer should be read-only, default as false.

Exceptions

<i>NullPointerException</i>	if buffer is NULL
<i>IndexOutOfBoundsException</i>	if offset is greater than array capacity.

6.215.1.3 **decaf::internal::nio::DoubleArrayBuffer::DoubleArrayBuffer** (**const** **decaf::lang::Pointer**< **ByteArrayAdapter** > & *array*, **int** *offset*, **int** *length*, **bool** *readOnly* = *false*)

Creates a byte buffer that wraps the passed **ByteArrayAdapter** and start at the given offset.

The capacity and limit of the new **DoubleArrayBuffer** (p. 1248) will be that of the remaining capacity of the passed buffer.

Parameters

<i>array</i>	The ByteArrayAdapter to wrap.
<i>offset</i>	The position that is this buffers start position.
<i>length</i>	The limit of how many bytes into the array this Buffer can write.
<i>readOnly</i>	Boolean indicating if this buffer should be read-only, default as false.

Exceptions

<i>NullPointerException</i>	if array is NULL
<i>IndexOutOfBoundsException</i>	if offset + length is greater than array size.

6.215.1.4 **decaf::internal::nio::DoubleArrayBuffer::DoubleArrayBuffer** (**const** **DoubleArrayBuffer** & *other*)

Create a **DoubleArrayBuffer** (p. 1248) that mirrors this one, meaning it shares a reference to this buffers **ByteArrayAdapter** and when changes are made to that data it is reflected in both.

Parameters

<i>other</i>	The DoubleArrayBuffer (p. 1248) this one is to mirror.
--------------	---

6.215.1.5 virtual **decaf::internal::nio::DoubleArrayBuffer::~~DoubleArrayBuffer** ()
[virtual]

6.215.2 Member Function Documentation

6.215.2.1 virtual **double*** **decaf::internal::nio::DoubleArrayBuffer::array** ()
[virtual]

Returns the double array that backs this buffer (optional operation).

Modifications to this buffer's content will cause the returned array's content to be modified, and vice versa.

Invoke the `hasArray` method before invoking this method in order to ensure that this buffer has an accessible backing array.

Returns

the array that backs this **Buffer** (p. 582).

Exceptions

<i>ReadOnlyBuffer-Exception</i> (p. 2244)	if this Buffer (p. 582) is read only.
<i>UnsupportedOperationException</i>	if the underlying store has no array.

Implements **decaf::nio::DoubleBuffer** (p. 1262).

6.215.2.2 virtual **int** **decaf::internal::nio::DoubleArrayBuffer::arrayOffset** ()
[virtual]

Returns the offset within this buffer's backing array of the first element of the buffer (optional operation).

Invoke the `hasArray` method before invoking this method in order to ensure that this buffer has an accessible backing array.

Returns

The offset into the backing array where index zero starts.

Exceptions

<i>ReadOnlyBuffer-Exception</i> (p. 2244)	if this Buffer (p. 582) is read only.
<i>UnsupportedOperationException</i>	if the underlying store has no array.

Implements **decaf::nio::DoubleBuffer** (p. 1263).

6.215.2.3 **virtual DoubleBuffer* decaf::internal::nio::DoubleArrayBuffer::asRead-OnlyBuffer () const** [virtual]

Creates a new, read-only double buffer that shares this buffer's content.

The content of the new buffer will be that of this buffer. Changes to this buffer's content will be visible in the new buffer; the new buffer itself, however, will be read-only and will not allow the shared content to be modified. The two buffers' position, limit, and mark values will be independent.

If this buffer is itself read-only then this method behaves in exactly the same way as the duplicate method.

The new buffer's capacity, limit, position, and mark values will be identical to those of this buffer.

Returns

The new, read-only double buffer which the caller then owns.

Implements **decaf::nio::DoubleBuffer** (p. 1263).

6.215.2.4 **virtual DoubleBuffer& decaf::internal::nio::DoubleArrayBuffer::compact ()** [virtual]

Compacts this buffer.

The bytes between the buffer's current position and its limit, if any, are copied to the beginning of the buffer. That is, the byte at index $p = \text{position}()$ (p. 587) is copied to index zero, the byte at index $p + 1$ is copied to index one, and so forth until the byte at index $\text{limit}()$ (p. 586) - 1 is copied to index $n = \text{limit}()$ (p. 586) - 1 - p . The buffer's position is then set to $n+1$ and its limit is set to its capacity. The mark, if defined, is discarded.

The buffer's position is set to the number of bytes copied, rather than to zero, so that an invocation of this method can be followed immediately by an invocation of another relative put method.

Returns

a reference to this **DoubleBuffer** (p. 1259).

Exceptions

ReadOnlyBuffer-Exception (p. 2244)	if this buffer is read-only.
--	------------------------------

Implements **decaf::nio::DoubleBuffer** (p. 1263).

6.215.2.5 virtual **DoubleBuffer*** decaf::internal::nio::DoubleArrayBuffer::duplicate
() [virtual]

Creates a new double buffer that shares this buffer's content.

The content of the new buffer will be that of this buffer. Changes to this buffer's content will be visible in the new buffer, and vice versa; the two buffers' position, limit, and mark values will be independent.

The new buffer's capacity, limit, position, and mark values will be identical to those of this buffer. The new buffer will be read-only if, and only if, this buffer is read-only.

Returns

a new double **Buffer** (p. 582) which the caller owns.

Implements **decaf::nio::DoubleBuffer** (p. 1264).

6.215.2.6 virtual double decaf::internal::nio::DoubleArrayBuffer::get ()
[virtual]

Relative get method.

Reads the value at this buffer's current position, and then increments the position.

Returns

the double at the current position.

Exceptions

BufferUnderflow-Exception (p. 611)	if there no more data to return.
--	----------------------------------

Implements **decaf::nio::DoubleBuffer** (p. 1264).

6.215.2.7 virtual double decaf::internal::nio::DoubleArrayBuffer::get (int *index*) const
[virtual]

Absolute get method.

Reads the value at the given index.

Parameters

<i>index</i>	The index in the Buffer (p. 582) where the double is to be read.
--------------	---

Returns

the double that is located at the given index.

Exceptions

<i>IndexOutOfBoundsException</i>	if index is not smaller than the buffer's limit
----------------------------------	---

Implements **decaf::nio::DoubleBuffer** (p. 1265).

6.215.2.8 `virtual bool decaf::internal::nio::DoubleArrayBuffer::hasArray () const`
`[inline, virtual]`

Tells whether or not this buffer is backed by an accessible double array.

If this method returns true then the array and arrayOffset methods may safely be invoked. Subclasses should override this method if they do not have a backing array as this class always returns true.

Returns

true if, and only if, this buffer is backed by an array and is not read-only

Implements **decaf::nio::DoubleBuffer** (p. 1266).

6.215.2.9 `virtual bool decaf::internal::nio::DoubleArrayBuffer::isReadOnly () const`
`[inline, virtual]`

Tells whether or not this buffer is read-only.

Returns

true if, and only if, this buffer is read-only.

Implements **decaf::nio::Buffer** (p. 586).

6.215.2.10 `virtual DoubleBuffer& decaf::internal::nio::DoubleArrayBuffer::put (`
`double value) [virtual]`

Writes the given doubles into this buffer at the current position, and then increments the position.

Parameters

<i>value</i>	The doubles value to be written.
--------------	----------------------------------

Returns

a reference to this buffer.

Exceptions

BufferOverflow-Exception (p. 609)	if this buffer's current position is not smaller than its limit.
ReadOnlyBuffer-Exception (p. 2244)	if this buffer is read-only.

Implements **decaf::nio::DoubleBuffer** (p. 1269).

6.215.2.11 virtual **DoubleBuffer& decaf::internal::nio::DoubleArrayBuffer::put** (int *index*, double *value*) [virtual]

Writes the given doubles into this buffer at the given index.

Parameters

<i>index</i>	The position in the Buffer (p. 582) to write the data.
<i>value</i>	The doubles to write.

Returns

a reference to this buffer

Exceptions

IndexOutOfBoundsException	if index greater than the buffer's limit minus the size of the type being written, or the index is negative.
ReadOnlyBuffer-Exception (p. 2244)	if this buffer is read-only.

Implements **decaf::nio::DoubleBuffer** (p. 1269).

6.215.2.12 virtual void **decaf::internal::nio::DoubleArrayBuffer::setReadOnly** (bool *value*) [inline, protected, virtual]

Sets this **DoubleArrayBuffer** (p. 1248) as Read-Only or not Read-Only.

Parameters

<i>value</i>	Boolean value, true if this buffer is to be read-only, false otherwise.
--------------	---

6.215.2.13 `virtual DoubleBuffer* decaf::internal::nio::DoubleArrayBuffer::slice ()`
`const [virtual]`

Creates a new **DoubleBuffer** (p. 1259) whose content is a shared subsequence of this buffer's content.

The content of the new buffer will start at this buffer's current position. Changes to this buffer's content will be visible in the new buffer, and vice versa; the two buffers' position, limit, and mark values will be independent.

The new buffer's position will be zero, its capacity and its limit will be the number of bytes remaining in this buffer, and its mark will be undefined. The new buffer will be read-only if, and only if, this buffer is read-only.

Returns

the newly create **DoubleBuffer** (p. 1259) which the caller owns.

Implements **decaf::nio::DoubleBuffer** (p. 1270).

The documentation for this class was generated from the following file:

- `src/main/decaf/internal/nio/DoubleArrayBuffer.h`

6.216 decaf::nio::DoubleBuffer Class Reference

This class defines four categories of operations upon double buffers:

```
#include <src/main/decaf/nio/DoubleBuffer.h>
```

Inheritance diagram for `decaf::nio::DoubleBuffer`:

Public Member Functions

- `virtual ~DoubleBuffer ()`
- `virtual std::string toString () const`
- `virtual double * array ()=0`
Returns the double array that backs this buffer (optional operation).
- `virtual int arrayOffset ()=0`
Returns the offset within this buffer's backing array of the first element of the buffer (optional operation).
- `virtual DoubleBuffer * asReadOnlyBuffer () const =0`
Creates a new, read-only double buffer that shares this buffer's content.
- `virtual DoubleBuffer & compact ()=0`
Compacts this buffer.
- `virtual DoubleBuffer * duplicate ()=0`

Creates a new double buffer that shares this buffer's content.

- virtual double **get** ()=0
Relative get method.
- virtual double **get** (int index) const =0
Absolute get method.
- **DoubleBuffer & get** (std::vector< double > buffer)
Relative bulk get method.
- **DoubleBuffer & get** (double *buffer, int size, int offset, int length)
Relative bulk get method.
- virtual bool **hasArray** () const =0
Tells whether or not this buffer is backed by an accessible double array.
- **DoubleBuffer & put** (**DoubleBuffer** &src)
This method transfers the doubles remaining in the given source buffer into this buffer.
- **DoubleBuffer & put** (const double *buffer, int size, int offset, int length)
This method transfers doubles into this buffer from the given source array.
- **DoubleBuffer & put** (std::vector< double > &buffer)
This method transfers the entire content of the given source doubles array into this buffer.
- virtual **DoubleBuffer & put** (double value)=0
Writes the given doubles into this buffer at the current position, and then increments the position.
- virtual **DoubleBuffer & put** (int index, double value)=0
Writes the given doubles into this buffer at the given index.
- virtual **DoubleBuffer * slice** () const =0
*Creates a new **DoubleBuffer** (p. 1259) whose content is a shared subsequence of this buffer's content.*
- virtual int **compareTo** (const **DoubleBuffer** &value) const
- virtual bool **equals** (const **DoubleBuffer** &value) const
- virtual bool **operator==** (const **DoubleBuffer** &value) const
- virtual bool **operator<** (const **DoubleBuffer** &value) const

Static Public Member Functions

- static **DoubleBuffer * allocate** (int capacity)
*Allocates a new **DoubleBuffer** (p. 1259).*
- static **DoubleBuffer * wrap** (double *array, int size, int offset, int length)
*Wraps the passed buffer with a new **DoubleBuffer** (p. 1259).*
- static **DoubleBuffer * wrap** (std::vector< double > &buffer)
*Wraps the passed STL double Vector in a **DoubleBuffer** (p. 1259).*

Protected Member Functions

- **DoubleBuffer** (int *capacity*)

*Creates a **DoubleBuffer** (p. 1259) object that has its backing array allocated internally and is then owned and deleted when this object is deleted.*

6.216.1 Detailed Description

This class defines four categories of operations upon double buffers:

- o Absolute and relative get and put methods that read and write single doubles;
- o Relative bulk get methods that transfer contiguous sequences of doubles from this buffer into an array;
- and o Relative bulk put methods that transfer contiguous sequences of doubles from a double array or some other double buffer into this buffer
- o Methods for compacting, duplicating, and slicing a double buffer.

Double buffers can be created either by allocation, which allocates space for the buffer's content, by wrapping an existing double array into a buffer, or by creating a view of an existing byte buffer

Methods in this class that do not otherwise have a value to return are specified to return the buffer upon which they are invoked. This allows method invocations to be chained.

Since

1.0

6.216.2 Constructor & Destructor Documentation

6.216.2.1 `decaf::nio::DoubleBuffer::DoubleBuffer (int capacity)` [protected]

Creates a **DoubleBuffer** (p. 1259) object that has its backing array allocated internally and is then owned and deleted when this object is deleted.

The array is initially created with all elements initialized to zero.

Parameters

<i>capacity</i>	The size and limit of the Buffer (p. 582) in doubles
-----------------	---

Exceptions

<i>IllegalArgument-Exception</i>	if capacity is negative.
----------------------------------	--------------------------

6.216.2.2 `virtual decaf::nio::DoubleBuffer::~DoubleBuffer ()` [inline, virtual]

6.216.3 Member Function Documentation

6.216.3.1 **static DoubleBuffer*** **decaf::nio::DoubleBuffer::allocate** (int *capacity*)
[static]

Allocates a new **DoubleBuffer** (p. 1259).

The new buffer's position will be zero, its limit will be its capacity, and its mark will be undefined. It will have a backing array, and its array offset will be zero.

Parameters

<i>capacity</i>	The size of the Double buffer in doubles.
-----------------	---

Returns

the **DoubleBuffer** (p. 1259) that was allocated, caller owns.

Exceptions

<i>IllegalArgumentException</i>	is the capacity value is negative.
---------------------------------	------------------------------------

6.216.3.2 **virtual double*** **decaf::nio::DoubleBuffer::array** () [pure virtual]

Returns the double array that backs this buffer (optional operation).

Modifications to this buffer's content will cause the returned array's content to be modified, and vice versa.

Invoke the `hasArray` method before invoking this method in order to ensure that this buffer has an accessible backing array.

Returns

the array that backs this **Buffer** (p. 582).

Exceptions

<i>ReadOnlyBufferException</i> (p. 2244)	if this Buffer (p. 582) is read only.
<i>UnsupportedOperationException</i>	if the underlying store has no array.

Implemented in **decaf::internal::nio::DoubleArrayBuffer** (p. 1254).

6.216.3.3 `virtual int decaf::nio::DoubleBuffer::arrayOffset ()` [pure virtual]

Returns the offset within this buffer's backing array of the first element of the buffer (optional operation).

Invoke the `hasArray` method before invoking this method in order to ensure that this buffer has an accessible backing array.

Returns

The offset into the backing array where index zero starts.

Exceptions

<i>ReadOnlyBufferException</i> (p. 2244)	if this Buffer (p. 582) is read only.
<i>UnsupportedOperationException</i>	if the underlying store has no array.

Implemented in **decaf::internal::nio::DoubleArrayBuffer** (p. 1254).

6.216.3.4 `virtual DoubleBuffer* decaf::nio::DoubleBuffer::asReadOnlyBuffer ()` `const` [pure virtual]

Creates a new, read-only double buffer that shares this buffer's content.

The content of the new buffer will be that of this buffer. Changes to this buffer's content will be visible in the new buffer; the new buffer itself, however, will be read-only and will not allow the shared content to be modified. The two buffers' position, limit, and mark values will be independent.

If this buffer is itself read-only then this method behaves in exactly the same way as the `duplicate` method.

The new buffer's capacity, limit, position, and mark values will be identical to those of this buffer.

Returns

The new, read-only double buffer which the caller then owns.

Implemented in **decaf::internal::nio::DoubleArrayBuffer** (p. 1255).

6.216.3.5 `virtual DoubleBuffer& decaf::nio::DoubleBuffer::compact ()` [pure virtual]

Compacts this buffer.

The bytes between the buffer's current position and its limit, if any, are copied to the beginning of the buffer. That is, the byte at index `p = position()` (p. 587) is copied to

index zero, the byte at index $p + 1$ is copied to index one, and so forth until the byte at index **limit()** (p. 586) - 1 is copied to index $n = \text{limit}() - 1 - p$. The buffer's position is then set to $n+1$ and its limit is set to its capacity. The mark, if defined, is discarded.

The buffer's position is set to the number of bytes copied, rather than to zero, so that an invocation of this method can be followed immediately by an invocation of another relative put method.

Returns

a reference to this **DoubleBuffer** (p. 1259).

Exceptions

ReadOnlyBuffer-Exception (p. 2244)	if this buffer is read-only.
--	------------------------------

Implemented in **decaf::internal::nio::DoubleArrayBuffer** (p. 1255).

6.216.3.6 `virtual int decaf::nio::DoubleBuffer::compareTo (const DoubleBuffer & value) const` [virtual]

6.216.3.7 `virtual DoubleBuffer* decaf::nio::DoubleBuffer::duplicate ()` [pure virtual]

Creates a new double buffer that shares this buffer's content.

The content of the new buffer will be that of this buffer. Changes to this buffer's content will be visible in the new buffer, and vice versa; the two buffers' position, limit, and mark values will be independent.

The new buffer's capacity, limit, position, and mark values will be identical to those of this buffer. The new buffer will be read-only if, and only if, this buffer is read-only.

Returns

a new double **Buffer** (p. 582) which the caller owns.

Implemented in **decaf::internal::nio::DoubleArrayBuffer** (p. 1256).

6.216.3.8 `virtual bool decaf::nio::DoubleBuffer::equals (const DoubleBuffer & value) const` [virtual]

6.216.3.9 `virtual double decaf::nio::DoubleBuffer::get ()` [pure virtual]

Relative get method.

Reads the value at this buffer's current position, and then increments the position.

Returns

the double at the current position.

Exceptions

<i>BufferUnderflow-Exception</i> (p. 611)	if there no more data to return.
---	----------------------------------

Implemented in **decaf::internal::nio::DoubleArrayBuffer** (p. 1256).

6.216.3.10 `virtual double decaf::nio::DoubleBuffer::get (int index) const` [pure virtual]

Absolute get method.

Reads the value at the given index.

Parameters

<i>index</i>	The index in the Buffer (p. 582) where the double is to be read.
--------------	---

Returns

the double that is located at the given index.

Exceptions

<i>IndexOutOfBoundsException</i>	if index is not smaller than the buffer's limit
---	---

Implemented in **decaf::internal::nio::DoubleArrayBuffer** (p. 1256).

6.216.3.11 `DoubleBuffer& decaf::nio::DoubleBuffer::get (std::vector< double > buffer)`

Relative bulk get method.

This method transfers values from this buffer into the given destination vector. An invocation of this method of the form `src.get(a)` behaves in exactly the same way as the invocation. The vector must be sized to the amount of data that is to be read, that is to say, the caller should call `buffer.resize(N)` before calling this get method.

Returns

a reference to this **Buffer** (p. 582).

Exceptions

<i>BufferUnderflow-Exception</i> (p. 611)	if there are fewer than length doubles remaining in this buffer
---	---

6.216.3.12 DoubleBuffer& decaf::nio::DoubleBuffer::get (double * *buffer*, int *size*, int *offset*, int *length*)

Relative bulk get method.

This method transfers doubles from this buffer into the given destination array. If there are fewer doubles remaining in the buffer than are required to satisfy the request, that is, if `length > remaining()` (p. 588), then no bytes are transferred and a **BufferUnderflow-Exception** (p. 611) is thrown.

Otherwise, this method copies length doubles from this buffer into the given array, starting at the current position of this buffer and at the given offset in the array. The position of this buffer is then incremented by length.

Parameters

<i>buffer</i>	The pointer to an allocated buffer to fill.
<i>size</i>	The size of the buffer passed.
<i>offset</i>	The position in the buffer to start filling.
<i>length</i>	The amount of data to put in the passed buffer.

Returns

a reference to this **Buffer** (p. 582).

Exceptions

<i>BufferUnderflow-Exception</i> (p. 611)	if there are fewer than length doubles remaining in this buffer
<i>NullPointerException</i>	if the passed buffer is null.
<i>IndexOutOfBoundsException</i>	if the preconditions of size, offset, or length are not met.

6.216.3.13 virtual bool decaf::nio::DoubleBuffer::hasArray () const [pure virtual]

Tells whether or not this buffer is backed by an accessible double array.

If this method returns true then the array and arrayOffset methods may safely be invoked. Subclasses should override this method if they do not have a backing array as this class always returns true.

Returns

true if, and only if, this buffer is backed by an array and is not read-only

Implemented in **decaf::internal::nio::DoubleArrayBuffer** (p. 1257).

6.216.3.14 **virtual bool decaf::nio::DoubleBuffer::operator< (const DoubleBuffer & value)**
const [virtual]

6.216.3.15 **virtual bool decaf::nio::DoubleBuffer::operator== (const DoubleBuffer & value)**
const [virtual]

6.216.3.16 **DoubleBuffer& decaf::nio::DoubleBuffer::put (DoubleBuffer & src)**

This method transfers the doubles remaining in the given source buffer into this buffer.

If there are more doubles remaining in the source buffer than in this buffer, that is, if `src.remaining() > remaining()` (p. 588), then no doubles are transferred and a **BufferOverflowException** (p. 609) is thrown.

Otherwise, this method copies `n = src.remaining()` doubles from the given buffer into this buffer, starting at each buffer's current position. The positions of both buffers are then incremented by `n`.

Parameters

<code>src</code>	The buffer to take doubles from an place in this one.
------------------	---

Returns

a reference to this buffer.

Exceptions

BufferOverflowException (p. 609)	if there is insufficient space in this buffer for the remaining doubles in the source buffer
IllegalArgumentException	if the source buffer is this buffer.
ReadOnlyBufferException (p. 2244)	if this buffer is read-only.

6.216.3.17 **DoubleBuffer& decaf::nio::DoubleBuffer::put (const double * buffer, int size, int offset, int length)**

This method transfers doubles into this buffer from the given source array.

If there are more doubles to be copied from the array than remain in this buffer, that is, if

length > **remaining()** (p. 588), then no doubles are transferred and a **BufferOverflow-Exception** (p. 609) is thrown.

Otherwise, this method copies length bytes from the given array into this buffer, starting at the given offset in the array and at the current position of this buffer. The position of this buffer is then incremented by length.

Parameters

<i>buffer</i>	The array from which doubles are to be read.
<i>size</i>	The size of the buffer passed.
<i>offset</i>	The offset within the array of the first char to be read.
<i>length</i>	The number of doubles to be read from the given array.

Returns

a reference to this buffer.

Exceptions

BufferOverflow-Exception (p. 609)	if there is insufficient space in this buffer
ReadOnlyBuffer-Exception (p. 2244)	if this buffer is read-only
<i>NullPointerException</i>	if the passed buffer is null.
<i>IndexOutOfBoundsException</i>	if the preconditions of size, offset, or length are not met.

6.216.3.18 DoubleBuffer& decaf::nio::DoubleBuffer::put (std::vector< double > & buffer)

This method transfers the entire content of the given source doubles array into this buffer.

This is the same as calling put(&buffer[0], 0, buffer.size()).

Parameters

<i>buffer</i>	The buffer whose contents are copied to this DoubleBuffer (p. 1259).
---------------	---

Returns

a reference to this buffer.

Exceptions

<i>BufferOverflow-Exception</i> (p. 609)	if there is insufficient space in this buffer.
<i>ReadOnlyBuffer-Exception</i> (p. 2244)	if this buffer is read-only.

6.216.3.19 `virtual DoubleBuffer& decaf::nio::DoubleBuffer::put (double value)`
`[pure virtual]`

Writes the given doubles into this buffer at the current position, and then increments the position.

Parameters

<i>value</i>	The doubles value to be written.
--------------	----------------------------------

Returns

a reference to this buffer.

Exceptions

<i>BufferOverflow-Exception</i> (p. 609)	if this buffer's current position is not smaller than its limit.
<i>ReadOnlyBuffer-Exception</i> (p. 2244)	if this buffer is read-only.

Implemented in `decaf::internal::nio::DoubleArrayBuffer` (p. 1257).

6.216.3.20 `virtual DoubleBuffer& decaf::nio::DoubleBuffer::put (int index, double value)`
`[pure virtual]`

Writes the given doubles into this buffer at the given index.

Parameters

<i>index</i>	The position in the Buffer (p. 582) to write the data.
<i>value</i>	The doubles to write.

Returns

a reference to this buffer

Exceptions

<i>IndexOutOfBounds-Exception</i>	if index greater than the buffer's limit minus the size of the type being written, or the index is negative.
ReadOnlyBuffer-Exception (p. 2244)	if this buffer is read-only.

Implemented in **decaf::internal::nio::DoubleArrayBuffer** (p. 1258).

6.216.3.21 `virtual DoubleBuffer* decaf::nio::DoubleBuffer::slice () const` [pure virtual]

Creates a new **DoubleBuffer** (p. 1259) whose content is a shared subsequence of this buffer's content.

The content of the new buffer will start at this buffer's current position. Changes to this buffer's content will be visible in the new buffer, and vice versa; the two buffers' position, limit, and mark values will be independent.

The new buffer's position will be zero, its capacity and its limit will be the number of bytes remaining in this buffer, and its mark will be undefined. The new buffer will be read-only if, and only if, this buffer is read-only.

Returns

the newly create **DoubleBuffer** (p. 1259) which the caller owns.

Implemented in **decaf::internal::nio::DoubleArrayBuffer** (p. 1259).

6.216.3.22 `virtual std::string decaf::nio::DoubleBuffer::toString () const` [virtual]

Returns

a std::string describing this object

6.216.3.23 `static DoubleBuffer* decaf::nio::DoubleBuffer::wrap (double * array, int size, int offset, int length)` [static]

Wraps the passed buffer with a new **DoubleBuffer** (p. 1259).

The new buffer will be backed by the given double array; that is, modifications to the buffer will cause the array to be modified and vice versa. The new buffer's capacity will be array.length, its position will be offset, its limit will be offset + length, and its mark will be undefined. Its backing array will be the given array, and its array offset will be zero.

Parameters

<i>array</i>	The array that will back the new buffer.
<i>size</i>	The size of the passed in array.
<i>offset</i>	The offset of the subarray to be used.
<i>length</i>	The length of the subarray to be used.

Returns

a new **DoubleBuffer** (p. 1259) that is backed by buffer, caller owns.

Exceptions

<i>NullPointerException</i>	if the array pointer is NULL.
<i>IndexOutOfBoundsException</i>	if the preconditions of size, offset, or length are not met.

6.216.3.24 `static DoubleBuffer* decaf::nio::DoubleBuffer::wrap (std::vector< double > & buffer) [static]`

Wraps the passed STL double Vector in a **DoubleBuffer** (p. 1259).

The new buffer will be backed by the given double array; modifications to the buffer will cause the array to be modified and vice versa. The new buffer's capacity and limit will be `buffer.size()`, its position will be zero, and its mark will be undefined. Its backing array will be the given array, and its array offset will be zero.

Parameters

<i>buffer</i>	The vector that will back the new buffer, the vector must have been sized to the desired size already by calling <code>vector.resize(N)</code> .
---------------	--

Returns

a new **DoubleBuffer** (p. 1259) that is backed by buffer, caller owns.

The documentation for this class was generated from the following file:

- `src/main/decaf/nio/DoubleBuffer.h`

6.217 decaf::lang::DYNAMIC_CAST_TOKEN Struct Reference

```
#include <src/main/decaf/lang/Pointer.h>
```

The documentation for this struct was generated from the following file:

- `src/main/decaf/lang/Pointer.h`

6.218 activemq::cmsutil::DynamicDestinationResolver Class - Reference

Resolves a CMS destination name to a `Destination`.

```
#include <src/main/activemq/cmsutil/DynamicDestination-Resolver.h>
```

Inheritance diagram for `activemq::cmsutil::DynamicDestinationResolver`:

Data Structures

- class **SessionResolver**
Manages maps of names to topics and queues for a single session.

Public Member Functions

- **DynamicDestinationResolver** ()
- virtual **~DynamicDestinationResolver** () throw ()
- virtual void **init** (**ResourceLifecycleManager** *mgr)
Initializes this destination resolver for use.
- virtual void **destroy** ()
Destroys any allocated resources.
- virtual **cms::Destination** * **resolveDestinationName** (**cms::Session** *session, const std::string &destName, bool pubSubDomain)
Resolves the given name to a destination.

6.218.1 Detailed Description

Resolves a CMS destination name to a `Destination`.

6.218.2 Constructor & Destructor Documentation

6.218.2.1 `activemq::cmsutil::DynamicDestinationResolver::DynamicDestinationResolver ()`

6.218.2.2 `virtual activemq::cmsutil::DynamicDestinationResolver::~~DynamicDestinationResolver () throw ()`
[virtual]

6.218.3 Member Function Documentation

6.218.3.1 `virtual void activemq::cmsutil::DynamicDestinationResolver::destroy ()`
`[virtual]`

Destroys any allocated resources.

Implements **activemq::cmsutil::DestinationResolver** (p. 1223).

6.218.3.2 `virtual void activemq::cmsutil::DynamicDestinationResolver::init (`
`ResourceLifecycleManager * mgr) [inline, virtual]`

Initializes this destination resolver for use.

Ensures that any previously allocated resources are first destroyed (e.g. calls **destroy()** (p. 1273)).

Parameters

<i>mgr</i>	the resource lifecycle manager.
------------	---------------------------------

Implements **activemq::cmsutil::DestinationResolver** (p. 1223).

6.218.3.3 `virtual cms::Destination* activemq::cmsutil::DynamicDestination-`
`Resolver::resolveDestinationName (cms::Session * session, const`
`std::string & destName, bool pubSubDomain) [virtual]`

Resolves the given name to a destination.

If `pubSubDomain` is true, a topic will be returned, otherwise a queue will be returned.

Parameters

<i>session</i>	the session for which to retrieve resolve the destination.
<i>destName</i>	the name to be resolved.
<i>pubSub-</i> <i>Domain</i>	If true, the name will be resolved to a Topic, otherwise a Queue.

Returns

the resolved destination

Exceptions

<i>cms::CMS-</i> <i>Exception</i> (p. 826)	if resolution failed.
--	-----------------------

Implements **activemq::cmsutil::DestinationResolver** (p. 1223).

The documentation for this class was generated from the following file:

- src/main/activemq/cmsutil/**DynamicDestinationResolver.h**

6.219 decaf::util::Map< K, V, COMPARATOR >::Entry Class - Reference

```
#include <src/main/decaf/util/Map.h>
```

Public Member Functions

- **Entry** ()
- virtual **~Entry** ()
- virtual const K & **getKey** () const =0
- virtual const V & **getValue** () const =0
- virtual void **setValue** (const V &value)=0

```
template<typename K, typename V, typename COMPARATOR = std::less<K>> class decaf::util-  
::Map< K, V, COMPARATOR >::Entry
```

6.219.1 Constructor & Destructor Documentation

6.219.1.1 `template<typename K, typename V, typename COMPARATOR = std::less<K>>
decaf::util::Map< K, V, COMPARATOR >::Entry::Entry () [inline]`

6.219.1.2 `template<typename K, typename V, typename COMPARATOR = std::less<K>>
virtual decaf::util::Map< K, V, COMPARATOR >::Entry::~Entry ()
[inline, virtual]`

6.219.2 Member Function Documentation

6.219.2.1 `template<typename K, typename V, typename COMPARATOR = std::less<K>>
virtual const K& decaf::util::Map< K, V, COMPARATOR >::Entry::getKey ()
const [pure virtual]`

6.219.2.2 `template<typename K, typename V, typename COMPARATOR = std::less<K>>
virtual const V& decaf::util::Map< K, V, COMPARATOR >::Entry::getValue ()
const [pure virtual]`

6.219.2.3 `template<typename K, typename V, typename COMPARATOR = std::less<K>>
virtual void decaf::util::Map< K, V, COMPARATOR >::Entry::setValue (const V
& value) [pure virtual]`

The documentation for this class was generated from the following file:

- src/main/decaf/util/**Map.h**

6.220 decaf::io::EOFException Class Reference

```
#include <src/main/decaf/io/EOFException.h>
```

Inheritance diagram for decaf::io::EOFException:

Public Member Functions

- **EOFException** () throw ()
Default Constructor.
- **EOFException** (const lang::Exception &ex) throw ()
Copy Constructor.
- **EOFException** (const EOFException &ex) throw ()
Copy Constructor.
- **EOFException** (const char *file, const int lineNumber, const std::exception *cause, const char *msg,...) throw ()
Constructor - Initializes the file name and line number where this message occurred.
- **EOFException** (const std::exception *cause) throw ()
Constructor.
- **EOFException** (const char *file, const int lineNumber, const char *msg,...) throw ()
Constructor.
- virtual **EOFException** * clone () const
Clones this exception.
- virtual ~**EOFException** () throw ()

6.220.1 Constructor & Destructor Documentation

6.220.1.1 decaf::io::EOFException::EOFException () throw () [inline]

Default Constructor.

6.220.1.2 decaf::io::EOFException::EOFException (const lang::Exception & ex) throw () [inline]

Copy Constructor.

Parameters

ex	the exception to copy
----	-----------------------

6.220.1.3 **decaf::io::EOFException::EOFException** (*const EOFException & ex*)
throw () `[inline]`

Copy Constructor.

Parameters

<i>ex</i>	the exception to copy, which is an instance of this type
-----------	--

6.220.1.4 **decaf::io::EOFException::EOFException** (*const char * file*, *const int lineNumber*, *const std::exception * cause*, *const char * msg*, ...) **throw ()**
`[inline]`

Constructor - Initializes the file name and line number where this message occurred.

Sets the message to report, using an optional list of arguments to parse into the message

Parameters

<i>file</i>	The file name where exception occurs
<i>lineNumber</i>	The line number where the exception occurred.
<i>cause</i>	The exception that was the cause for this one to be thrown.
<i>msg</i>	The message to report
...	list of primitives that are formatted into the message

6.220.1.5 **decaf::io::EOFException::EOFException** (*const std::exception * cause*)
throw () `[inline]`

Constructor.

Parameters

<i>cause</i>	Pointer to the exception that caused this one to be thrown, the object is cloned caller retains ownership.
--------------	--

6.220.1.6 **decaf::io::EOFException::EOFException** (*const char * file*, *const int lineNumber*, *const char * msg*, ...) **throw ()** `[inline]`

Constructor.

Parameters

<i>file</i>	The file name where exception occurs
<i>lineNumber</i>	The line number where the exception occurred.
<i>msg</i>	The message to report
...	list of primitives that are formatted into the message

6.220.1.7 `virtual decaf::io::EOFException::~~EOFException () throw () [inline, virtual]`

6.220.2 Member Function Documentation

6.220.2.1 `virtual EOFException* decaf::io::EOFException::clone () const [inline, virtual]`

Clones this exception.

This is useful for cases where you need to preserve the type of the original exception as well as the message. All subclasses should override.

Returns

a new instance of an Exception that is a copy of this one.

Reimplemented from **decaf::io::IOException** (p. 1547).

The documentation for this class was generated from the following file:

- `src/main/decaf/io/EOFException.h`

6.221 decaf::util::logging::ErrorManager Class Reference

ErrorManager (p. 1277) objects can be attached to Handlers to process any error that occur on a **Handler** (p. 1401) during Logging.

```
#include <src/main/decaf/util/logging/ErrorManager.h>
```

Public Member Functions

- **ErrorManager** ()
- virtual **~ErrorManager** ()
- virtual void **error** (const std::string &message, **decaf::lang::Exception** *ex, int code)

*The error method is called when a **Handler** (p. 1401) failure occurs.*

Static Public Attributes

- static const int **GENERIC_FAILURE**
GENERIC_FAILURE is used for failure that don't fit into one of the other categories.
- static const int **WRITE_FAILURE**
WRITE_FAILURE is used when a write to an output stream fails.
- static const int **FLUSH_FAILURE**
FLUSH_FAILURE is used when a flush to an output stream fails.

- static const int **CLOSE_FAILURE**
CLOSE_FAILURE is used when a close of an output stream fails.
- static const int **OPEN_FAILURE**
OPEN_FAILURE is used when an open of an output stream fails.
- static const int **FORMAT_FAILURE**
FORMAT_FAILURE is used when formatting fails for any reason.

6.221.1 Detailed Description

ErrorManager (p. 1277) objects can be attached to Handlers to process any error that occur on a **Handler** (p. 1401) during Logging.

When processing logging output, if a **Handler** (p. 1401) encounters problems then rather than throwing an Exception back to the issuer of the logging call (who is unlikely to be interested) the **Handler** (p. 1401) should call its associated **ErrorManager** (p. 1277).

Since

1.0

6.221.2 Constructor & Destructor Documentation

6.221.2.1 **decaf::util::logging::ErrorManager::ErrorManager ()**

6.221.2.2 **virtual decaf::util::logging::ErrorManager::~~ErrorManager ()**
[virtual]

6.221.3 Member Function Documentation

6.221.3.1 **virtual void decaf::util::logging::ErrorManager::error (const std::string & message, decaf::lang::Exception * ex, int code)** [virtual]

The error method is called when a **Handler** (p. 1401) failure occurs.

This method may be overridden in subclasses. The default behavior in this base class is that the first call is reported to System.err, and subsequent calls are ignored.

Parameters

<i>msg</i>	- a descriptive string (may be empty)
<i>ex</i>	- an exception (may be NULL)
<i>code</i>	- an error code defined in ErrorManager (p. 1277)

6.221.4 Field Documentation

6.221.4.1 `const int decaf::util::logging::ErrorManager::CLOSE_FAILURE`
[static]

CLOSE_FAILURE is used when a close of an output stream fails.

6.221.4.2 `const int decaf::util::logging::ErrorManager::FLUSH_FAILURE`
[static]

FLUSH_FAILURE is used when a flush to an output stream fails.

6.221.4.3 `const int decaf::util::logging::ErrorManager::FORMAT_FAILURE`
[static]

FORMAT_FAILURE is used when formatting fails for any reason.

6.221.4.4 `const int decaf::util::logging::ErrorManager::GENERIC_FAILURE`
[static]

GENERIC_FAILURE is used for failure that don't fit into one of the other categories.

6.221.4.5 `const int decaf::util::logging::ErrorManager::OPEN_FAILURE`
[static]

OPEN_FAILURE is used when an open of an output stream fails.

6.221.4.6 `const int decaf::util::logging::ErrorManager::WRITE_FAILURE`
[static]

WRITE_FAILURE is used when a write to an output stream fails.

The documentation for this class was generated from the following file:

- `src/main/decaf/util/logging/ErrorManager.h`

6.222 decaf::lang::Exception Class Reference

```
#include <src/main/decaf/lang/Exception.h>
```

Inheritance diagram for decaf::lang::Exception:

Public Member Functions

- **Exception** () throw ()
Default Constructor.
- **Exception** (const **Exception** &ex) throw ()
Copy Constructor.
- **Exception** (const std::exception ***cause**) throw ()
Constructor.
- **Exception** (const char *file, const int lineNumber, const char *msg,...) throw ()
Constructor - Initializes the file name and line number where this message occurred.
- **Exception** (const char *file, const int lineNumber, const std::exception ***cause**, const char *msg,...) throw ()
Constructor - Initializes the file name and line number where this message occurred.
- virtual ~**Exception** () throw ()
- virtual std::string **getMessage** () const
Gets the message for this exception.
- virtual const std::exception * **getCause** () const
Gets the exception that caused this one to be thrown, this allows for chaining of exceptions in the case of a method that throws only a particular exception but wishes to allow for the real causal exception to be passed only in case the caller knows about that type of exception and wishes to respond to it.
- virtual void **initCause** (const std::exception ***cause**)
Initializes the contained cause exception with the one given.
- virtual const char * **what** () const throw ()
Implement method from std::exception.
- virtual void **setMessage** (const char *msg,...)
Sets the cause for this exception.
- virtual void **setMark** (const char *file, const int lineNumber)
Adds a file/line number to the stack trace.
- virtual **Exception** * **clone** () const
Clones this exception.
- virtual std::vector< std::pair < std::string, int > > **getStackTrace** () const
Provides the stack trace for every point where this exception was caught, marked, and rethrown.
- virtual void **printStackTrace** () const
Prints the stack trace to std::err.
- virtual void **printStackTrace** (std::ostream &stream) const
Prints the stack trace to the given output stream.
- virtual std::string **getStackTraceString** () const
Gets the stack trace as one contiguous string.
- **Exception** & **operator=** (const **Exception** &ex)
Assignment operator.

Protected Member Functions

- virtual void **setStackTrace** (const std::vector< std::pair< std::string, int > > &trace)
- virtual void **buildMessage** (const char *format, va_list &args)

Protected Attributes

- std::string **message**
The cause of this exception.
- std::exception * **cause**
*The **Exception** (p. 1279) that caused this one to be thrown.*
- std::vector< std::pair < std::string, int > > **stackTrace**
The stack trace.

6.222.1 Constructor & Destructor Documentation

6.222.1.1 decaf::lang::Exception::Exception () throw ()

Default Constructor.

6.222.1.2 decaf::lang::Exception::Exception (const Exception & ex) throw ()

Copy Constructor.

Parameters

<i>ex</i>	The Exception (p. 1279) instance to copy.
-----------	--

6.222.1.3 decaf::lang::Exception::Exception (const std::exception * cause) throw ()

Constructor.

Parameters

<i>cause</i>	Pointer (p. 2083) to the exception that caused this one to be thrown, the object is cloned caller retains ownership.
--------------	---

6.222.1.4 decaf::lang::Exception::Exception (const char * file, const int lineNumber, const char * msg, ...) throw ()

Constructor - Initializes the file name and line number where this message occurred.

Sets the message to report, using an optional list of arguments to parse into the message

Parameters

<i>file</i>	The file name where exception occurs
<i>lineNumber</i>	The line number where the exception occurred.
<i>msg</i>	The message to report
...	list of primitives that are formatted into the message

6.222.1.5 decaf::lang::Exception::Exception (const char * *file*, const int *lineNumber*, const std::exception * *cause*, const char * *msg*, ...) throw ()

Constructor - Initializes the file name and line number where this message occurred.

Sets the message to report, using an optional list of arguments to parse into the message

Parameters

<i>file</i>	The file name where exception occurs
<i>lineNumber</i>	The line number where the exception occurred.
<i>cause</i>	The exception that was the cause for this one to be thrown.
<i>msg</i>	The message to report
...	list of primitives that are formatted into the message

6.222.1.6 virtual decaf::lang::Exception::~~Exception () throw () [virtual]

6.222.2 Member Function Documentation

6.222.2.1 virtual void decaf::lang::Exception::buildMessage (const char * *format*, va_list & *vargs*) [protected, virtual]

Referenced by decaf::lang::exceptions::NumberFormatException::NumberFormatException().

6.222.2.2 virtual Exception* decaf::lang::Exception::clone () const [virtual]

Clones this exception.

This is useful for cases where you need to preserve the type of the original exception as well as the message. All subclasses should override.

Returns

Copy of this **Exception** (p. 1279) object

Implements **decaf::lang::Throwable** (p. 2733).

Reimplemented in **decaf::net::URISyntaxException** (p. 2859), **decaf::internal::net::ssl::openssl::OpenSSLSocketException** (p. 2032), **decaf::lang::exceptions::**

`InvalidStateException` (p. 1545), `decaf::lang::exceptions::IllegalMonitorStateException` (p. 1418), `decaf::lang::exceptions::InterruptedException` (p. 1531), `decaf::security::GeneralSecurityException` (p. 1397), `decaf::security::NoSuchProviderException` (p. 1989), `decaf::security::NoSuchAlgorithmException` (p. 1983), `decaf::security::SignatureException` (p. 2440), `decaf::lang::exceptions::IllegalStateException` (p. 1422), `decaf::lang::exceptions::IllegalThreadStateException` (p. 1425), `decaf::lang::exceptions::NullPointerException` (p. 1991), `decaf::security::InvalidKeyException` (p. 1538), `decaf::lang::exceptions::IllegalArgumentException` (p. 1416), `decaf::util::concurrent::BrokenBarrierException` (p. 558), `decaf::lang::exceptions::RuntimeException` (p. 2317), `decaf::security::KeyException` (p. 1603), `decaf::security::KeyManagementException` (p. 1605), `decaf::util::concurrent::ExecutionException` (p. 1297), `decaf::util::concurrent::TimeoutException` (p. 2739), `decaf::util::NoSuchElementException` (p. 1986), `decaf::lang::exceptions::ClassCastException` (p. 815), `decaf::lang::exceptions::IndexOutOfBoundsException` (p. 1431), `decaf::lang::exceptions::NumberFormatException` (p. 1997), `decaf::util::concurrent::CancellationException` (p. 750), `decaf::util::concurrent::RejectedExecutionException` (p. 2266), `decaf::net::BindException` (p. 534), `decaf::lang::exceptions::UnsupportedOperationException` (p. 2827), `decaf::net::ConnectException` (p. 933), `decaf::nio::InvalidMarkException` (p. 1541), `decaf::io::UTFDataFormatException` (p. 2877), `decaf::io::UnsupportedEncodingException` (p. 2824), `decaf::net::HttpRetryException` (p. 1411), `decaf::net::MalformedURLException` (p. 1768), `decaf::net::NoRouteToHostException` (p. 1981), `decaf::net::PortUnreachableException` (p. 2109), `decaf::net::ProtocolException` (p. 2213), `decaf::net::SocketTimeoutException` (p. 2495), `decaf::net::UnknownHostException` (p. 2818), `decaf::net::UnknownServiceException` (p. 2821), `decaf::util::zip::ZipException` (p. 2955), `decaf::io::EOFException` (p. 1277), `decaf::io::InterruptedIOException` (p. 1533), `decaf::nio::ReadOnlyBufferException` (p. 2246), `decaf::util::ConcurrentModificationException` (p. 904), `decaf::util::zip::DataFormatException` (p. 1078), `decaf::io::IOException` (p. 1547), `decaf::nio::BufferOverflowException` (p. 611), `decaf::nio::BufferUnderflowException` (p. 614), `decaf::net::SocketException` (p. 2473), `decaf::security::cert::CertificateExpiredException` (p. 760), `decaf::security::cert::CertificateNotYetValidException` (p. 762), `decaf::security::cert::CertificateParsingException` (p. 764), `decaf::security::cert::CertificateEncodingException` (p. 756), `decaf::security::cert::CertificateException` (p. 758), `activemq::exceptions::ActiveMQException` (p. 263), `activemq::exceptions::BrokerException` (p. 564), and `activemq::exceptions::ConnectionFailedException` (p. 959).

6.222.2.3 `virtual const std::exception* decaf::lang::Exception::getCause() const`
`[inline, virtual]`

Gets the exception that caused this one to be thrown, this allows for chaining of exceptions in the case of a method that throws only a particular exception but wishes to allow for the real causal exception to be passed only in case the caller knows about that type of exception and wishes to respond to it.

Returns

a const pointer reference to the causal exception, if there was no cause associated with this exception then NULL is returned.

Implements **decaf::lang::Throwable** (p. 2734).

6.222.2.4 `virtual std::string decaf::lang::Exception::getMessage () const`
[inline, virtual]

Gets the message for this exception.

Returns

Text formatted error message

Implements **decaf::lang::Throwable** (p. 2735).

Referenced by `activemq::exceptions::BrokerException::BrokerException()`.

6.222.2.5 `virtual std::vector< std::pair< std::string, int> >`
`decaf::lang::Exception::getStackTrace () const` [virtual]

Provides the stack trace for every point where this exception was caught, marked, and rethrown.

The first item in the returned vector is the first point where the mark was set (e.g. where the exception was created).

Returns

the stack trace.

Implements **decaf::lang::Throwable** (p. 2735).

6.222.2.6 `virtual std::string decaf::lang::Exception::getStackTraceString () const`
[virtual]

Gets the stack trace as one contiguous string.

Returns

string with formatted stack trace data

Implements **decaf::lang::Throwable** (p. 2735).

6.222.2.7 `virtual void decaf::lang::Exception::initCause (const std::exception * cause)`
`[virtual]`

Initializes the contained cause exception with the one given.

A copy is made to avoid ownership issues.

Parameters

<i>cause</i>	The exception that was the cause of this one.
--------------	---

Implements **decaf::lang::Throwable** (p. 2736).

6.222.2.8 `Exception& decaf::lang::Exception::operator= (const Exception & ex)`

Assignment operator.

Parameters

<i>ex</i>	const reference to another Exception (p. 1279)
-----------	---

6.222.2.9 `virtual void decaf::lang::Exception::printStackTrace () const`
`[virtual]`

Prints the stack trace to std::err.

Implements **decaf::lang::Throwable** (p. 2736).

6.222.2.10 `virtual void decaf::lang::Exception::printStackTrace (std::ostream & stream) const` `[virtual]`

Prints the stack trace to the given output stream.

Parameters

<i>stream</i>	the target output stream.
---------------	---------------------------

Implements **decaf::lang::Throwable** (p. 2736).

6.222.2.11 `virtual void decaf::lang::Exception::setMark (const char * file, const int lineNumber)` `[virtual]`

Adds a file/line number to the stack trace.

Parameters

<i>file</i>	The name of the file calling this method (use <code>__FILE__</code>).
<i>lineNumber</i>	The line number in the calling file (use <code>__LINE__</code>).

Implements **decaf::lang::Throwable** (p. 2736).

Referenced by decaf::lang::exceptions::NumberFormatException::NumberFormatException().

6.222.2.12 virtual void decaf::lang::Exception::setMessage (const char * *msg*, ...)
[virtual]

Sets the cause for this exception.

Parameters

<i>msg</i>	the format string for the msg.
...	params to format into the string

6.222.2.13 virtual void decaf::lang::Exception::setStackTrace (const std::vector< std::pair< std::string, int > > & *trace*) [protected, virtual]

6.222.2.14 virtual const char* decaf::lang::Exception::what () const throw ()
[inline, virtual]

Implement method from std::exception.

Returns

the const char* of **getMessage ()** (p. 1284).

6.222.3 Field Documentation

6.222.3.1 std::exception* decaf::lang::Exception::cause [protected]

The **Exception** (p. 1279) that caused this one to be thrown.

6.222.3.2 std::string decaf::lang::Exception::message [protected]

The cause of this exception.

6.222.3.3 std::vector< std::pair< std::string, int > > decaf::lang::Exception::stackTrace
[protected]

The stack trace.

The documentation for this class was generated from the following file:

- src/main/decaf/lang/**Exception.h**

6.223 cms::ExceptionListener Class Reference

If a CMS provider detects a serious problem, it notifies the client application through an **ExceptionListener** (p. 1286) that is registered with the **Connection** (p. 933).

```
#include <src/main/cms/ExceptionListener.h>
```

Public Member Functions

- virtual **~ExceptionListener** ()
- virtual void **onException** (const **cms::CMSException** &ex)=0

Called when an exception occurs.

6.223.1 Detailed Description

If a CMS provider detects a serious problem, it notifies the client application through an **ExceptionListener** (p. 1286) that is registered with the **Connection** (p. 933).

An exception listener allows a client to be notified of a problem asynchronously. Some connections only consume messages via the asynchronous event mechanism so they would have no other way to learn that their connection has failed.

Since

1.0

6.223.2 Constructor & Destructor Documentation

6.223.2.1 virtual **cms::ExceptionListener::~~ExceptionListener** () [virtual]

6.223.3 Member Function Documentation

6.223.3.1 virtual void **cms::ExceptionListener::onException** (const **cms::CMSException** & ex) [pure virtual]

Called when an exception occurs.

Once notified of an exception the caller should no longer use the resource that generated the exception.

Parameters

ex	Exception Object that occurred.
-----------	---------------------------------

The documentation for this class was generated from the following file:

- src/main/cms/ExceptionListener.h

6.224 activemq::commands::ExceptionResponse Class Reference

```
#include <src/main/activemq/commands/ExceptionResponse.h>
```

Inheritance diagram for activemq::commands::ExceptionResponse:

Public Member Functions

- **ExceptionResponse** ()
- virtual **~ExceptionResponse** ()
- virtual unsigned char **getDataStructureType** () const
*Get the **DataStructure** (p. 1133) Type as defined in CommandTypes.h.*
- virtual **ExceptionResponse** * **cloneDataStructure** () const
Clone this object and return a new instance that the caller now owns, this will be an exact copy of this one.
- virtual void **copyDataStructure** (const **DataStructure** *src)
Copy the contents of the passed object into this objects members, overwriting any existing data.
- virtual std::string **toString** () const
*Returns a string containing the information for this **DataStructure** (p. 1133) such as its type and value of its elements.*
- virtual bool **equals** (const **DataStructure** *value) const
*Compares the **DataStructure** (p. 1133) passed in to this one, and returns if they are equivalent.*
- virtual const **Pointer** < **BrokerError** > & **getException** () const
- virtual **Pointer**< **BrokerError** > & **getException** ()
- virtual void **setException** (const **Pointer**< **BrokerError** > &exception)

Static Public Attributes

- static const unsigned char **ID_EXCEPTIONRESPONSE** = 31

Protected Attributes

- **Pointer**< **BrokerError** > **exception**

6.224.1 Constructor & Destructor Documentation

6.224.1.1 `activemq::commands::ExceptionResponse::ExceptionResponse ()`

6.224.1.2 `virtual activemq::commands::ExceptionResponse::~~ExceptionResponse () [virtual]`

6.224.2 Member Function Documentation

6.224.2.1 `virtual ExceptionResponse* activemq::commands::ExceptionResponse::cloneDataStructure () const [virtual]`

Clone this object and return a new instance that the caller now owns, this will be an exact copy of this one.

Returns

new copy of this object.

Reimplemented from `activemq::commands::Response` (p. 2299).

6.224.2.2 `virtual void activemq::commands::ExceptionResponse::copyDataStructure (const DataStructure * src) [virtual]`

Copy the contents of the passed object into this objects members, overwriting any existing data.

Returns

src - Source Object

Reimplemented from `activemq::commands::Response` (p. 2299).

6.224.2.3 `virtual bool activemq::commands::ExceptionResponse::equals (const DataStructure * value) const [virtual]`

Compares the `DataStructure` (p. 1133) passed in to this one, and returns if they are equivalent.

Equivalent here means that they are of the same type, and that each element of the objects are the same.

Returns

true if `DataStructure` (p. 1133)'s are Equal.

Reimplemented from `activemq::commands::Response` (p. 2299).

6.224.2.4 virtual unsigned char **activemq::commands::ExceptionResponse::getDataStructureType** () const [virtual]

Get the **DataSet** (p. 1133) Type as defined in CommandTypes.h.

Returns

The type of the data structure

Reimplemented from **activemq::commands::Response** (p. 2300).

6.224.2.5 virtual const **Pointer<BrokerError>**& **activemq::commands::ExceptionResponse::getException** () const [virtual]

6.224.2.6 virtual **Pointer<BrokerError>**& **activemq::commands::ExceptionResponse::getException** () [virtual]

6.224.2.7 virtual void **activemq::commands::ExceptionResponse::setException** (const **Pointer< BrokerError >** & *exception*) [virtual]

6.224.2.8 virtual std::string **activemq::commands::ExceptionResponse::toString** () const [virtual]

Returns a string containing the information for this **DataSet** (p. 1133) such as its type and value of its elements.

Returns

formatted string useful for debugging.

Reimplemented from **activemq::commands::Response** (p. 2300).

6.224.3 Field Documentation

6.224.3.1 **Pointer<BrokerError>** **activemq::commands::ExceptionResponse::exception** [protected]

6.224.3.2 const unsigned char **activemq::commands::ExceptionResponse::ID_EXCEPTIONRESPONSE** = 31 [static]

The documentation for this class was generated from the following file:

- src/main/activemq/commands/**ExceptionResponse.h**

6.225 activemq::wireformat::openwire::marshal::generated:- ExceptionResponseMarshaller Class Reference

Marshaling code for Open Wire Format for **ExceptionResponseMarshaller** (p. 1290).

```
#include <src/main/activemq/wireformat/openwire/marshal/generated/-  
ExceptionResponseMarshaller.h>
```

Inheritance diagram for activemq::wireformat::openwire::marshal::generated:-
ExceptionResponseMarshaller:

Public Member Functions

- **ExceptionResponseMarshaller** ()
- virtual **~ExceptionResponseMarshaller** ()
- virtual **commands::DataStructure * createObject** () const
Creates a new instance of the class that this class is a marshaling director for.
- virtual unsigned char **getDataStructureType** () const
Gets the DataStructureType that this class marshals/unmarshals.
- virtual void **tightUnmarshal** (**OpenWireFormat** *wireFormat, **commands:-DataStructure** *dataStructure, **decaf::io::DataInputStream** *dataIn, **utils:-BooleanStream** *bs)
Tight Un-marhsal to the given stream.
- virtual int **tightMarshal1** (**OpenWireFormat** *wireFormat, **commands::Data-Structure** *dataStructure, **utils::BooleanStream** *bs)
Tight Marshal to the given stream.
- virtual void **tightMarshal2** (**OpenWireFormat** *wireFormat, **commands::Data-Structure** *dataStructure, **decaf::io::DataOutputStream** *dataOut, **utils:-BooleanStream** *bs)
Tight Marshal to the given stream.
- virtual void **looseUnmarshal** (**OpenWireFormat** *wireFormat, **commands:-DataStructure** *dataStructure, **decaf::io::DataInputStream** *dataIn)
Loose Un-marhsal to the given stream.
- virtual void **looseMarshal** (**OpenWireFormat** *wireFormat, **commands::Data-Structure** *dataStructure, **decaf::io::DataOutputStream** *dataOut)
Tight Marshal to the given stream.

6.225.1 Detailed Description

Marshaling code for Open Wire Format for **ExceptionResponseMarshaller** (p. 1290).

NOTE!: This file is auto generated - do not modify! if you need to make a change, please see the Java Classes in the activemq-openwire-generator module

6.225 activemq::wireformat::openwire::marshal::generated::Exception-ResponseMarshaller Class

Reference

1295

6.225.2 Constructor & Destructor Documentation

6.225.2.1 **activemq::wireformat::openwire::marshal::generated::Exception-ResponseMarshaller::ExceptionResponseMarshaller ()**
[inline]

6.225.2.2 **virtual activemq::wireformat::openwire::marshal::generated::Exception-ResponseMarshaller::~~ExceptionResponseMarshaller ()** [inline, virtual]

6.225.3 Member Function Documentation

6.225.3.1 **virtual commands::DataStructure* activemq::wireformat::openwire-::marshal::generated::ExceptionResponseMarshaller::createObject ()**
const [virtual]

Creates a new instance of the class that this class is a marshaling director for.

Returns

newly allocated Command

Reimplemented from **activemq::wireformat::openwire::marshal::generated::ResponseMarshaller** (p. 2308).

6.225.3.2 **virtual unsigned char activemq::wireformat::openwire::marshal::generated-::ExceptionResponseMarshaller::getDataStructureType ()** const
[virtual]

Gets the DataStructureType that this class marshals/unmarshals.

Returns

byte Id of this classes DataStructureType

Reimplemented from **activemq::wireformat::openwire::marshal::generated::ResponseMarshaller** (p. 2309).

6.225.3.3 **virtual void activemq::wireformat::openwire::marshal-::generated::ExceptionResponseMarshaller::looseMarshal (**
OpenWireFormat * format, commands::DataStructure * command,
decaf::io::DataOutputStream * ds) [virtual]

Tight Marshal to the given stream.

Parameters

<i>format</i>	- The OpenWireFormat properties
<i>command</i>	- the object to Marshal
<i>ds</i>	- DataOutputStream to marshal to

Exceptions

<i>IOException</i>	if an error occurs.
--------------------	---------------------

Reimplemented from **activemq::wireformat::openwire::marshal::generated::ResponseMarshaller** (p. 2309).

6.225.3.4 `virtual void activemq::wireformat::openwire::marshal::generated::ExceptionResponseMarshaller::looseUnmarshal (OpenWireFormat * format, commands::DataStructure * command, decaf::io::DataInputStream * dis) [virtual]`

Loose Un-marhsal to the given stream.

Parameters

<i>format</i>	- The OpenWireFormat properties
<i>command</i>	- the object to Un-Marshal
<i>dis</i>	- the DataInputStream to Un-Marshal from

Exceptions

<i>IOException</i>	if an error occurs.
--------------------	---------------------

Reimplemented from **activemq::wireformat::openwire::marshal::generated::ResponseMarshaller** (p. 2310).

6.225.3.5 `virtual int activemq::wireformat::openwire::marshal::generated::ExceptionResponseMarshaller::tightMarshal1 (OpenWireFormat * format, commands::DataStructure * command, utils::BooleanStream * bs) [virtual]`

Tight Marhsal to the given stream.

Parameters

<i>format</i>	The OpenWireFormat properties
<i>command</i>	The object to Marshal
<i>bs</i>	The boolean stream to marshal to.

Exceptions

<i>IOException</i>	if an error occurs.
--------------------	---------------------

Reimplemented from **activemq::wireformat::openwire::marshal::generated::ResponseMarshaller** (p. 2310).

6.225 activemq::wireformat::openwire::marshal::generated::Exception-ResponseMarshaller Class

Reference

1297

```
6.225.3.6 virtual void activemq::wireformat::openwire::marshal-  
::generated::ExceptionResponseMarshaller::tightMarshal2 (  
    OpenWireFormat * format, commands::DataStructure * command,  
    decaf::io::DataOutputStream * ds, utils::BooleanStream * bs )  
[virtual]
```

Tight Marhsal to the given stream.

Parameters

<i>format</i>	- The OpenWireFormat properties
<i>command</i>	- the object to Marshal
<i>ds</i>	- the DataOutputStream to Marshal to
<i>bs</i>	- boolean stream to marshal to.

Exceptions

<i>IOException</i>	if an error occurs.
--------------------	---------------------

Reimplemented from **activemq::wireformat::openwire::marshal::generated::-ResponseMarshaller** (p. 2311).

```
6.225.3.7 virtual void activemq::wireformat::openwire::marshal-  
::generated::ExceptionResponseMarshaller::tightUnmarshal (  
    OpenWireFormat * format, commands::DataStructure * command,  
    decaf::io::DataInputStream * dis, utils::BooleanStream * bs )  
[virtual]
```

Tight Un-marhsal to the given stream.

Parameters

<i>format</i>	- The OpenWireFormat properties
<i>command</i>	- the object to Un-Marshal
<i>dis</i>	- the DataInputStream to Un-Marshal from
<i>bs</i>	- boolean stream to unmarshal from.

Exceptions

<i>IOException</i>	if an error occurs.
--------------------	---------------------

Reimplemented from **activemq::wireformat::openwire::marshal::generated::-ResponseMarshaller** (p. 2311).

The documentation for this class was generated from the following file:

- src/main/activemq/wireformat/openwire/marshal/generated/**ExceptionResponseMarshaller.h**

6.226 decaf::util::concurrent::ExecutionException Class Reference

```
#include <src/main/decaf/util/concurrent/ExecutionException.h>
```

Inheritance diagram for decaf::util::concurrent::ExecutionException:

Public Member Functions

- **ExecutionException** () throw ()
Default Constructor.
- **ExecutionException** (const **decaf::lang::Exception** &ex) throw ()
Conversion Constructor from some other Exception.
- **ExecutionException** (const **ExecutionException** &ex) throw ()
Copy Constructor.
- **ExecutionException** (const std::exception ***cause**) throw ()
Constructor.
- **ExecutionException** (const char *file, const int lineNumber, const char *msg,...) throw ()
Constructor - Initializes the file name and line number where this message occurred.
- **ExecutionException** (const char *file, const int lineNumber, const std::exception ***cause**, const char *msg,...) throw ()
Constructor - Initializes the file name and line number where this message occurred.
- virtual **ExecutionException** * **clone** () const
Clones this exception.
- virtual ~**ExecutionException** () throw ()

6.226.1 Constructor & Destructor Documentation

6.226.1.1 **decaf::util::concurrent::ExecutionException::ExecutionException** ()
throw () [inline]

Default Constructor.

6.226.1.2 **decaf::util::concurrent::ExecutionException::ExecutionException** (const **decaf::lang::Exception** & ex) throw () [inline]

Conversion Constructor from some other Exception.

Parameters

ex	- An exception that should become this type of Exception
-----------	--

6.226.1.3 **decaf::util::concurrent::ExecutionException::ExecutionException (**
const ExecutionException & ex) throw () `[inline]`

Copy Constructor.

Parameters

<i>ex</i>	- The Exception to copy in this new instance.
-----------	---

6.226.1.4 **decaf::util::concurrent::ExecutionException::ExecutionException (**
const std::exception * cause) throw () `[inline]`

Constructor.

Parameters

<i>cause</i>	Pointer to the exception that caused this one to be thrown, the object is cloned caller retains ownership.
--------------	--

6.226.1.5 **decaf::util::concurrent::ExecutionException::ExecutionException**
(const char * file, const int lineNumber, const char * msg, ...) throw ()
`[inline]`

Constructor - Initializes the file name and line number where this message occurred.

Sets the message to report, using an optional list of arguments to parse into the message

Parameters

<i>file</i>	- The file name where exception occurs
<i>lineNumber</i>	- The line number where the exception occurred.
<i>msg</i>	- The message to report
<i>...</i>	- The list of primitives that are formatted into the message

6.226.1.6 **decaf::util::concurrent::ExecutionException::ExecutionException (**
const char * file, const int lineNumber, const std::exception * cause, const char *
msg, ...) throw () `[inline]`

Constructor - Initializes the file name and line number where this message occurred.

Sets the message to report, using an optional list of arguments to parse into the message

Parameters

<i>file</i>	- The file name where exception occurs
<i>lineNumber</i>	- The line number where the exception occurred.

<i>cause</i>	- The exception that was the cause for this one to be thrown.
<i>msg</i>	- The message to report
...	- list of primitives that are formatted into the message

6.226.1.7 `virtual decaf::util::concurrent::ExecutionException-
::~ExecutionException () throw () [inline,
virtual]`

6.226.2 Member Function Documentation

6.226.2.1 `virtual ExecutionException* decaf::util::concurrent-
::ExecutionException::clone () const [inline,
virtual]`

Clones this exception.

This is useful for cases where you need to preserve the type of the original exception as well as the message. All subclasses should override.

Returns

a new instance of an exception that is a clone of this one.

Reimplemented from **decaf::lang::Exception** (p. 1282).

The documentation for this class was generated from the following file:

- `src/main/decaf/util/concurrent/ExecutionException.h`

6.227 decaf::util::concurrent::Executor Class Reference

An object that executes submitted **decaf.lang Runnable** (p. 2312) tasks.

```
#include <src/main/decaf/util/concurrent/Executor.h>
```

Inheritance diagram for `decaf::util::concurrent::Executor`:

Public Member Functions

- `virtual ~Executor ()`
- `virtual void execute (decaf::lang::Runnable *command)=0`

Executes the given command at some time in the future.

6.227.1 Detailed Description

An object that executes submitted **decaf.lang Runnable** (p. 2312) tasks.

This interface provides a way of decoupling task submission from the mechanics of how each task will be run, including details of thread use, scheduling, etc. An **Executor** (p. 1297) is normally used instead of explicitly creating threads. For example, rather than invoking `new Thread(new RunnableTask()).start()` for each of a set of tasks, you might use:

```
Executor (p.1297) executor = anExecutor;
executor->execute( new RunnableTask1() );
executor->execute( new RunnableTask2() );
...
```

However, the **Executor** (p. 1297) interface does not strictly require that execution be asynchronous. In the simplest case, an executor can run the submitted task immediately in the caller's thread:

```
class DirectExecutor : public Executor (p.1297) {
public:

    void execute( Runnable* r ) {
        r->run();
    }

}
```

More typically, tasks are executed in some thread other than the caller's thread. The executor below spawns a new thread for each task.

```
class ThreadPerTaskExecutor : public Executor (p.1297) {
public:
    std::vector<Thread*> threads;

    void execute( Runnable* r ) {
        threads.push_back( new Thread( r ) );
        threads.rbegin()->start();
    }

}
```

The **Executor** (p. 1297) implementations provided in this package implement **decaf.util.concurrent.ExecutorService** (p. 1302), which is a more extensive interface. The **decaf.util.concurrent.ThreadPoolExecutor** (p. 2715) class provides an extensible thread pool implementation. The **decaf.util.concurrentExecutor** (p. ??) class provides convenient factory methods for these **Executors** (p. 1299).

Since

1.0

6.227.2 Constructor & Destructor Documentation

6.227.2.1 `virtual decaf::util::concurrent::Executor::~~Executor () [inline, virtual]`

6.227.3 Member Function Documentation

6.227.3.1 `virtual void decaf::util::concurrent::Executor::execute (decaf::lang::Runnable * command) [pure virtual]`

Executes the given command at some time in the future.

The command may execute in a new thread, in a pooled thread, or in the calling thread, at the discretion of the **Executor** (p. 1297) implementation.

Parameters

<i>command</i>	the runnable task
----------------	-------------------

Exceptions

RejectedExecutionException (p. 2263)	if this task cannot be accepted for execution.
<i>NullPointerException</i>	if command is null

Implemented in **decaf::util::concurrent::ThreadPoolExecutor** (p. 2724).

The documentation for this class was generated from the following file:

- `src/main/decaf/util/concurrent/Executor.h`

6.228 decaf::util::concurrent::Executors Class Reference

Implements a set of utilities for use with **Executors** (p. 1299), **ExecutorService** (p. 1302), **ThreadFactory** (p. 2714), and **Callable** (p. 746) types, as well as providing factory methods for instance of these types configured for the most common use cases.

```
#include <src/main/decaf/util/concurrent/Executors.h>
```

Public Member Functions

- virtual **~Executors** ()

Static Public Member Functions

- static **ThreadFactory** * **getDefaultThreadFactory** ()

Creates and returns a new **ThreadFactory** (p. 2714) that expresses the default behavior for ThreadFactories used in **Executor** (p. 1297) classes.

- static **ExecutorService** * **newFixedThreadPool** (int nThreads)

Creates a new **ThreadPoolExecutor** (p. 2715) with a fixed number of threads to process incoming tasks.

- static **ExecutorService** * **newFixedThreadPool** (int nThreads, **ThreadFactory** *threadFactory)

Creates a new **ThreadPoolExecutor** (p. 2715) with a fixed number of threads to process incoming tasks.

Friends

- class **decaf::lang::Thread**

6.228.1 Detailed Description

Implements a set of utilities for use with **Executors** (p. 1299), **ExecutorService** (p. 1302), **ThreadFactory** (p. 2714), and **Callable** (p. 746) types, as well as providing factory methods for instance of these types configured for the most common use cases.

Since

1.0

6.228.2 Constructor & Destructor Documentation

6.228.2.1 virtual **decaf::util::concurrent::Executors::~~Executors** () [virtual]

6.228.3 Member Function Documentation

6.228.3.1 static **ThreadFactory*** **decaf::util::concurrent::Executors::getDefaultThreadFactory** () [static]

Creates and returns a new **ThreadFactory** (p. 2714) that expresses the default behavior for ThreadFactories used in **Executor** (p. 1297) classes.

The default factory create a new non-daemon thread with normal priority and a name whose value is equal to pool-N-thread-M, where N is the sequence number of this factory, and M is the sequence number of the thread created by this factory.

Returns

a new instance of the default thread factory used in **Executors** (p. 1299), the caller takes ownership of the returned pointer.

6.228.3.2 **static ExecutorService* decaf::util::concurrent::Executors::newFixedThreadPool (int *nThreads*)**
[static]

Creates a new **ThreadPoolExecutor** (p. 2715) with a fixed number of threads to process incoming tasks.

The thread pool will use an unbounded queue to store pending tasks. At any given time the maximum threads in the pool will be equal to the number given to this factory method. If a thread in the pool dies a new one will be spawned to take its place in the pool. Tasks that are submitted when all pooled threads are busy will be held until a thread is freed if the pool has allocated its assigned number of threads already.

Parameters

<i>nThreads</i>	The number of threads to assign as the max for the new ExecutorService (p. 1302).
-----------------	--

Returns

pointer to a new **ExecutorService** (p. 1302) that is owned by the caller.

Exceptions

<i>IllegalArgumentException</i>	if <i>nThreads</i> is less than or equal to zero.
---------------------------------	---

6.228.3.3 **static ExecutorService* decaf::util::concurrent::Executors::newFixedThreadPool (int *nThreads*, ThreadFactory * *threadFactory*)**
[static]

Creates a new **ThreadPoolExecutor** (p. 2715) with a fixed number of threads to process incoming tasks.

The thread pool will use an unbounded queue to store pending tasks. At any given time the maximum threads in the pool will be equal to the number given to this factory method. If a thread in the pool dies a new one will be spawned to take its place in the pool. Tasks that are submitted when all pooled threads are busy will be held until a thread is freed if the pool has allocated its assigned number of threads already.

Parameters

<i>nThreads</i>	The number of threads to assign as the max for the new ExecutorService (p. 1302).
<i>threadFactory</i>	Instance of a ThreadFactory (p. 2714) that will be used by the Executor (p. 1297) to spawn new worker threads. This parameter cannot be NULL.

Returns

pointer to a new **ExecutorService** (p. 1302) that is owned by the caller.

Exceptions

<i>NullPointerException</i>	if threadFactory is NULL.
<i>IllegalArgumentException</i>	if nThreads is less than or equal to zero.

6.228.4 Friends And Related Function Documentation**6.228.4.1 friend class decaf::lang::Thread** [friend]

The documentation for this class was generated from the following file:

- src/main/decaf/util/concurrent/**Executors.h**

6.229 decaf::util::concurrent::ExecutorService Class Reference

An **Executor** (p. 1297) that provides methods to manage termination and methods that can produce a **Future** (p. 1390) for tracking progress of one or more asynchronous tasks.

```
#include <src/main/decaf/util/concurrent/ExecutorService.h>
```

Inheritance diagram for decaf::util::concurrent::ExecutorService:

Public Member Functions

- virtual **~ExecutorService** ()
- virtual bool **awaitTermination** (long long timeout, const **TimeUnit** &unit)=0
The caller will block until the executor has completed termination meaning all tasks that were scheduled before shutdown have now completed and the executor is ready for deletion.
- virtual void **shutdown** ()=0
*Performs an orderly shutdown of this **Executor** (p. 1297).*
- virtual **ArrayList** < **decaf::lang::Runnable** * > **shutdownNow** ()=0
*Attempts to stop all currently executing tasks and returns an **ArrayList** (p. 445) containing the **Runnables** that did not get executed, these object become the property of the caller and are not deleted by this class, they are removed from the work queue and forgotten about.*
- virtual bool **isShutdown** () const =0

Returns whether this executor has been shutdown or not.

- virtual bool **isTerminated** () const =0

Returns whether all tasks have completed after this executor was shut down.

6.229.1 Detailed Description

An **Executor** (p. 1297) that provides methods to manage termination and methods that can produce a **Future** (p. 1390) for tracking progress of one or more asynchronous tasks.

An **ExecutorService** (p. 1302) can be shut down, which will cause it to reject new tasks. Two different methods are provided for shutting down an **ExecutorService** (p. 1302). The **shutdown()** (p. 1304) method will allow previously submitted tasks to execute before terminating, while the **shutdownNow()** (p. 1305) method prevents waiting tasks from starting and attempts to stop currently executing tasks. Upon termination, an executor has no tasks actively executing, no tasks awaiting execution, and no new tasks can be submitted. An unused **ExecutorService** (p. 1302) should be shut down to allow reclamation of its resources.

Method **submit** extends base method **Executor.execute** (p. 1299)(**decaf.lang.-Runnable** (p. 2312)) by creating and returning a **Future** (p. 1390) that can be used to cancel execution and/or wait for completion. Methods **invokeAny** and **invokeAll** perform the most commonly useful forms of bulk execution, executing a collection of tasks and then waiting for at least one, or all, to complete. (Class **ExecutorCompletionService** can be used to write customized variants of these methods.)

The **Executors** (p. 1299) class provides factory methods for the executor services provided in this package.

Since

1.0

6.229.2 Constructor & Destructor Documentation

6.229.2.1 virtual **decaf::util::concurrent::ExecutorService::~~ExecutorService** ()
[inline, virtual]

6.229.3 Member Function Documentation

6.229.3.1 virtual bool **decaf::util::concurrent::ExecutorService::awaitTermination** (long long *timeout*, const **TimeUnit** & *unit*) [pure virtual]

The caller will block until the executor has completed termination meaning all tasks that were scheduled before shutdown have now completed and the executor is ready for deletion.

If the timeout period elapses before the executor reaches the terminated state then this method return false to indicate it has not terminated.

Parameters

<i>timeout</i>	The amount of time to wait before abandoning the wait for termination.
<i>unit</i>	The unit of time that the timeout value represents.

Returns

true if the executor terminated or false if the timeout expired.

Exceptions

<i>InterruptedException</i>	if this call is interrupted while awaiting termination.
-----------------------------	---

Implemented in **decaf::util::concurrent::ThreadPoolExecutor** (p. 2723).

6.229.3.2 virtual bool **decaf::util::concurrent::ExecutorService::isShutdown** ()
const [pure virtual]

Returns whether this executor has been shutdown or not.

Returns

true if this executor has been shutdown.

Implemented in **decaf::util::concurrent::ThreadPoolExecutor** (p. 2727).

6.229.3.3 virtual bool **decaf::util::concurrent::ExecutorService::isTerminated** ()
const [pure virtual]

Returns whether all tasks have completed after this executor was shut down.

Returns

true if all tasks have completed after a request to shut down was made.

Implemented in **decaf::util::concurrent::ThreadPoolExecutor** (p. 2727).

6.229.3.4 virtual void **decaf::util::concurrent::ExecutorService::shutdown** ()
[pure virtual]

Performs an orderly shutdown of this **Executor** (p. 1297).

Previously queued tasks are allowed to complete but no new tasks are accepted for execution. Calling this method more than once has no affect on this executor.

Implemented in **decaf::util::concurrent::ThreadPoolExecutor** (p. 2731).

```
6.229.3.5 virtual ArrayList<decaf::lang::Runnable*> decaf::util-
::concurrent::ExecutorService::shutdownNow ( ) [pure
virtual]
```

Attempts to stop all currently executing tasks and returns an **ArrayList** (p. 445) containing the **Runnable**s that did not get executed, these objects become the property of the caller and are not deleted by this class, they are removed from the work queue and forgotten about.

There is no guarantee that this method will halt execution of currently executing tasks.

Returns

an **ArrayList** (p. 445) containing all **Runnable** instance that were still waiting to be executed by this class, call now owns those pointers.

Implemented in **decaf::util::concurrent::ThreadPoolExecutor** (p. 2731).

The documentation for this class was generated from the following file:

- src/main/decaf/util/concurrent/**ExecutorService.h**

6.230 activemq::transport::failover::FailoverTransport Class - Reference

```
#include <src/main/activemq/transport/failover/Failover-
Transport.h>
```

Inheritance diagram for **activemq::transport::failover::FailoverTransport**:

Public Member Functions

- **FailoverTransport** ()
- virtual **~FailoverTransport** ()
- void **reconnect** (bool rebalance)

*Indicates that the **Transport** (p. 2790) needs to reconnect to another URI in its list.*
- void **add** (const std::string &uri)

Adds a New URI to the List of URIs this transport can Connect to.
- virtual void **addURI** (bool rebalance, const **List**< **URI** > &uris)

*Add a URI to the list of URI's that will represent the set of Transports that this **Transport** (p. 2790) is a composite of.*
- virtual void **removeURI** (bool rebalance, const **List**< **URI** > &uris)

*Remove a URI from the set of URI's that represents the set of Transports that this **Transport** (p. 2790) is composed of, removing a URI for which the composite has created a connected **Transport** (p. 2790) should result in that **Transport** (p. 2790) being disposed of.*

- virtual void **start** ()

*Starts the **Transport** (p. 2790), the send methods of a **Transport** (p. 2790) will throw an exception if used before the **Transport** (p. 2790) is started.*
- virtual void **stop** ()

*Stops the **Transport** (p. 2790).*
- virtual void **close** ()

Closes this object and deallocates the appropriate resources.
- virtual void **oneway** (const **Pointer**< **Command** > &command)

Sends a one-way command.
- virtual **Pointer**< **Response** > **request** (const **Pointer**< **Command** > &command)

Sends the given command to the broker and then waits for the response.
- virtual **Pointer**< **Response** > **request** (const **Pointer**< **Command** > &command, unsigned int timeout)

Sends the given command to the broker and then waits for the response.
- virtual **Pointer** < **wireformat::WireFormat** > **getWireFormat** () const

Gets the WireFormat instance that is in use by this transport.
- virtual void **setWireFormat** (const **Pointer**< **wireformat::WireFormat** > &wireFormat AMQCPP_UNUSED)
- virtual void **setTransportListener** (**TransportListener** *listener)

Sets the observer of asynchronous events from this transport.
- virtual **TransportListener** * **getTransportListener** () const

Gets the observer of asynchronous events from this transport.
- virtual bool **isFaultTolerant** () const

*Is this **Transport** (p. 2790) fault tolerant, meaning that it will reconnect to a broker on disconnect.*
- virtual bool **isConnected** () const

*Is the **Transport** (p. 2790) Connected to its Broker.*
- virtual bool **isClosed** () const

*Has the **Transport** (p. 2790) been shutdown and no longer usable.*
- bool **isInitialized** () const
- void **setInitialized** (bool value)
- virtual **Transport** * **narrow** (const std::type_info &typeid)

*Narrows down a Chain of Transports to a specific **Transport** (p. 2790) to allow a higher level transport to skip intermediate Transports in certain circumstances.*
- virtual std::string **getRemoteAddress** () const
- virtual void **reconnect** (const **decaf::net::URI** &uri)

reconnect to another location
- virtual void **updateURIs** (bool rebalance, const **decaf::util::List**< **decaf::net::URI** > &uris)

*Updates the set of URIs the **Transport** (p. 2790) can connect to.*
- virtual bool **isPending** () const
- virtual bool **iterate** ()

Performs the actual Reconnect operation for the **FailoverTransport** (p. 1305), when a connection is made this method returns false to indicate it doesn't need to run again, otherwise it returns true to indicate its still trying to connect.

- long long **getTimeout** () const
- void **setTimeout** (long long value)
- long long **getInitialReconnectDelay** () const
- void **setInitialReconnectDelay** (long long value)
- long long **getMaxReconnectDelay** () const
- void **setMaxReconnectDelay** (long long value)
- long long **getBackOffMultiplier** () const
- void **setBackOffMultiplier** (long long value)
- bool **isUseExponentialBackOff** () const
- void **setUseExponentialBackOff** (bool value)
- bool **isRandomize** () const
- void **setRandomize** (bool value)
- int **getMaxReconnectAttempts** () const
- void **setMaxReconnectAttempts** (int value)
- int **getStartupMaxReconnectAttempts** () const
- void **setStartupMaxReconnectAttempts** (int value)
- long long **getReconnectDelay** () const
- void **setReconnectDelay** (long long value)
- bool **isBackup** () const
- void **setBackup** (bool value)
- int **getBackupPoolSize** () const
- void **setBackupPoolSize** (int value)
- bool **isTrackMessages** () const
- void **setTrackMessages** (bool value)
- bool **isTrackTransactionProducers** () const
- void **setTrackTransactionProducers** (bool value)
- int **getMaxCacheSize** () const
- void **setMaxCacheSize** (int value)
- bool **isReconnectSupported** () const
- void **setReconnectSupported** (bool value)
- bool **isUpdateURIsSupported** () const
- void **setUpdateURIsSupported** (bool value)
- void **setConnectionInterruptProcessingComplete** (const **Pointer**< **commands-::ConnectionId** > &connectionId)

Protected Member Functions

- void **restoreTransport** (const **Pointer**< **Transport** > &transport)
*Given a **Transport** (p. 2790) restore the state of the Client's connection to the Broker using the data accumulated in the State Tracker.*
- void **handleTransportFailure** (const **decaf::lang::Exception** &error)
*Called when this class' **TransportListener** (p. 2810) is notified of a Failure.*
- void **handleConnectionControl** (const **Pointer**< **Command** > &control)
*Called when the Broker sends a **ConnectionControl** command which could signal that this Client needs to reconnect in order to rebalance the connections on a Broker or the set of Known brokers has changed.*

Friends

- class `FailoverTransportListener`

6.230.1 Constructor & Destructor Documentation

6.230.1.1 `activemq::transport::failover::FailoverTransport::FailoverTransport ()`

6.230.1.2 `virtual activemq::transport::failover::FailoverTransport::~~FailoverTransport ()` `[virtual]`

6.230.2 Member Function Documentation

6.230.2.1 `void activemq::transport::failover::FailoverTransport::add (const std::string & uri)`

Adds a New URI to the List of URIs this transport can Connect to.

Parameters

<i>uri</i>	A String version of a URI to add to the URIs to failover to.
------------	--

6.230.2.2 `virtual void activemq::transport::failover::FailoverTransport::addURI (bool rebalance, const List< URI > & uris)` `[virtual]`

Add a URI to the list of URI's that will represent the set of Transports that this **Transport** (p. 2790) is a composite of.

Parameters

<i>rebalance</i>	Indicates if the addition should cause a forced reconnect or not.
<i>uris</i>	The new URI set to add to the set this composite maintains.

Implements `activemq::transport::CompositeTransport` (p. 897).

6.230.2.3 `virtual void activemq::transport::failover::FailoverTransport::close ()` `[virtual]`

Closes this object and deallocates the appropriate resources.

The object is generally no longer usable after calling close.

Exceptions

<i>IOException</i>	if an error occurs while closing.
--------------------	-----------------------------------

Implements `decaf::io::Closeable` (p. 817).

- 6.230.2.4 `long long activemq::transport::failover::FailoverTransport::getBackOffMultiplier () const`
- 6.230.2.5 `int activemq::transport::failover::FailoverTransport::getBackupPoolSize () const`
- 6.230.2.6 `long long activemq::transport::failover::FailoverTransport::getInitialReconnectDelay () const`
- 6.230.2.7 `int activemq::transport::failover::FailoverTransport::getMaxCacheSize () const`
- 6.230.2.8 `int activemq::transport::failover::FailoverTransport::getMaxReconnectAttempts () const`
- 6.230.2.9 `long long activemq::transport::failover::FailoverTransport::getMaxReconnectDelay () const`
- 6.230.2.10 `long long activemq::transport::failover::FailoverTransport::getReconnectDelay () const`
- 6.230.2.11 `virtual std::string activemq::transport::failover::FailoverTransport::getRemoteAddress () const`
[virtual]

Returns

the remote address for this connection

Implements **activemq::transport::Transport** (p. 2792).

- 6.230.2.12 `int activemq::transport::failover::FailoverTransport::getStartupMaxReconnectAttempts () const`
- 6.230.2.13 `long long activemq::transport::failover::FailoverTransport::getTimeout () const`
- 6.230.2.14 `virtual TransportListener* activemq::transport::failover::FailoverTransport::getTransportListener () const`
[virtual]

Gets the observer of asynchronous events from this transport.

Returns

the listener of transport events.

Implements **activemq::transport::Transport** (p. 2792).

6.230.2.15 `virtual Pointer<wireformat::WireFormat> activemq::transport-
::failover::FailoverTransport::getWireFormat () const`
[virtual]

Gets the WireFormat instance that is in use by this transport.

In the case of nested transport this method delegates down to the lowest level transport that actually maintains a WireFormat info instance.

Returns

The WireFormat the object used to encode / decode commands.

Implements **activemq::transport::Transport** (p. 2792).

6.230.2.16 `void activemq::transport::failover::FailoverTransport::handle-
ConnectionControl (const Pointer< Command > & control)`
[protected]

Called when the Broker sends a ConnectionControl command which could signal that this Client needs to reconnect in order to rebalance the connections on a Broker or the set of Known brokers has changed.

Parameters

<i>control</i>	The ConnectionControl command sent from the Broker.
----------------	---

6.230.2.17 `void activemq::transport::failover::FailoverTransport::handle-
TransportFailure (const decaf::lang::Exception & error)`
[protected]

Called when this class' **TransportListener** (p. 2810) is notified of a Failure.

Parameters

<i>error</i>	- The CMS Exception that was thrown.
--------------	--------------------------------------

Exceptions

<i>Exception</i>	if an error occurs.
------------------	---------------------

6.230.2.18 `bool activemq::transport::failover::FailoverTransport::isBackup ()`
const

6.230.2.19 `virtual bool activemq::transport::failover::FailoverTransport::isClosed () const [virtual]`

Has the **Transport** (p. 2790) been shutdown and no longer usable.

Returns

true if the **Transport** (p. 2790)

Implements **activemq::transport::Transport** (p. 2793).

6.230.2.20 `virtual bool activemq::transport::failover::FailoverTransport::isConnected () const [virtual]`

Is the **Transport** (p. 2790) Connected to its Broker.

Returns

true if a connection has been made.

Implements **activemq::transport::Transport** (p. 2793).

6.230.2.21 `virtual bool activemq::transport::failover::FailoverTransport::isFaultTolerant () const [inline, virtual]`

Is this **Transport** (p. 2790) fault tolerant, meaning that it will reconnect to a broker on disconnect.

Returns

true if the **Transport** (p. 2790) is fault tolerant.

Implements **activemq::transport::Transport** (p. 2793).

6.230.2.22 `bool activemq::transport::failover::FailoverTransport::isInitialized () const`

6.230.2.23 `virtual bool activemq::transport::failover::FailoverTransport::isPending () const [virtual]`

Returns

true if there is a need for the iterate method to be called by this classes task runner.

Implements **activemq::threads::CompositeTask** (p. 893).

6.230.2.24 **bool** **activemq::transport::failover::FailoverTransport::isRandomize** ()
const

6.230.2.25 **bool** **activemq::transport::failover::FailoverTransport::isReconnect-Supported** () const [virtual]

Returns

true if reconnect is supported.

Implements **activemq::transport::Transport** (p. 2794).

6.230.2.26 **bool** **activemq::transport::failover::FailoverTransport::isTrackMessages** () const

6.230.2.27 **bool** **activemq::transport::failover::FailoverTransport::isTrack-TransactionProducers** () const

6.230.2.28 **bool** **activemq::transport::failover::FailoverTransport::isUpdateURIs-Supported** () const [virtual]

Returns

true if updating uris is supported.

Implements **activemq::transport::Transport** (p. 2794).

6.230.2.29 **bool** **activemq::transport::failover::FailoverTransport::isUse-ExponentialBackOff** () const

6.230.2.30 **virtual bool** **activemq::transport::failover::FailoverTransport::iterate** ()
[virtual]

Performs the actual Reconnect operation for the **FailoverTransport** (p. 1305), when a connection is made this method returns false to indicate it doesn't need to run again, otherwise it returns true to indicate its still trying to connect.

Returns

false to indicate a connection, true to indicate it needs to try again.

Implements **activemq::threads::Task** (p. 2676).

6.230.2.31 **virtual Transport*** **activemq::transport::failover::-FailoverTransport::narrow** (const std::type_info & *typeid*)
[virtual]

Narrows down a Chain of Transports to a specific **Transport** (p. 2790) to allow a higher level transport to skip intermediate Transports in certain circumstances.

Parameters

<i>typeid</i>	- The type_info of the Object we are searching for.
---------------	---

Returns

the requested Object. or NULL if its not in this chain.

Implements **activemq::transport::Transport** (p. 2794).

6.230.2.32 **virtual void activemq::transport::failover::FailoverTransport::oneway (const Pointer< Command > & command)** [virtual]

Sends a one-way command.

Does not wait for any response from the broker.

Parameters

<i>command</i>	The command to be sent.
----------------	-------------------------

Exceptions

<i>IOException</i>	if an exception occurs during writing of the command.
<i>Unsupported-Operation</i>	if this method is not implemented by this transport.

Implements **activemq::transport::Transport** (p. 2795).

6.230.2.33 **void activemq::transport::failover::FailoverTransport::reconnect (bool rebalance)**

Indicates that the **Transport** (p. 2790) needs to reconnect to another URI in its list.

Parameters

<i>rebalance</i>	Indicates if the current connection should be broken and reconnected.
------------------	---

6.230.2.34 **virtual void activemq::transport::failover::FailoverTransport::reconnect (const decaf::net::URI & uri)** [virtual]

reconnect to another location

Parameters

<i>uri</i>	The new URI to connect this Transport (p. 2790) to.
------------	--

Exceptions

<i>IOException</i>	on failure or if reconnect is not supported.
--------------------	--

Implements **activemq::transport::Transport** (p. 2795).

6.230.2.35 virtual void **activemq::transport::failover::FailoverTransport::removeURI** (bool *rebalance*, const List< URI > & *uris*) [virtual]

Remove a URI from the set of URI's that represents the set of Transports that this **Transport** (p. 2790) is composed of, removing a URI for which the composite has created a connected **Transport** (p. 2790) should result in that **Transport** (p. 2790) being disposed of.

Parameters

<i>rebalance</i>	Indicates if the removal should cause a forced reconnect or not.
<i>uris</i>	The new URI set to remove to the set this composite maintains.

Implements **activemq::transport::CompositeTransport** (p. 897).

6.230.2.36 virtual Pointer<Response> **activemq::transport::failover::FailoverTransport::request** (const Pointer< Command > & *command*) [virtual]

Sends the given command to the broker and then waits for the response.

Parameters

<i>command</i>	the command to be sent.
----------------	-------------------------

Returns

the response from the broker.

Exceptions

<i>IOException</i>	if an exception occurs during the read of the command.
<i>Unsupported-OperationException</i>	if this method is not implemented by this transport.

Implements **activemq::transport::Transport** (p. 2795).

6.230.2.37 **virtual** **Pointer**<**Response**> **activemq::transport::failover::FailoverTransport::request** (**const** **Pointer**< **Command** > & *command*, unsigned int *timeout*) [virtual]

Sends the given command to the broker and then waits for the response.

Parameters

<i>command</i>	The command to be sent.
<i>timeout</i>	The time to wait for this response.

Returns

the response from the broker.

Exceptions

<i>IOException</i>	if an exception occurs during the read of the command.
<i>UnsupportedOperationException</i>	if this method is not implemented by this transport.

Implements **activemq::transport::Transport** (p. 2796).

6.230.2.38 **void** **activemq::transport::failover::FailoverTransport::restoreTransport** (**const** **Pointer**< **Transport** > & *transport*) [protected]

Given a **Transport** (p. 2790) restore the state of the Client's connection to the Broker using the data accumulated in the State Tracker.

Parameters

<i>transport</i>	The new Transport (p. 2790) connected to the Broker.
------------------	---

Exceptions

<i>IOException</i>	if an errors occurs while restoring the old state.
--------------------	--

6.230.2.39 **void** **activemq::transport::failover::FailoverTransport::setBackOffMultiplier** (long long *value*)

6.230.2.40 **void** **activemq::transport::failover::FailoverTransport::setBackup** (bool *value*)

6.230.2.41 **void** **activemq::transport::failover::FailoverTransport::setBackupPoolSize** (int *value*)

- 6.230.2.42 void activemq::transport::failover::FailoverTransport::set-ConnectionInterruptProcessingComplete (const Pointer< commands::ConnectionId > & *connectionId*)
- 6.230.2.43 void activemq::transport::failover::FailoverTransport::setInitialized (bool *value*)
- 6.230.2.44 void activemq::transport::failover::FailoverTransport::setInitial-ReconnectDelay (long long *value*)
- 6.230.2.45 void activemq::transport::failover::FailoverTransport::setMaxCache-Size (int *value*)
- 6.230.2.46 void activemq::transport::failover::FailoverTransport::setMax-ReconnectAttempts (int *value*)
- 6.230.2.47 void activemq::transport::failover::FailoverTransport::setMax-ReconnectDelay (long long *value*)
- 6.230.2.48 void activemq::transport::failover::FailoverTransport::setRandomize (bool *value*)
- 6.230.2.49 void activemq::transport::failover::FailoverTransport::setReconnect-Delay (long long *value*)
- 6.230.2.50 void activemq::transport::failover::FailoverTransport::setReconnect-Supported (bool *value*)
- 6.230.2.51 void activemq::transport::failover::FailoverTransport::setStartupMaxReconnectAttempts (int *value*)
- 6.230.2.52 void activemq::transport::failover::FailoverTransport::setTimeout (long long *value*)
- 6.230.2.53 void activemq::transport::failover::FailoverTransport::setTrack-Messages (bool *value*)
- 6.230.2.54 void activemq::transport::failover::FailoverTransport::setTrackTransactionProducers (bool *value*)
- 6.230.2.55 virtual void activemq::transport::failover::FailoverTransport::setTransportListener (TransportListener * *listener*)
[virtual]

Sets the observer of asynchronous events from this transport.

Parameters

<i>listener</i>	the listener of transport events.
-----------------	-----------------------------------

Implements **activemq::transport::Transport** (p. 2797).

6.230.2.56 **void activemq::transport::failover::FailoverTransport::setUpdateURIs-Supported (bool *value*)**

6.230.2.57 **void activemq::transport::failover::FailoverTransport::setUse-ExponentialBackOff (bool *value*)**

6.230.2.58 **virtual void activemq::transport::failover::FailoverTransport::set-WireFormat (const Pointer< wireformat::WireFormat > &wireFormat *AMQCPP_UNUSED*)** [inline, virtual]

6.230.2.59 **virtual void activemq::transport::failover::FailoverTransport::start ()** [virtual]

Starts the **Transport** (p. 2790), the send methods of a **Transport** (p. 2790) will throw an exception if used before the **Transport** (p. 2790) is started.

Exceptions

<i>IOException</i>	if an error occurs while starting the Transport (p. 2790).
--------------------	---

Implements **activemq::transport::Transport** (p. 2797).

6.230.2.60 **virtual void activemq::transport::failover::FailoverTransport::stop ()** [virtual]

Stops the **Transport** (p. 2790).

Exceptions

<i>IOException</i>	if an error occurs while stopping the transport.
--------------------	--

Implements **activemq::transport::Transport** (p. 2797).

6.230.2.61 **virtual void activemq::transport::failover::FailoverTransport::updateURIs (bool *rebalance*, const decaf::util::List< decaf::net::URI > &uris)** [virtual]

Updates the set of URIs the **Transport** (p. 2790) can connect to.

If the **Transport** (p. 2790) doesn't support updating its URIs then an **IOException** is thrown.

6.231 activemq::transport::failover::FailoverTransportFactory Class Reference

Parameters

<i>rebalance</i>	Indicates if a forced reconnection should be performed as a result of the update.
<i>uris</i>	The new list of URIs that can be used for connection.

Exceptions

<i>IOException</i>	if an error occurs or updates aren't supported.
--------------------	---

Implements **activemq::transport::Transport** (p. 2798).

6.230.3 Friends And Related Function Documentation

6.230.3.1 friend class FailoverTransportListener [friend]

The documentation for this class was generated from the following file:

- src/main/activemq/transport/failover/**FailoverTransport.h**

6.231 activemq::transport::failover::FailoverTransportFactory - Class Reference

Creates an instance of a **FailoverTransport** (p. 1305).

```
#include <src/main/activemq/transport/failover/Failover-TransportFactory.h>
```

Inheritance diagram for activemq::transport::failover::FailoverTransportFactory:

Public Member Functions

- virtual **~FailoverTransportFactory** ()
- virtual **Pointer< Transport > create** (const **decaf::net::URI** &location)
*Creates a fully configured **Transport** (p. 2790) instance which could be a chain of filters and transports.*
- virtual **Pointer< Transport > createComposite** (const **decaf::net::URI** &location)
*Creates a slimmed down **Transport** (p. 2790) instance which can be used in composite transport instances.*

Protected Member Functions

- virtual **Pointer**< **Transport** > **doCreateComposite** (const **decaf::net::URI** &location, const **decaf::util::Properties** &properties)

*Creates a slimmed down **Transport** (p. 2790) instance which can be used in composite transport instances.*

6.231.1 Detailed Description

Creates an instance of a **FailoverTransport** (p. 1305).

Since

3.0

6.231.2 Constructor & Destructor Documentation

6.231.2.1 virtual **activemq::transport::failover::FailoverTransportFactory::~FailoverTransportFactory** () [**inline**, **virtual**]

6.231.3 Member Function Documentation

6.231.3.1 virtual **Pointer**<**Transport**> **activemq::transport::failover::FailoverTransportFactory::create** (const **decaf::net::URI** &location) [**virtual**]

Creates a fully configured **Transport** (p. 2790) instance which could be a chain of filters and transports.

Parameters

<i>location</i>	- URI location to connect to plus any properties to assign.
-----------------	---

Exceptions

<i>ActiveMQException</i>	if an error occurs
--------------------------	--------------------

Implements **activemq::transport::TransportFactory** (p. 2799).

6.231.3.2 virtual **Pointer**<**Transport**> **activemq::transport::failover::FailoverTransportFactory::createComposite** (const **decaf::net::URI** &location) [**virtual**]

Creates a slimmed down **Transport** (p. 2790) instance which can be used in composite transport instances.

6.232 activemq::transport::failover::FailoverTransportListener Class Reference

Parameters

<i>location</i>	- URI location to connect to plus any properties to assign.
-----------------	---

Exceptions

<i>ActiveMQException</i>	if an error occurs
--------------------------	--------------------

Implements **activemq::transport::TransportFactory** (p. 2800).

6.231.3.3 virtual **Pointer<Transport>** **activemq::transport::failover::FailoverTransportFactory::doCreateComposite** (**const decaf::net::URI & location**, **const decaf::util::Properties & properties**) [protected, virtual]

Creates a slimed down **Transport** (p. 2790) instance which can be used in composite transport instances.

Parameters

<i>location</i>	- URI location to connect to.
<i>properties</i>	- Properties to apply to the transport.

Returns

Pointer to a new **FailoverTransport** (p. 1305) instance.

Exceptions

<i>ActiveMQException</i>	if an error occurs
--------------------------	--------------------

The documentation for this class was generated from the following file:

- src/main/activemq/transport/failover/**FailoverTransportFactory.h**

6.232 activemq::transport::failover::FailoverTransportListener - Class Reference

Utility class used by the **Transport** (p. 2790) to perform the work of responding to events from the active **Transport** (p. 2790).

```
#include <src/main/activemq/transport/failover/Failover-TransportListener.h>
```

Inheritance diagram for **activemq::transport::failover::FailoverTransportListener**:

Public Member Functions

- **FailoverTransportListener** (**FailoverTransport** *parent)
- virtual **~FailoverTransportListener** ()
- virtual void **onCommand** (const **Pointer**< **Command** > &command)
Event handler for the receipt of a command.
- virtual void **onException** (const **decaf::lang::Exception** &ex)
Event handler for an exception from a command transport.
- virtual void **transportInterrupted** ()
The transport has suffered an interruption from which it hopes to recover.
- virtual void **transportResumed** ()
The transport has resumed after an interruption.

6.232.1 Detailed Description

Utility class used by the **Transport** (p. 2790) to perform the work of responding to events from the active **Transport** (p. 2790).

Since

3.0

6.232.2 Constructor & Destructor Documentation

6.232.2.1 **activemq::transport::failover::FailoverTransportListener::FailoverTransportListener** (**FailoverTransport** * parent)

6.232.2.2 **virtual activemq::transport::failover::FailoverTransportListener::~~FailoverTransportListener** ()
 [virtual]

6.232.3 Member Function Documentation

6.232.3.1 **virtual void activemq::transport::failover::FailoverTransportListener::onCommand** (const **Pointer**< **Command** > & command)
 [virtual]

Event handler for the receipt of a command.

The transport passes off all received commands to its listeners, the listener then owns the Object. If there is no registered listener the **Transport** (p. 2790) deletes the command upon receipt.

Parameters

<i>command</i>	the received command object.
----------------	------------------------------

Implements `activemq::transport::TransportListener` (p. 2811).

6.232.3.2 `virtual void activemq::transport::failover::FailoverTransportListener::onException (const decaf::lang::Exception & ex)`
[virtual]

Event handler for an exception from a command transport.

Parameters

<code>ex</code>	The exception.
-----------------	----------------

Implements `activemq::transport::TransportListener` (p. 2811).

6.232.3.3 `virtual void activemq::transport::failover::FailoverTransportListener::transportInterrupted ()` [virtual]

The transport has suffered an interruption from which it hopes to recover.

Implements `activemq::transport::TransportListener` (p. 2812).

6.232.3.4 `virtual void activemq::transport::failover::FailoverTransportListener::transportResumed ()` [virtual]

The transport has resumed after an interruption.

Implements `activemq::transport::TransportListener` (p. 2812).

The documentation for this class was generated from the following file:

- `src/main/activemq/transport/failover/FailoverTransportListener.h`

6.233 `activemq::core::FifoMessageDispatchChannel` Class - Reference

```
#include <src/main/activemq/core/FifoMessageDispatchChannel.h>
```

Inheritance diagram for `activemq::core::FifoMessageDispatchChannel`:

Public Member Functions

- `FifoMessageDispatchChannel ()`
- `virtual ~FifoMessageDispatchChannel ()`

- virtual void **enqueue** (const **Pointer**< **MessageDispatch** > &message)
Add a Message to the Channel behind all pending message.
- virtual void **enqueueFirst** (const **Pointer**< **MessageDispatch** > &message)
Add a message to the front of the Channel.
- virtual bool **isEmpty** () const
- virtual bool **isClosed** () const
- virtual bool **isRunning** () const
- virtual **Pointer**< **MessageDispatch** > **dequeue** (long long timeout)
Used to get an enqueued message.
- virtual **Pointer**< **MessageDispatch** > **dequeueNowait** ()
Used to get an enqueued message if there is one queued right now.
- virtual **Pointer**< **MessageDispatch** > **peek** () const
Peek in the Queue and return the first message in the Channel without removing it from the channel.
- virtual void **start** ()
Starts dispatch of messages from the Channel.
- virtual void **stop** ()
Stops dispatch of message from the Channel.
- virtual void **close** ()
Close this channel no messages will be dispatched after this method is called.
- virtual void **clear** ()
Clear the Channel, all pending messages are removed.
- virtual int **size** () const
- virtual std::vector< **Pointer** < **MessageDispatch** > > **removeAll** ()
Remove all messages that are currently in the Channel and return them as a list of Messages.
- virtual void **lock** () throw (decaf::lang::exceptions::RuntimeException)
Locks the object.
- virtual bool **tryLock** () throw (decaf::lang::exceptions::RuntimeException)
Attempts to Lock the object, if the lock is already held by another thread than this method returns false.
- virtual void **unlock** () throw (decaf::lang::exceptions::RuntimeException)
Unlocks the object.
- virtual void **wait** () throw (decaf::lang::exceptions::RuntimeException, decaf::lang::exceptions::IllegalMonitorStateException, decaf::lang::exceptions::InterruptedException)
Waits on a signal from this object, which is generated by a call to Notify.
- virtual void **wait** (long long millisecs) throw (decaf::lang::exceptions::RuntimeException, decaf::lang::exceptions::IllegalMonitorStateException, decaf::lang::exceptions::InterruptedException)
Waits on a signal from this object, which is generated by a call to Notify.
- virtual void **wait** (long long millisecs, int nanos) throw (decaf::lang::exceptions::RuntimeException, decaf::lang::exceptions::IllegalArgumentException, decaf::lang::exceptions::IllegalMonitorStateException, decaf::lang::exceptions::InterruptedException)

Waits on a signal from this object, which is generated by a call to Notify.

- virtual void **notify** () throw (decaf::lang::exceptions::RuntimeException, decaf::lang::exceptions::IllegalMonitorStateException)

Signals a waiter on this object that it can now wake up and continue.

- virtual void **notifyAll** () throw (decaf::lang::exceptions::RuntimeException, decaf::lang::exceptions::IllegalMonitorStateException)

Signals the waiters on this object that it can now wake up and continue.

6.233.1 Constructor & Destructor Documentation

6.233.1.1 **activemq::core::FifoMessageDispatchChannel::FifoMessageDispatchChannel ()**

6.233.1.2 **virtual activemq::core::FifoMessageDispatchChannel::~~FifoMessageDispatchChannel ()** [virtual]

6.233.2 Member Function Documentation

6.233.2.1 **virtual void activemq::core::FifoMessageDispatchChannel::clear ()**
[virtual]

Clear the Channel, all pending messages are removed.

Implements **activemq::core::MessageDispatchChannel** (p. 1887).

6.233.2.2 **virtual void activemq::core::FifoMessageDispatchChannel::close ()**
[virtual]

Close this channel no messages will be dispatched after this method is called.

Implements **activemq::core::MessageDispatchChannel** (p. 1887).

6.233.2.3 **virtual Pointer<MessageDispatch> activemq::core::FifoMessageDispatchChannel::dequeue (long long timeout)**
[virtual]

Used to get an enqueued message.

The amount of time this method blocks is based on the timeout value. - if timeout==-1 then it blocks until a message is received. - if timeout==0 then it tries to not block at all, it returns a message if it is available - if timeout>0 then it blocks up to timeout amount of time. Expired messages will be consumed by this method.

Returns

null if we timeout or if the consumer is closed.

Exceptions

<i>ActiveMQException</i>

Implements **activemq::core::MessageDispatchChannel** (p. 1887).

6.233.2.4 **virtual Pointer<MessageDispatch> activemq::core::FifoMessageDispatchChannel::dequeueNoWait ()**
[virtual]

Used to get an enqueued message if there is one queued right now.

If there is no waiting message than this method returns Null.

Returns

a message if there is one in the queue.

Implements **activemq::core::MessageDispatchChannel** (p. 1888).

6.233.2.5 **virtual void activemq::core::FifoMessageDispatchChannel::enqueue (**
const Pointer< MessageDispatch > & message) [virtual]

Add a Message to the Channel behind all pending message.

Parameters

<i>message</i>	- The message to add to the Channel.
----------------	--------------------------------------

Implements **activemq::core::MessageDispatchChannel** (p. 1888).

6.233.2.6 **virtual void activemq::core::FifoMessageDispatchChannel::enqueueFirst**
(const Pointer< MessageDispatch > & message) [virtual]

Add a message to the front of the Channel.

Parameters

<i>message</i>	- The Message to add to the front of the Channel.
----------------	---

Implements **activemq::core::MessageDispatchChannel** (p. 1888).

6.233.2.7 **virtual bool activemq::core::FifoMessageDispatchChannel::isClosed ()**
const [inline, virtual]

Returns

has the Queue been closed.

Implements **activemq::core::MessageDispatchChannel** (p. 1888).

6.233.2.8 `virtual bool activemq::core::FifoMessageDispatchChannel::isEmpty ()`
`const [virtual]`

Returns

true if there are no messages in the Channel.

Implements **activemq::core::MessageDispatchChannel** (p. 1889).

6.233.2.9 `virtual bool activemq::core::FifoMessageDispatchChannel::isRunning ()`
`const [inline, virtual]`

Returns

true if the Channel currently running and will dequeue message.

Implements **activemq::core::MessageDispatchChannel** (p. 1889).

6.233.2.10 `virtual void activemq::core::FifoMessageDispatchChannel::lock ()`
`throw (decaf::lang::exceptions::RuntimeException) [inline, virtual]`

Locks the object.

Exceptions

<i>RuntimeException</i>	if an error occurs while locking the object.
-------------------------	--

Implements **decaf::util::concurrent::Synchronizable** (p. 2640).

6.233.2.11 `virtual void activemq::core::FifoMessageDispatchChannel::notify`
`() throw (decaf::lang::exceptions::RuntimeException,`
`decaf::lang::exceptions::IllegalMonitorStateException) [inline,`
`virtual]`

Signals a waiter on this object that it can now wake up and continue.

Must have this object locked before calling.

Exceptions

<i>IllegalMonitorStateException</i>	- if the current thread is not the owner of the the Synchronizable Object.
<i>RuntimeException</i>	if an error occurs while notifying one of the waiting threads.

Implements **decaf::util::concurrent::Synchronizable** (p. 2641).

6.233.2.12 `virtual void activemq::core::FifoMessageDispatchChannel::notifyAll
() throw (decaf::lang::exceptions::RuntimeException,
decaf::lang::exceptions::IllegalMonitorStateException) [inline,
virtual]`

Signals the waiters on this object that it can now wake up and continue.

Must have this object locked before calling.

Exceptions

<i>IllegalMonitorStateException</i>	- if the current thread is not the owner of the the Synchronizable Object.
<i>RuntimeException</i>	if an error occurs while notifying the waiting threads.

Implements **decaf::util::concurrent::Synchronizable** (p. 2642).

6.233.2.13 `virtual Pointer<MessageDispatch> activemq::core-
::FifoMessageDispatchChannel::peek () const
[virtual]`

Peek in the Queue and return the first message in the Channel without removing it from the channel.

Returns

a message if there is one in the queue.

Implements **activemq::core::MessageDispatchChannel** (p. 1889).

6.233.2.14 `virtual std::vector< Pointer<MessageDispatch> >
activemq::core::FifoMessageDispatchChannel::removeAll ()
[virtual]`

Remove all messages that are currently in the Channel and return them as a list of Messages.

Returns

a list of Messages that was previously in the Channel.

Implements **activemq::core::MessageDispatchChannel** (p. 1889).

6.233.2.15 `virtual int activemq::core::FifoMessageDispatchChannel::size () const
[virtual]`

Returns

the number of Messages currently in the Channel.

Implements **activemq::core::MessageDispatchChannel** (p. 1890).

6.233.2.16 `virtual void activemq::core::FifoMessageDispatchChannel::start ()`
[virtual]

Starts dispatch of messages from the Channel.

Implements **activemq::core::MessageDispatchChannel** (p. 1890).

6.233.2.17 `virtual void activemq::core::FifoMessageDispatchChannel::stop ()`
[virtual]

Stops dispatch of message from the Channel.

Implements **activemq::core::MessageDispatchChannel** (p. 1890).

6.233.2.18 `virtual bool activemq::core::FifoMessageDispatchChannel::tryLock (`
`) throw (decaf::lang::exceptions::RuntimeException) [inline,`
`virtual]`

Attempts to Lock the object, if the lock is already held by another thread than this method returns false.

Returns

true if the lock was acquired, false if it is already held by another thread.

Exceptions

<i>RuntimeException</i>	if an error occurs while locking the object.
-------------------------	--

Implements **decaf::util::concurrent::Synchronizable** (p. 2643).

6.233.2.19 `virtual void activemq::core::FifoMessageDispatchChannel::unlock (`
`) throw (decaf::lang::exceptions::RuntimeException) [inline,`
`virtual]`

Unlocks the object.

Exceptions

<i>RuntimeException</i>	if an error occurs while unlocking the object.
-------------------------	--

Implements **decaf::util::concurrent::Synchronizable** (p. 2645).

6.233.2.20 `virtual void activemq::core::FifoMessageDispatchChannel::wait () throw (decaf::lang::exceptions::RuntimeException, decaf::lang::exceptions::IllegalMonitorStateException, decaf::lang::exceptions::InterruptedException) [inline, virtual]`

Waits on a signal from this object, which is generated by a call to Notify.

Must have this object locked before calling.

Exceptions

<i>RuntimeException</i>	if an error occurs while waiting on the object.
<i>InterruptedException</i>	if the wait is interrupted before it completes.
<i>IllegalMonitorStateException</i>	- if the current thread is not the owner of the the Synchronizable Object.

Implements **decaf::util::concurrent::Synchronizable** (p. 2646).

6.233.2.21 `virtual void activemq::core::FifoMessageDispatchChannel::wait (long long millisecs) throw (decaf::lang::exceptions::RuntimeException, decaf::lang::exceptions::IllegalMonitorStateException, decaf::lang::exceptions::InterruptedException) [inline, virtual]`

Waits on a signal from this object, which is generated by a call to Notify.

Must have this object locked before calling. This wait will timeout after the specified time interval.

Parameters

<i>millisecs</i>	the time in milliseconds to wait, or WAIT_INIFINITE
------------------	---

Exceptions

<i>RuntimeException</i>	if an error occurs while waiting on the object.
<i>InterruptedException</i>	if the wait is interrupted before it completes.
<i>IllegalMonitorStateException</i>	- if the current thread is not the owner of the the Synchronizable Object.

Implements **decaf::util::concurrent::Synchronizable** (p. 2647).

```
6.233.2.22 virtual void activemq::core::FifoMessageDispatchChannel::wait ( long
long millisecs, int nanos ) throw ( decaf::lang::exceptions::Runtime-
Exception, decaf::lang::exceptions::IllegalArgumentException,
decaf::lang::exceptions::IllegalMonitorStateException,
decaf::lang::exceptions::InterruptedException ) [inline,
virtual]
```

Waits on a signal from this object, which is generated by a call to Notify.

Must have this object locked before calling. This wait will timeout after the specified time interval. This method is similar to the one argument wait function except that it add a finer grained control over the amount of time that it waits by adding in the additional nanosecond argument.

NOTE: The ability to wait accurately at a nanosecond scale depends on the platform and OS that the Decaf API is running on, some systems do not provide an accurate enough clock to provide this level of granularity.

Parameters

<i>millisecs</i>	the time in milliseconds to wait, or WAIT_INFINITE
<i>nanos</i>	additional time in nanoseconds with a range of 0-999999

Exceptions

<i>IllegalArgumentException</i>	if an error occurs or the nanos argument is not in the range of [0-999999]
<i>RuntimeException</i>	if an error occurs while waiting on the object.
<i>InterruptedException</i>	if the wait is interrupted before it completes.
<i>IllegalMonitorStateException</i>	- if the current thread is not the owner of the the Synchronizable Object.

Implements **decaf::util::concurrent::Synchronizable** (p. 2648).

The documentation for this class was generated from the following file:

- src/main/activemq/core/**FifoMessageDispatchChannel.h**

6.234 decaf::io::FileDescriptor Class Reference

This class servers as an opaque wrapper around an underlying OS level resource that can be used as a source / sink for byte level data such as sockets and files.

```
#include <src/main/decaf/io/FileDescriptor.h>
```

Inheritance diagram for decaf::io::FileDescriptor:

Public Member Functions

- **FileDescriptor** ()
- virtual **~FileDescriptor** ()
- void **sync** ()

*Force any/all buffered data for this **FileDescriptor** (p. 1330) to be flushed to the underlying device.*

- bool **valid** ()

Indicates whether the File Descriptor is valid.

Static Public Attributes

- static **FileDescriptor in**
A handle to the standard input stream.
- static **FileDescriptor out**
A handle to the standard output stream.
- static **FileDescriptor err**
A handle to the standard error stream.

Protected Member Functions

- **FileDescriptor** (long value, bool **readonly**)

Protected Attributes

- long **descriptor**
- bool **readonly**

6.234.1 Detailed Description

This class servers as an opaque wrapper around an underlying OS level resource that can be used as a source / sink for byte level data such as sockets and files.

Since

1.0

6.234.2 Constructor & Destructor Documentation

6.234.2.1 **decaf::io::FileDescriptor::FileDescriptor** (long value, bool **readonly**)
[protected]

6.234.2.2 **decaf::io::FileDescriptor::FileDescriptor** ()

6.234.2.3 virtual **decaf::io::FileDescriptor::~FileDescriptor**() [virtual]

6.234.3 Member Function Documentation

6.234.3.1 void **decaf::io::FileDescriptor::sync**()

Force any/all buffered data for this **FileDescriptor** (p. 1330) to be flushed to the underlying device.

This method blocks until all data is flushed to the underlying device and is used to place the device into a known state. In the case of data that is buffered at a higher level such as a **BufferedOutputStream** (p. 595) the stream must first be flushed before this method can force the data to be sent to the output device.

6.234.3.2 bool **decaf::io::FileDescriptor::valid**()

Indicates whether the File Descriptor is valid.

Returns

true for a valid descriptor such as open socket or file, false otherwise.

6.234.4 Field Documentation

6.234.4.1 long **decaf::io::FileDescriptor::descriptor** [protected]

6.234.4.2 **FileDescriptor decaf::io::FileDescriptor::err** [static]

A handle to the standard error stream.

Usually, this file descriptor is not used directly, but rather via the output stream known as **System::err**.

6.234.4.3 **FileDescriptor decaf::io::FileDescriptor::in** [static]

A handle to the standard input stream.

Usually, this file descriptor is not used directly, but rather via the input stream known as **System::in**.

6.234.4.4 **FileDescriptor decaf::io::FileDescriptor::out** [static]

A handle to the standard output stream.

Usually, this file descriptor is not used directly, but rather via the output stream known as **System::out**.

6.234.4.5 `bool decaf::io::FileDescriptor::readonly` `[protected]`

The documentation for this class was generated from the following file:

- `src/main/decaf/io/FileDescriptor.h`

6.235 `decaf::util::logging::Filter` Class Reference

A **Filter** (p. 1333) can be used to provide fine grain control over what is logged, beyond the control provided by log levels.

```
#include <src/main/decaf/util/logging/Filter.h>
```

Public Member Functions

- virtual `~Filter()`
- virtual `bool isLoggable (const LogRecord &record) const =0`
Check if a given log record should be published.

6.235.1 Detailed Description

A **Filter** (p. 1333) can be used to provide fine grain control over what is logged, beyond the control provided by log levels.

Each **Logger** (p. 1693) and each **Handler** (p. 1401) can have a filter associated with it. The **Logger** (p. 1693) or **Handler** (p. 1401) will call the `isLoggable` method to check if a given **LogRecord** (p. 1719) should be published. If `isLoggable` returns false, the **LogRecord** (p. 1719) will be discarded.

6.235.2 Constructor & Destructor Documentation

6.235.2.1 `virtual decaf::util::logging::Filter::~Filter ()` `[inline, virtual]`

6.235.3 Member Function Documentation

6.235.3.1 `virtual bool decaf::util::logging::Filter::isLoggable (const LogRecord &record) const` `[pure virtual]`

Check if a given log record should be published.

Parameters

<i>record</i>	the LogRecord (p. 1719) to check.
---------------	--

Returns

true if the record is loggable.

The documentation for this class was generated from the following file:

- src/main/decaf/util/logging/**Filter.h**

6.236 decaf::io::FilterInputStream Class Reference

A **FilterInputStream** (p. 1334) contains some other input stream, which it uses as its basic source of data, possibly transforming the data along the way or providing additional functionality.

```
#include <src/main/decaf/io/FilterInputStream.h>
```

Inheritance diagram for decaf::io::FilterInputStream:

Public Member Functions

- **FilterInputStream** (**InputStream** *inputStream, bool own=false)

*Constructor to create a wrapped **InputStream** (p. 1464).*

- virtual ~**FilterInputStream** ()
- virtual int **available** () const

Indicates the number of bytes available.

The default implementation of this methods returns 0. Classes that override this method may return the total number of bytes that are currently available for reading and others may simply return a value of one indicating that there is some data available. The caller should view the result of this method as an absolute.

The default implementation of this method returns zero.

Returns

the number of bytes available on this input stream.

Exceptions

IOException (p. 1545)	<i>if an I/O error occurs.</i>
------------------------------	--------------------------------

- virtual void **close** ()

*Closes the **InputStream** (p. 1464) freeing any resources that might have been acquired during the lifetime of this stream.*

The default implementation of this method does nothing.

Exceptions

IOException (p. 1545)	<i>if an I/O error occurs while closing the InputStream (p. 1464).</i>
------------------------------	---

- virtual long long **skip** (long long num)

Skips over and discards n bytes of data from this input stream.

The skip method may, for a variety of reasons, end up skipping over some smaller

number of bytes, possibly 0. This may result from any of a number of conditions; reaching end of file before *n* bytes have been skipped is only one possibility. The actual number of bytes skipped is returned.

The skip method of **InputStream** (p. 1464) creates a byte array and then repeatedly reads into it until *num* bytes have been read or the end of the stream has been reached. Subclasses are encouraged to provide a more efficient implementation of this method.

Parameters

num	The number of bytes to skip.
-----	------------------------------

Returns

total bytes skipped

Exceptions

IOException (p. 1545)	if an I/O error occurs.
UnsupportedOperationException	if the concrete stream class does not support skipping bytes.

- virtual void **mark** (int readLimit)

Marks the current position in the stream. A subsequent call to the reset method repositions this stream at the last marked position so that subsequent reads re-read the same bytes.

If a stream instance reports that marks are supported then the stream will ensure that the same bytes can be read again after the reset method is called so long the readLimit is not reached.

Calling mark on a closed stream instance should have no effect.

The default implementation of this method does nothing.

Parameters

readLimit	The max bytes read before marked position is invalid.
-----------	---

- virtual void **reset** ()

Repositions this stream to the position at the time the mark method was last called on this input stream.

If the method markSupported returns true, then: * If the method mark has not been called since the stream was created, or the number of bytes read from the stream since mark was last called is larger than the argument to mark at that last call, then an **IOException** (p. 1545) might be thrown. * If such an **IOException** (p. 1545) is not thrown, then the stream is reset to a state such that all the bytes read since the most recent call to mark (or since the start of the file, if mark has not been called) will be resupplied to subsequent callers of the read method, followed by any bytes that otherwise would have been the next input data as of the time of the call to reset.

If the method markSupported returns false, then: * The call to reset may throw an **IOException** (p. 1545). * If an **IOException** (p. 1545) is not thrown, then the stream is reset to a fixed state that depends on the particular type of the input stream and how it was created. The bytes that will be supplied to subsequent callers of the read method depend on the particular type of the input stream.

The default implementation of this method throws an **IOException** (p. 1545).

Exceptions

IOException (p. 1545)	if an I/O error occurs.
------------------------------	-------------------------

- virtual bool **markSupported** () const

Determines if this input stream supports the mark and reset methods.

Whether or not mark and reset are supported is an invariant property of a particular

input stream instance.

The default implementation of this method returns false.

Returns

true if this stream instance supports marks

Protected Member Functions

- virtual int **doReadByte** ()
- virtual int **doReadArray** (unsigned char *buffer, int size)
- virtual int **doReadArrayBounded** (unsigned char *buffer, int size, int offset, int length)
- virtual bool **isClosed** () const

Protected Attributes

- **InputStream** * **inputStream**
- bool **own**
- volatile bool **closed**

6.236.1 Detailed Description

A **FilterInputStream** (p. 1334) contains some other input stream, which it uses as its basic source of data, possibly transforming the data along the way or providing additional functionality.

The class **FilterInputStream** (p. 1334) itself simply overrides all methods of **InputStream** (p. 1464) with versions that pass all requests to the contained input stream. Subclasses of **FilterInputStream** (p. 1334) may further override some of these methods and may also provide additional methods and fields.

6.236.2 Constructor & Destructor Documentation

6.236.2.1 **decaf::io::FilterInputStream::FilterInputStream** (**InputStream** * *inputStream*, bool *own* = false)

Constructor to create a wrapped **InputStream** (p. 1464).

Parameters

<i>inputStream</i>	The stream to wrap and filter.
<i>own</i>	Indicates if we own the stream object, defaults to false.

6.236.2.2 virtual **decaf::io::FilterInputStream::~~FilterInputStream** ()
[virtual]

6.236.3 Member Function Documentation

6.236.3.1 `virtual int decaf::io::FilterInputStream::available () const` [virtual]

Indicates the number of bytes available.

The default implementation of this methods returns 0. Classes that override this method may return the total number of bytes that are currently available for reading and others may simply return a value of one indicating that there is some data available. The caller should view the result of this method as an absolute.

The default implementation of this method returns zero.

Returns

the number of bytes available on this input stream.

Exceptions

<i>IOException</i> (p. 1545)	if an I/O error occurs.
--	-------------------------

Reimplemented from **decaf::io::InputStream** (p. 1466).

Reimplemented in **decaf::io::PushbackInputStream** (p. 2217), **decaf::util::zip::InflaterInputStream** (p. 1460), and **decaf::io::BufferedInputStream** (p. 592).

6.236.3.2 `virtual void decaf::io::FilterInputStream::close ()` [virtual]

Closes the **InputStream** (p. 1464) freeing any resources that might have been acquired during the lifetime of this stream.

The default implementation of this method does nothing.

Exceptions

<i>IOException</i> (p. 1545)	if an I/O error occurs while closing the InputStream (p. 1464).
--	--

Reimplemented from **decaf::io::InputStream** (p. 1466).

Reimplemented in **decaf::util::zip::InflaterInputStream** (p. 1461), and **decaf::io::BufferedInputStream** (p. 592).

6.236.3.3 `virtual int decaf::io::FilterInputStream::doReadArray (unsigned char * buffer, int size)` [protected, virtual]

Reimplemented from **decaf::io::InputStream** (p. 1467).

6.236.3.4 `virtual int decaf::io::FilterInputStream::doReadArrayBounded (unsigned char * buffer, int size, int offset, int length)` [protected, virtual]

Reimplemented from `decaf::io::InputStream` (p. 1467).

Reimplemented in `decaf::util::zip::InflaterInputStream` (p. 1461), `decaf::io::PushbackInputStream` (p. 2217), `decaf::io::BufferedInputStream` (p. 593), `decaf::util::zip::CheckedInputStream` (p. 807), and `activemq::io::LoggingInputStream` (p. 1707).

6.236.3.5 `virtual int decaf::io::FilterInputStream::doReadByte ()` [protected, virtual]

Implements `decaf::io::InputStream` (p. 1467).

Reimplemented in `decaf::util::zip::InflaterInputStream` (p. 1461), `decaf::io::PushbackInputStream` (p. 2218), `decaf::io::BufferedInputStream` (p. 593), `decaf::util::zip::CheckedInputStream` (p. 807), and `activemq::io::LoggingInputStream` (p. 1707).

6.236.3.6 `virtual bool decaf::io::FilterInputStream::isClosed () const` [protected, virtual]

Returns

true if this stream has been closed.

6.236.3.7 `virtual void decaf::io::FilterInputStream::mark (int readLimit)` [virtual]

Marks the current position in the stream A subsequent call to the reset method repositions this stream at the last marked position so that subsequent reads re-read the same bytes.

If a stream instance reports that marks are supported then the stream will ensure that the same bytes can be read again after the reset method is called so long the readLimit is not reached.

Calling mark on a closed stream instance should have no effect.

The default implementation of this method does nothing.

Parameters

<i>readLimit</i>	The max bytes read before marked position is invalid.
------------------	---

Reimplemented from `decaf::io::InputStream` (p. 1468).

Reimplemented in `decaf::io::PushbackInputStream` (p. 2218), `decaf::util::zip::InflaterInputStream` (p. 1462), and `decaf::io::BufferedInputStream` (p. 593).

6.236.3.8 `virtual bool decaf::io::FilterInputStream::markSupported () const`
`[virtual]`

Determines if this input stream supports the mark and reset methods.

Whether or not mark and reset are supported is an invariant property of a particular input stream instance.

The default implementation of this method returns false.

Returns

true if this stream instance supports marks

Reimplemented from `decaf::io::InputStream` (p. 1468).

Reimplemented in `decaf::io::PushbackInputStream` (p. 2218), `decaf::util::zip::InflaterInputStream` (p. 1462), and `decaf::io::BufferedInputStream` (p. 593).

6.236.3.9 `virtual void decaf::io::FilterInputStream::reset ()` `[virtual]`

Repositions this stream to the position at the time the mark method was last called on this input stream.

If the method `markSupported` returns true, then: * If the method `mark` has not been called since the stream was created, or the number of bytes read from the stream since mark was last called is larger than the argument to mark at that last call, then an **IOException** (p. 1545) might be thrown. * If such an **IOException** (p. 1545) is not thrown, then the stream is reset to a state such that all the bytes read since the most recent call to mark (or since the start of the file, if mark has not been called) will be resupplied to subsequent callers of the read method, followed by any bytes that otherwise would have been the next input data as of the time of the call to reset.

If the method `markSupported` returns false, then: * The call to reset may throw an **IOException** (p. 1545). * If an **IOException** (p. 1545) is not thrown, then the stream is reset to a fixed state that depends on the particular type of the input stream and how it was created. The bytes that will be supplied to subsequent callers of the read method depend on the particular type of the input stream.

The default implementation of this method throws an **IOException** (p. 1545).

Exceptions

IOException (p. 1545)	if an I/O error occurs.
---------------------------------	-------------------------

Reimplemented from `decaf::io::InputStream` (p. 1471).

Reimplemented in `decaf::io::PushbackInputStream` (p. 2218), `decaf::util::zip::InflaterInputStream` (p. 1462), and `decaf::io::BufferedInputStream` (p. 594).

6.236.3.10 virtual long long decaf::io::FilterInputStream::skip (long long *num*)
[virtual]

Skips over and discards *n* bytes of data from this input stream.

The skip method may, for a variety of reasons, end up skipping over some smaller number of bytes, possibly 0. This may result from any of a number of conditions; reaching end of file before *n* bytes have been skipped is only one possibility. The actual number of bytes skipped is returned.

The skip method of **InputStream** (p. 1464) creates a byte array and then repeatedly reads into it until *num* bytes have been read or the end of the stream has been reached. Subclasses are encouraged to provide a more efficient implementation of this method.

Parameters

<i>num</i>	The number of bytes to skip.
------------	------------------------------

Returns

total bytes skipped

Exceptions

<i>IOException</i> (p. 1545)	if an I/O error occurs.
<i>Unsupported-Operation-Exception</i>	if the concrete stream class does not support skipping bytes.

Reimplemented from **decaf::io::InputStream** (p. 1472).

Reimplemented in **decaf::io::PushbackInputStream** (p. 2219), **decaf::util::zip::InflaterInputStream** (p. 1463), **decaf::io::BufferedInputStream** (p. 594), and **decaf::util::zip::CheckedInputStream** (p. 807).

6.236.4 Field Documentation

6.236.4.1 volatile bool decaf::io::FilterInputStream::closed [protected]

6.236.4.2 InputStream* decaf::io::FilterInputStream::inputStream
[protected]

6.236.4.3 bool decaf::io::FilterInputStream::own [protected]

The documentation for this class was generated from the following file:

- src/main/decaf/io/**FilterInputStream.h**

6.237 decaf::io::FilterOutputStream Class Reference

This class is the superclass of all classes that filter output streams.

```
#include <src/main/decaf/io/FilterOutputStream.h>
```

Inheritance diagram for decaf::io::FilterOutputStream:

Public Member Functions

- **FilterOutputStream** (**OutputStream** *outputStream, bool own=false)

Constructor, creates a wrapped output stream.

- virtual ~**FilterOutputStream** ()
- virtual void **flush** ()

Flushes this stream by writing any buffered output to the underlying stream.

Exceptions

IOException (p. 1545)	<i>if an I/O error occurs.</i>
------------------------------	--------------------------------

The default implementation of this method does nothing.

- virtual void **close** ()

*Closes this object and deallocates the appropriate resources.
The object is generally no longer usable after calling close.*

Exceptions

IOException (p. 1545)	<i>if an error occurs while closing.</i>
------------------------------	--

The default implementation of this method does nothing.

- virtual std::string **toString** () const

*Output a String representation of this object.
The default version of this method just prints the Class Name.*

Returns

a string representation of the object.

Protected Member Functions

- virtual void **doWriteByte** (unsigned char value)
- virtual void **doWriteArray** (const unsigned char *buffer, int size)
- virtual void **doWriteArrayBounded** (const unsigned char *buffer, int size, int offset, int length)
- virtual bool **isClosed** () const

Protected Attributes

- **OutputStream** * outputStream
- bool own

- volatile bool **closed**

6.237.1 Detailed Description

This class is the superclass of all classes that filter output streams.

These streams sit on top of an already existing output stream (the underlying output stream) which it uses as its basic sink of data, but possibly transforming the data along the way or providing additional functionality.

The class **FilterOutputStream** (p. 1341) itself simply overrides all methods of **OutputStream** (p. 2066) with versions that pass all requests to the underlying output stream. Subclasses of **FilterOutputStream** (p. 1341) may further override some of these methods as well as provide additional methods and fields.

Due to the lack of garbage collection in C++ a design decision was made to add a boolean parameter to the constructor indicating if the wrapped **InputStream** (p. 1464) is owned by this object. That way creation of the underlying stream can occur in a Java like way. Ex:

DataOutputStream (p. 1109) os = new **DataOutputStream** (p. 1109)(new **OutputStream**() (p. 2068), true)

6.237.2 Constructor & Destructor Documentation

6.237.2.1 decaf::io::FilterOutputStream::FilterOutputStream (**OutputStream** * *outputStream*, bool *own* = false)

Constructor, creates a wrapped output stream.

Parameters

<i>output-Stream</i>	the OutputStream (p. 2066) to wrap
<i>own</i>	If true, this object will control the lifetime of the output stream that it encapsulates.

6.237.2.2 virtual decaf::io::FilterOutputStream::~FilterOutputStream ()
[virtual]

6.237.3 Member Function Documentation

6.237.3.1 virtual void decaf::io::FilterOutputStream::close () [virtual]

Closes this object and deallocates the appropriate resources.

The object is generally no longer usable after calling close.

Exceptions

<i>IOException</i> (p. 1545)	if an error occurs while closing.
--	-----------------------------------

The default implementation of this method does nothing.

The close method of **FilterOutputStream** (p. 1341) calls its flush method, and then calls the close method of its underlying output stream.

Reimplemented from **decaf::io::OutputStream** (p. 2068).

Reimplemented in **decaf::util::zip::DeflaterOutputStream** (p. 1192).

6.237.3.2 **virtual void decaf::io::FilterOutputStream::doWriteArray (const unsigned char * *buffer*, int *size*)** [protected, virtual]

Reimplemented from **decaf::io::OutputStream** (p. 2068).

Reimplemented in **decaf::io::BufferedOutputStream** (p. 596).

6.237.3.3 **virtual void decaf::io::FilterOutputStream::doWriteArrayBounded (const unsigned char * *buffer*, int *size*, int *offset*, int *length*)** [protected, virtual]

Reimplemented from **decaf::io::OutputStream** (p. 2068).

Reimplemented in **decaf::util::zip::DeflaterOutputStream** (p. 1193), **decaf::io::DataOutputStream** (p. 1110), **decaf::io::BufferedOutputStream** (p. 596), **decaf::util::zip::CheckedOutputStream** (p. 809), and **activemq::io::LoggingOutputStream** (p. 1708).

6.237.3.4 **virtual void decaf::io::FilterOutputStream::doWriteByte (unsigned char *value*)** [protected, virtual]

Implements **decaf::io::OutputStream** (p. 2069).

Reimplemented in **decaf::util::zip::DeflaterOutputStream** (p. 1193), **decaf::io::DataOutputStream** (p. 1111), **decaf::io::BufferedOutputStream** (p. 597), **decaf::util::zip::CheckedOutputStream** (p. 810), and **activemq::io::LoggingOutputStream** (p. 1708).

6.237.3.5 **virtual void decaf::io::FilterOutputStream::flush ()** [virtual]

Flushes this stream by writing any buffered output to the underlying stream.

Exceptions

<i>IOException</i> (p. 1545)	if an I/O error occurs.
--	-------------------------

The default implementation of this method does nothing.

The flush method of **FilterOutputStream** (p. 1341) calls the flush method of its underlying output stream.

Reimplemented from **decaf::io::OutputStream** (p. 2069).

Reimplemented in **decaf::io::BufferedOutputStream** (p. 597).

6.237.3.6 `virtual bool decaf::io::FilterOutputStream::isClosed () const`
`[protected, virtual]`

Returns

true if this stream has been closed.

6.237.3.7 `virtual std::string decaf::io::FilterOutputStream::toString () const`
`[virtual]`

Output a String representation of this object.

The default version of this method just prints the Class Name.

Returns

a string representation of the object.

The toString method of **FilterOutputStream** (p. 1341) calls the toString method of its underlying output stream.

Reimplemented from **decaf::io::OutputStream** (p. 2070).

6.237.4 Field Documentation

6.237.4.1 `volatile bool decaf::io::FilterOutputStream::closed` `[protected]`

6.237.4.2 `OutputStream* decaf::io::FilterOutputStream::outputStream`
`[protected]`

6.237.4.3 `bool decaf::io::FilterOutputStream::own` `[protected]`

The documentation for this class was generated from the following file:

- `src/main/decaf/io/FilterOutputStream.h`

6.238 decaf::lang::Float Class Reference

```
#include <src/main/decaf/lang/Float.h>
```

Inheritance diagram for decaf::lang::Float:

Public Member Functions

- **Float** (float value)
- **Float** (double value)
- **Float** (const std::string &value)
- virtual ~**Float** ()
- virtual int **compareTo** (const **Float** &f) const
*Compares this **Float** (p. 1344) instance with another.*
- bool **equals** (const **Float** &f) const
- virtual bool **operator==** (const **Float** &f) const
Compares equality between this object and the one passed.
- virtual bool **operator<** (const **Float** &f) const
Compares this object to another and returns true if this object is considered to be less than the one passed.
- virtual int **compareTo** (const float &f) const
*Compares this **Float** (p. 1344) instance with another.*
- bool **equals** (const float &f) const
- virtual bool **operator==** (const float &f) const
Compares equality between this object and the one passed.
- virtual bool **operator<** (const float &f) const
Compares this object to another and returns true if this object is considered to be less than the one passed.
- std::string **toString** () const
- virtual double **doubleValue** () const
Answers the double value which the receiver represents.
- virtual float **floatValue** () const
Answers the float value which the receiver represents.
- virtual unsigned char **byteValue** () const
Answers the byte value which the receiver represents.
- virtual short **shortValue** () const
Answers the short value which the receiver represents.
- virtual int **intValue** () const
Answers the int value which the receiver represents.
- virtual long long **longValue** () const
Answers the long value which the receiver represents.
- bool **isInfinite** () const
- bool **isNaN** () const

Static Public Member Functions

- static int **compare** (float f1, float f2)
Compares the two specified double values.
- static int **floatToIntBits** (float value)
Returns a representation of the specified floating-point value according to the IEEE 754 floating-point "single format" bit layout.
- static int **floatToRawIntBits** (float value)
Returns a representation of the specified floating-point value according to the IEEE 754 floating-point "single format" bit layout, preserving Not-a-Number (NaN) values.
- static float **intBitsToFloat** (int bits)
Returns the float value corresponding to a given bit representation.
- static bool **isInfinite** (float value)
- static bool **isNaN** (float value)
- static float **parseFloat** (const std::string &value)
*Returns a new float initialized to the value represented by the specified string, as performed by the `valueOf` method of class **Float** (p. 1344).*
- static std::string **toHexString** (float value)
Returns a hexadecimal string representation of the float argument.
- static std::string **toString** (float value)
Returns a string representation of the float argument.
- static **Float valueOf** (float value)
*Returns a **Float** (p. 1344) instance representing the specified float value.*
- static **Float valueOf** (const std::string &value)
*Returns a **Float** (p. 1344) instance that wraps a primitive float which is parsed from the string value passed.*

Static Public Attributes

- static const int **SIZE** = 32
The size in bits of the primitive int type.
- static const float **MAX_VALUE**
The maximum value that the primitive type can hold.
- static const float **MIN_VALUE**
The minimum value that the primitive type can hold.
- static const float **NaN**
*Constant for the Not a **Number** (p. 1992) Value.*
- static const float **POSITIVE_INFINITY**
Constant for Positive Infinity.
- static const float **NEGATIVE_INFINITY**
Constant for Negative Infinity.

6.238.1 Constructor & Destructor Documentation

6.238.1.1 `decaf::lang::Float::Float (float value)`

Parameters

<i>value</i>	- the primitive type to wrap
--------------	------------------------------

6.238.1.2 `decaf::lang::Float::Float (double value)`

Parameters

<i>value</i>	- the primitive type to wrap
--------------	------------------------------

6.238.1.3 `decaf::lang::Float::Float (const std::string & value)`

Parameters

<i>value</i>	- the string to convert to a primitive type to wrap
--------------	---

6.238.1.4 `virtual decaf::lang::Float::~~Float ()` `[inline, virtual]`

6.238.2 Member Function Documentation

6.238.2.1 `virtual unsigned char decaf::lang::Float::byteValue () const` `[inline, virtual]`

Answers the byte value which the receiver represents.

Returns

byte the value of the receiver.

Reimplemented from **decaf::lang::Number** (p. 1993).

6.238.2.2 `static int decaf::lang::Float::compare (float f1, float f2)` `[static]`

Compares the two specified double values.

The sign of the integer value returned is the same as that of the integer that would be returned by the call: `Float(f1).compareTo(Float(f2))`

Parameters

<i>f1</i>	- the first double to compare
<i>f2</i>	- the second double to compare

Returns

the value 0 if d1 is numerically equal to f2; a value less than 0 if f1 is numerically less than f2; and a value greater than 0 if f1 is numerically greater than f2.

6.238.2.3 `virtual int decaf::lang::Float::compareTo (const Float & f) const`
[virtual]

Compares this **Float** (p. 1344) instance with another.

Parameters

<code>f</code>	- the Float (p. 1344) instance to be compared
----------------	--

Returns

zero if this object represents the same integer value as the argument; a positive value if this object represents a value greater than the passed in value, and -1 if this object represents a value less than the passed in value.

Implements **decaf::lang::Comparable**< **Float** > (p. 886).

6.238.2.4 `virtual int decaf::lang::Float::compareTo (const float & f) const`
[virtual]

Compares this **Float** (p. 1344) instance with another.

Parameters

<code>f</code>	- the Float (p. 1344) instance to be compared
----------------	--

Returns

zero if this object represents the same integer value as the argument; a positive value if this object represents a value greater than the passed in value, and -1 if this object represents a value less than the passed in value.

Implements **decaf::lang::Comparable**< **float** > (p. 886).

6.238.2.5 `virtual double decaf::lang::Float::doubleValue () const` [inline,
virtual]

Answers the double value which the receiver represents.

Returns

double the value of the receiver.

Implements **decaf::lang::Number** (p. 1993).

6.238.2.6 `bool decaf::lang::Float::equals (const Float & f) const` `[inline, virtual]`

Parameters

<i>f</i> - the Float (p. 1344) object to compare against.
--

Returns

true if the two **Float** (p. 1344) Objects have the same value.

Implements **decaf::lang::Comparable**< **Float** > (p. 887).

6.238.2.7 `bool decaf::lang::Float::equals (const float & f) const` `[inline, virtual]`

Parameters

<i>f</i> - the Float (p. 1344) object to compare against.
--

Returns

true if the two **Float** (p. 1344) Objects have the same value.

Implements **decaf::lang::Comparable**< **float** > (p. 887).

6.238.2.8 `static int decaf::lang::Float::floatToIntBits (float value)` `[static]`

Returns a representation of the specified floating-point value according to the IEEE 754 floating-point "single format" bit layout.

Bit 31 (the bit that is selected by the mask 0x80000000) represents the sign of the floating-point number. Bits 30-23 (the bits that are selected by the mask 0x7f800000) represent the exponent. Bits 22-0 (the bits that are selected by the mask 0x007fffff) represent the significand (sometimes called the mantissa) of the floating-point number.

If the argument is positive infinity, the result is 0x7f800000. If the argument is negative infinity, the result is 0xff800000. If the argument is NaN, the result is 0x7fc00000.

In all cases, the result is an integer that, when given to the **intBitsToFloat(int)** (p. 1350) method, will produce a floating-point value the same as the argument to **floatToIntBits** (except all NaN values are collapsed to a single "canonical" NaN value).

Parameters

<i>value</i> - the float to convert to int bits

Returns

the int that holds the float's value

6.238.2.9 static int decaf::lang::Float::floatToRawIntBits (float *value*) [static]

Returns a representation of the specified floating-point value according to the IEEE 754 floating-point "single format" bit layout, preserving Not-a-Number (NaN) values.

Bit 31 (the bit that is selected by the mask 0x80000000) represents the sign of the floating-point number. Bits 30-23 (the bits that are selected by the mask 0x7f800000) represent the exponent. Bits 22-0 (the bits that are selected by the mask 0x007fffff) represent the significand (sometimes called the mantissa) of the floating-point number.

If the argument is positive infinity, the result is 0x7f800000. If the argument is negative infinity, the result is 0xff800000. If the argument is NaN, the result is the integer representing the actual NaN value. Unlike the floatToIntBits method, intToRawIntBits does not collapse all the bit patterns encoding a NaN to a single "canonical" NaN value.

In all cases, the result is an integer that, when given to the **intBitsToFloat(int)** (p. 1350) method, will produce a floating-point value the same as the argument to floatToRawIntBits.

Parameters

<i>value</i>	The float to convert to a raw int.
--------------	------------------------------------

Returns

the raw int value of the float

6.238.2.10 virtual float decaf::lang::Float::floatValue () const [inline, virtual]

Answers the float value which the receiver represents.

Returns

float the value of the receiver.

Implements **decaf::lang::Number** (p. 1993).

6.238.2.11 static float decaf::lang::Float::intBitsToFloat (int *bits*) [static]

Returns the float value corresponding to a given bit representation.

The argument is considered to be a representation of a floating-point value according to the IEEE 754 floating-point "single format" bit layout.

If the argument is 0x7f800000, the result is positive infinity. If the argument is 0xff800000, the result is negative infinity. If the argument is any value in the range 0x7f800001 through 0x7fffffff or in the range 0xff800001 through 0xffffffff, the result is a NaN. No IEEE 754 floating-point operation provided by C++ can distinguish between two NaN values of the same type with different bit patterns. Distinct values of NaN are only distinguishable by use of the **Float::floatToRawIntBits** (p. 1350) method.

Parameters

<i>bits</i>	- the bits of the float encoded as a float
-------------	--

Returns

a new float created from the int bits.

6.238.2.12 `virtual int decaf::lang::Float::intValue () const` `[inline, virtual]`

Answers the int value which the receiver represents.

Returns

int the value of the receiver.

Implements **decaf::lang::Number** (p. 1993).

6.238.2.13 `bool decaf::lang::Float::isInfinite () const`

Returns

true if the float is equal to positive infinity.

6.238.2.14 `static bool decaf::lang::Float::isInfinite (float value)` `[static]`

Parameters

<i>value</i>	- The float to check.
--------------	-----------------------

Returns

true if the float is equal to infinity.

6.238.2.15 `bool decaf::lang::Float::isNaN () const`

Returns

true if the float is equal to NaN.

6.238.2.16 `static bool decaf::lang::Float::isNaN (float value)` `[static]`

Parameters

<i>value</i>	- The float to check.
--------------	-----------------------

Returns

true if the float is equal to NaN.

6.238.2.17 `virtual long long decaf::lang::Float::longValue () const [inline, virtual]`

Answers the long value which the receiver represents.

Returns

long the value of the receiver.

Implements **decaf::lang::Number** (p. 1994).

6.238.2.18 `virtual bool decaf::lang::Float::operator< (const Float & f) const [inline, virtual]`

Compares this object to another and returns true if this object is considered to be less than the one passed.

This

Parameters

<i>f</i>	- the value to be compared to this one.
----------	---

Returns

true if this object is equal to the one passed.

Implements **decaf::lang::Comparable< Float >** (p. 887).

6.238.2.19 `virtual bool decaf::lang::Float::operator< (const float & f) const [inline, virtual]`

Compares this object to another and returns true if this object is considered to be less than the one passed.

This

Parameters

<i>f</i>	- the value to be compared to this one.
----------	---

Returns

true if this object is equal to the one passed.

Implements **decaf::lang::Comparable**< **float** > (p. 887).

6.238.2.20 `virtual bool decaf::lang::Float::operator==(const Float & f) const [inline, virtual]`

Compares equality between this object and the one passed.

Parameters

<i>f</i>	- the value to be compared to this one.
----------	---

Returns

true if this object is equal to the one passed.

Implements **decaf::lang::Comparable**< **Float** > (p. 888).

6.238.2.21 `virtual bool decaf::lang::Float::operator==(const float & f) const [inline, virtual]`

Compares equality between this object and the one passed.

Parameters

<i>f</i>	- the value to be compared to this one.
----------	---

Returns

true if this object is equal to the one passed.

Implements **decaf::lang::Comparable**< **float** > (p. 888).

6.238.2.22 `static float decaf::lang::Float::parseFloat (const std::string & value) [static]`

Returns a new float initialized to the value represented by the specified string, as performed by the `valueOf` method of class **Float** (p. 1344).

Parameters

<i>value</i>	- the string to parse
--------------	-----------------------

Returns

a float parsed from the string

Exceptions

<i>NumberFormatException</i>	
------------------------------	--

6.238.2.23 `virtual short decaf::lang::Float::shortValue () const` `[inline, virtual]`

Answers the short value which the receiver represents.

Returns

short the value of the receiver.

Reimplemented from **decaf::lang::Number** (p. 1994).

6.238.2.24 `static std::string decaf::lang::Float::toHexString (float value)` `[static]`

Returns a hexadecimal string representation of the float argument.

All characters mentioned below are ASCII characters.

* If the argument is NaN, the result is the string "NaN". * Otherwise, the result is a string that represents the sign and magnitude (absolute value) of the argument. If the sign is negative, the first character of the result is '-'; if the sign is positive, no sign character appears in the result. As for the magnitude m: o If m is infinity, it is represented by the string "Infinity"; thus, positive infinity produces the result "Infinity" and negative infinity produces the result "-Infinity". o If m is zero, it is represented by the string "0x0.0p0"; thus, negative zero produces the result "-0x0.0p0" and positive zero produces the result "0x0.0p0". o If m is a float value with a normalized representation, substrings are used to represent the significand and exponent fields. The significand is represented by the characters "0x1." followed by a lowercase hexadecimal representation of the rest of the significand as a fraction. Trailing zeros in the hexadecimal representation are removed unless all the digits are zero, in which case a single zero is used. Next, the exponent is represented by "p" followed by a decimal string of the unbiased exponent as if produced by a call to **Integer.toString** (p. 1513) on the exponent value. o If m is a float value with a subnormal representation, the significand is represented by the characters "0x0." followed by a hexadecimal representation of the rest of the significand as a fraction. Trailing zeros in the hexadecimal representation are removed. Next, the exponent is represented by "p-126". Note that there must be at least one nonzero digit in a subnormal significand.

Parameters

<i>value</i>	- The float to convert to a string
--------------	------------------------------------

Returns

the Hex formatted float string.

6.238.2.25 `std::string decaf::lang::Float::toString () const`**Returns**

this **Float** (p. 1344) Object as a **String** (p. 2620) Representation

6.238.2.26 `static std::string decaf::lang::Float::toString (float value) [static]`

Returns a string representation of the float argument.

All characters mentioned below are ASCII characters.

If the argument is NaN, the result is the string "NaN". Otherwise, the result is a string that represents the sign and magnitude (absolute value) of the argument. If the sign is negative, the first character of the result is '-'; if the sign is positive, no sign character appears in the result. As for the magnitude m: o If m is infinity, it is represented by the characters "Infinity"; thus, positive infinity produces the result "Infinity" and negative infinity produces the result "-Infinity". o If m is zero, it is represented by the characters "0.0"; thus, negative zero produces the result "-0.0" and positive zero produces the result "0.0". o If m is greater than or equal to 10⁻³ but less than 10⁷, then it is represented as the integer part of m, in decimal form with no leading zeroes, followed by '.', followed by one or more decimal digits representing the fractional part of m. o If m is less than 10⁻³ or greater than or equal to 10⁷, then it is represented in so-called "computerized scientific notation." Let n be the unique integer such that 10ⁿ ≤ m < 10ⁿ⁺¹; then let a be the mathematically exact quotient of m and 10ⁿ so that 1 ≤ a < 10. The magnitude is then represented as the integer part of a, as a single decimal digit, followed by '.', followed by decimal digits representing the fractional part of a, followed by the letter 'E', followed by a representation of n as a decimal integer, as produced by the method **Integer.toString(int)** (p. 1514).

Parameters

<i>value</i>	- The float to convert to a string
--------------	------------------------------------

Returns

the formatted float string.

6.238.2.27 `static Float decaf::lang::Float::valueOf (float value) [static]`

Returns a **Float** (p. 1344) instance representing the specified float value.

Parameters

<i>value</i>	- float to wrap
--------------	-----------------

Returns

new **Float** (p. 1344) instance wrapping the primitive value

6.238.2.28 static **Float** decaf::lang::Float::valueOf (const std::string & *value*)
[static]

Returns a **Float** (p. 1344) instance that wraps a primitive float which is parsed from the string value passed.

Parameters

<i>value</i>	- the string to parse
--------------	-----------------------

Returns

a new **Float** (p. 1344) instance wrapping the float parsed from value

Exceptions

<i>NumberFormatException</i>	on error.
------------------------------	-----------

6.238.3 Field Documentation

6.238.3.1 const float decaf::lang::Float::MAX_VALUE [static]

The maximum value that the primitive type can hold.

6.238.3.2 const float decaf::lang::Float::MIN_VALUE [static]

The minimum value that the primitive type can hold.

6.238.3.3 const float decaf::lang::Float::NaN [static]

Constant for the Not a **Number** (p. 1992) Value.

6.238.3.4 const float decaf::lang::Float::NEGATIVE_INFINITY [static]

Constant for Negative Infinity.

6.238.3.5 `const float decaf::lang::Float::POSITIVE_INFINITY` [static]

Constant for Positive Infinity.

6.238.3.6 `const int decaf::lang::Float::SIZE = 32` [static]

The size in bits of the primitive int type.

The documentation for this class was generated from the following file:

- `src/main/decaf/lang/Float.h`

6.239 `decaf::internal::nio::FloatArrayBuffer` Class Reference

```
#include <src/main/decaf/internal/nio/FloatArrayBuffer.h>
```

Inheritance diagram for `decaf::internal::nio::FloatArrayBuffer`:

Public Member Functions

- **FloatArrayBuffer** (int size, bool readOnly=false)
Creates a **FloatArrayBuffer** (p. 1357) object that has its backing array allocated internally and is then owned and deleted when this object is deleted.
- **FloatArrayBuffer** (float *array, int size, int offset, int length, bool readOnly=false)
Creates a **FloatArrayBuffer** (p. 1357) object that wraps the given array.
- **FloatArrayBuffer** (const `decaf::lang::Pointer< ByteArrayAdapter >` &array, int offset, int capacity, bool readOnly=false)
Creates a byte buffer that wraps the passed `ByteArrayAdapter` and start at the given offset.
- **FloatArrayBuffer** (const **FloatArrayBuffer** &other)
Create a **FloatArrayBuffer** (p. 1357) that mirrors this one, meaning it shares a reference to this buffers `ByteArrayAdapter` and when changes are made to that data it is reflected in both.
- virtual `~FloatArrayBuffer` ()
- virtual float * array ()
Returns the float array that backs this buffer (optional operation).
Modifications to this buffer's content will cause the returned array's content to be modified, and vice versa.
Invoke the `hasArray` method before invoking this method in order to ensure that this buffer has an accessible backing array.
Returns
the array that backs this **Buffer** (p. 582).

Exceptions

ReadOnlyBuffer-Exception (p. 2244)	if this Buffer (p. 582) is read only.
UnsupportedOperation-Exception	if the underlying store has no array.

- virtual int **arrayOffset** ()

Returns the offset within this buffer's backing array of the first element of the buffer (optional operation).

Invoke the `hasArray` method before invoking this method in order to ensure that this buffer has an accessible backing array.

Returns

The offset into the backing array where index zero starts.

Exceptions

ReadOnlyBuffer-Exception (p. 2244)	if this Buffer (p. 582) is read only.
UnsupportedOperation-Exception	if the underlying store has no array.

- virtual **FloatBuffer** * **asReadOnlyBuffer** () const

Creates a new, read-only float buffer that shares this buffer's content.

The content of the new buffer will be that of this buffer. Changes to this buffer's content will be visible in the new buffer; the new buffer itself, however, will be read-only and will not allow the shared content to be modified. The two buffers' position, limit, and mark values will be independent.

If this buffer is itself read-only then this method behaves in exactly the same way as the `duplicate` method.

The new buffer's capacity, limit, position, and mark values will be identical to those of this buffer.

Returns

The new, read-only float buffer which the caller then owns.

- virtual **FloatBuffer** & **compact** ()

Compacts this buffer.

The bytes between the buffer's current position and its limit, if any, are copied to the beginning of the buffer. That is, the byte at index $p = \text{position}()$ (p. 587) is copied to index zero, the byte at index $p + 1$ is copied to index one, and so forth until the byte at index $\text{limit}()$ (p. 586) - 1 is copied to index $n = \text{limit}()$ (p. 586) - 1 - p . The buffer's position is then set to $n+1$ and its limit is set to its capacity. The mark, if defined, is discarded.

The buffer's position is set to the number of bytes copied, rather than to zero, so that an invocation of this method can be followed immediately by an invocation of another relative put method.

Returns

a reference to this **FloatBuffer** (p. 1368).

Exceptions

ReadOnlyBuffer-Exception (p. 2244)	if this buffer is read-only
--	-----------------------------

- virtual **FloatBuffer * duplicate ()**

Creates a new float buffer that shares this buffer's content.

The content of the new buffer will be that of this buffer. Changes to this buffer's content will be visible in the new buffer, and vice versa; the two buffers' position, limit, and mark values will be independent.

The new buffer's capacity, limit, position, and mark values will be identical to those of this buffer. The new buffer will be read-only if, and only if, this buffer is read-only.

Returns

*a new float **Buffer** (p. 582) which the caller owns.*

- virtual float **get ()**

Relative get method.

Reads the value at this buffer's current position, and then increments the position.

Returns

the float at the current position.

Exceptions

BufferUnderflow-Exception (p. 611)	<i>if there no more data to return.</i>
--	---

- virtual float **get (int index) const**

Absolute get method.

Reads the value at the given index.

Parameters

index	<i>The index in the Buffer (p. 582) where the float is to be read</i>
-------	--

Returns

the float that is located at the given index

Exceptions

IndexOutOfBoundsException Exception	<i>if index is not smaller than the buffer's limit</i>
---	--

- virtual bool **hasArray () const**

Tells whether or not this buffer is backed by an accessible float array.

If this method returns true then the array and arrayOffset methods may safely be invoked. Subclasses should override this method if they do not have a backing array as this class always returns true.

Returns

true if, and only if, this buffer is backed by an array and is not read-only

- virtual bool **isReadOnly () const**

Tells whether or not this buffer is read-only.

Returns

true if, and only if, this buffer is read-only.

- virtual **FloatBuffer & put (float value)**

Writes the given floats into this buffer at the current position, and then increments the position.

Parameters

value	<i>The floats value to be written.</i>
-------	--

Returns

a reference to this buffer.

Exceptions

BufferOverflow-Exception (p. 609)	<i>if this buffer's current position is not smaller than its limit</i>
ReadOnlyBuffer-Exception (p. 2244)	<i>if this buffer is read-only</i>

- virtual **FloatBuffer** & **put** (int index, float value)

Writes the given floats into this buffer at the given index.

Parameters

index	<i>The position in the Buffer (p. 582) to write the data.</i>
value	<i>The floats to write.</i>

Returns

a reference to this buffer.

Exceptions

IndexOutOfBoundsException	<i>if index greater than the buffer's limit minus the size of the type being written, or index is negative.</i>
ReadOnlyBuffer-Exception (p. 2244)	<i>if this buffer is read-only.</i>

- virtual **FloatBuffer** * **slice** () const

*Creates a new **FloatBuffer** (p. 1368) whose content is a shared subsequence of this buffer's content.*

The content of the new buffer will start at this buffer's current position. Changes to this buffer's content will be visible in the new buffer, and vice versa; the two buffers' position, limit, and mark values will be independent.

The new buffer's position will be zero, its capacity and its limit will be the number of bytes remaining in this buffer, and its mark will be undefined. The new buffer will be read-only if, and only if, this buffer is read-only.

Returns

*the newly create **FloatBuffer** (p. 1368) which the caller owns.*

Protected Member Functions

- virtual void **setReadOnly** (bool value)

*Sets this **FloatArrayBuffer** (p. 1357) as Read-Only.*

6.239.1 Constructor & Destructor Documentation

6.239.1.1 `decaf::internal::nio::FloatArrayBuffer::FloatArrayBuffer (int size, bool readOnly = false)`

Creates a **FloatArrayBuffer** (p. 1357) object that has its backing array allocated internally and is then owned and deleted when this object is deleted.

The array is initially created with all elements initialized to zero.

Parameters

<i>size</i>	The size of the array, this is the limit we read and write to.
<i>readOnly</i>	Boolean indicating if this buffer should be read-only, default as false.

Exceptions

<i>IllegalArgumentException</i>	if the capacity value is negative.
---------------------------------	------------------------------------

6.239.1.2 `decaf::internal::nio::FloatArrayBuffer::FloatArrayBuffer (float * array, int size, int offset, int length, bool readOnly = false)`

Creates a **FloatArrayBuffer** (p. 1357) object that wraps the given array.

If the own flag is set then it will delete this array when this object is deleted.

Parameters

<i>array</i>	The actual array to wrap.
<i>size</i>	The size of the given array.
<i>offset</i>	The position that is this buffers start position.
<i>length</i>	The limit of how many bytes into the array this Buffer can write.
<i>readOnly</i>	Boolean indicating if this buffer should be read-only, default as false.

Exceptions

<i>NullPointerException</i>	if buffer is NULL
<i>IndexOutOfBoundsException</i>	if offset is greater than array capacity.

6.239.1.3 `decaf::internal::nio::FloatArrayBuffer::FloatArrayBuffer (const decaf::lang::Pointer< ByteArrayAdapter > & array, int offset, int capacity, bool readOnly = false)`

Creates a byte buffer that wraps the passed ByteArrayAdapter and start at the given offset.

The capacity and limit of the new **FloatArrayBuffer** (p. 1357) will be that of the remaining capacity of the passed buffer.

Parameters

<i>array</i>	The ByteAdapter to wrap.
<i>offset</i>	The position that is this buffers start position.
<i>length</i>	The limit of how many bytes into the array this Buffer can write.
<i>readOnly</i>	Boolean indicating if this buffer should be read-only, default as false.

Exceptions

<i>NullPointerException</i>	if array is NULL
<i>IndexOutOfBoundsException</i>	if offset + length is greater than array size.

6.239.1.4 decaf::internal::nio::FloatArrayBuffer::FloatArrayBuffer (const FloatArrayBuffer & other)

Create a **FloatArrayBuffer** (p. 1357) that mirrors this one, meaning it shares a reference to this buffers ByteAdapter and when changes are made to that data it is reflected in both.

Parameters

<i>other</i>	The FloatArrayBuffer (p. 1357) this one is to mirror.
--------------	--

6.239.1.5 virtual decaf::internal::nio::FloatArrayBuffer::~FloatArrayBuffer () [virtual]

6.239.2 Member Function Documentation

6.239.2.1 virtual float* decaf::internal::nio::FloatArrayBuffer::array () [virtual]

Returns the float array that backs this buffer (optional operation).

Modifications to this buffer's content will cause the returned array's content to be modified, and vice versa.

Invoke the hasArray method before invoking this method in order to ensure that this buffer has an accessible backing array.

Returns

the array that backs this **Buffer** (p. 582).

Exceptions

<i>ReadOnlyBufferException</i> (p. 2244)	if this Buffer (p. 582) is read only.
---	--

<i>Unsupported- OperationException</i>	if the underlying store has no array.
--	---------------------------------------

Implements **decaf::nio::FloatBuffer** (p. 1370).

6.239.2.2 `virtual int decaf::internal::nio::FloatArrayBuffer::arrayOffset ()`
[virtual]

Returns the offset within this buffer's backing array of the first element of the buffer (optional operation).

Invoke the `hasArray` method before invoking this method in order to ensure that this buffer has an accessible backing array.

Returns

The offset into the backing array where index zero starts.

Exceptions

<i>ReadOnlyBuffer- Exception</i> (p. 2244)	if this Buffer (p. 582) is read only.
<i>Unsupported- OperationException</i>	if the underlying store has no array.

Implements **decaf::nio::FloatBuffer** (p. 1371).

6.239.2.3 `virtual FloatBuffer* decaf::internal::nio::FloatArrayBuffer::asReadOnly-
Buffer () const` [virtual]

Creates a new, read-only float buffer that shares this buffer's content.

The content of the new buffer will be that of this buffer. Changes to this buffer's content will be visible in the new buffer; the new buffer itself, however, will be read-only and will not allow the shared content to be modified. The two buffers' position, limit, and mark values will be independent.

If this buffer is itself read-only then this method behaves in exactly the same way as the `duplicate` method.

The new buffer's capacity, limit, position, and mark values will be identical to those of this buffer.

Returns

The new, read-only float buffer which the caller then owns.

Implements **decaf::nio::FloatBuffer** (p. 1371).

6.239.2.4 virtual FloatBuffer& decaf::internal::nio::FloatArrayBuffer::compact () [virtual]

Compacts this buffer.

The bytes between the buffer's current position and its limit, if any, are copied to the beginning of the buffer. That is, the byte at index $p = \text{position}()$ (p. 587) is copied to index zero, the byte at index $p + 1$ is copied to index one, and so forth until the byte at index $\text{limit}()$ (p. 586) - 1 is copied to index $n = \text{limit}()$ (p. 586) - 1 - p . The buffer's position is then set to $n+1$ and its limit is set to its capacity. The mark, if defined, is discarded.

The buffer's position is set to the number of bytes copied, rather than to zero, so that an invocation of this method can be followed immediately by an invocation of another relative put method.

Returns

a reference to this **FloatBuffer** (p. 1368).

Exceptions

ReadOnlyBufferException (p. 2244)	if this buffer is read-only
---	-----------------------------

Implements **decaf::nio::FloatBuffer** (p. 1372).

6.239.2.5 virtual FloatBuffer* decaf::internal::nio::FloatArrayBuffer::duplicate () [virtual]

Creates a new float buffer that shares this buffer's content.

The content of the new buffer will be that of this buffer. Changes to this buffer's content will be visible in the new buffer, and vice versa; the two buffers' position, limit, and mark values will be independent.

The new buffer's capacity, limit, position, and mark values will be identical to those of this buffer. The new buffer will be read-only if, and only if, this buffer is read-only.

Returns

a new float **Buffer** (p. 582) which the caller owns.

Implements **decaf::nio::FloatBuffer** (p. 1372).

6.239.2.6 virtual float decaf::internal::nio::FloatArrayBuffer::get () [virtual]

Relative get method.

Reads the value at this buffer's current position, and then increments the position.

Returns

the float at the current position.

Exceptions

<i>BufferUnderflow-Exception</i> (p. 611)	if there no more data to return.
---	----------------------------------

Implements **decaf::nio::FloatBuffer** (p. 1373).

6.239.2.7 virtual float **decaf::internal::nio::FloatArrayBuffer::get** (int *index*) const
[virtual]

Absolute get method.

Reads the value at the given index.

Parameters

<i>index</i>	The index in the Buffer (p. 582) where the float is to be read
--------------	---

Returns

the float that is located at the given index

Exceptions

<i>IndexOutOfBoundsException</i>	if index is not smaller than the buffer's limit
---	---

Implements **decaf::nio::FloatBuffer** (p. 1373).

6.239.2.8 virtual bool **decaf::internal::nio::FloatArrayBuffer::hasArray** () const
[inline, virtual]

Tells whether or not this buffer is backed by an accessible float array.

If this method returns true then the array and arrayOffset methods may safely be invoked. Subclasses should override this method if they do not have a backing array as this class always returns true.

Returns

true if, and only if, this buffer is backed by an array and is not read-only

Implements **decaf::nio::FloatBuffer** (p. 1375).

6.239.2.9 virtual bool **decaf::internal::nio::FloatArrayBuffer::isReadOnly** () const
[inline, virtual]

Tells whether or not this buffer is read-only.

Returns

true if, and only if, this buffer is read-only.

Implements **decaf::nio::Buffer** (p. 586).

6.239.2.10 virtual **FloatBuffer& decaf::internal::nio::FloatArrayBuffer::put** (float
value) [virtual]

Writes the given floats into this buffer at the current position, and then increments the position.

Parameters

<i>value</i>	The floats value to be written.
--------------	---------------------------------

Returns

a reference to this buffer.

Exceptions

BufferOverflow-Exception (p. 609)	if this buffer's current position is not smaller than its limit
ReadOnlyBuffer-Exception (p. 2244)	if this buffer is read-only

Implements **decaf::nio::FloatBuffer** (p. 1377).

6.239.2.11 virtual **FloatBuffer& decaf::internal::nio::FloatArrayBuffer::put** (int *index*,
float *value*) [virtual]

Writes the given floats into this buffer at the given index.

Parameters

<i>index</i>	The position in the Buffer (p. 582) to write the data.
<i>value</i>	The floats to write.

Returns

a reference to this buffer.

Exceptions

<i>IndexOutOfBoundsException</i>	if index greater than the buffer's limit minus the size of the type being written, or index is negative.
ReadOnlyBufferException (p. 2244)	if this buffer is read-only.

Implements **decaf::nio::FloatBuffer** (p. 1378).

6.239.2.12 virtual void **decaf::internal::nio::FloatArrayBuffer::setReadOnly** (bool *value*) [inline, protected, virtual]

Sets this **FloatArrayBuffer** (p. 1357) as Read-Only.

Parameters

<i>value</i>	Boolean value, true if this buffer is to be read-only, false otherwise.
--------------	---

6.239.2.13 virtual **FloatBuffer*** **decaf::internal::nio::FloatArrayBuffer::slice** () const [virtual]

Creates a new **FloatBuffer** (p. 1368) whose content is a shared subsequence of this buffer's content.

The content of the new buffer will start at this buffer's current position. Changes to this buffer's content will be visible in the new buffer, and vice versa; the two buffers' position, limit, and mark values will be independent.

The new buffer's position will be zero, its capacity and its limit will be the number of bytes remaining in this buffer, and its mark will be undefined. The new buffer will be read-only if, and only if, this buffer is read-only.

Returns

the newly create **FloatBuffer** (p. 1368) which the caller owns.

Implements **decaf::nio::FloatBuffer** (p. 1378).

The documentation for this class was generated from the following file:

- src/main/decaf/internal/nio/**FloatArrayBuffer.h**

6.240 decaf::nio::FloatBuffer Class Reference

This class defines four categories of operations upon float buffers:

```
#include <src/main/decaf/nio/FloatBuffer.h>
```

Inheritance diagram for decaf::nio::FloatBuffer:

Public Member Functions

- virtual **~FloatBuffer** ()
- virtual std::string **toString** () const
- virtual float * **array** ()=0
Returns the float array that backs this buffer (optional operation).
- virtual int **arrayOffset** ()=0
Returns the offset within this buffer's backing array of the first element of the buffer (optional operation).
- virtual **FloatBuffer** * **asReadOnlyBuffer** () const =0
Creates a new, read-only float buffer that shares this buffer's content.
- virtual **FloatBuffer** & **compact** ()=0
Compacts this buffer.
- virtual **FloatBuffer** * **duplicate** ()=0
Creates a new float buffer that shares this buffer's content.
- virtual float **get** ()=0
Relative get method.
- virtual float **get** (int index) const =0
Absolute get method.
- **FloatBuffer** & **get** (std::vector< float > buffer)
Relative bulk get method.
- **FloatBuffer** & **get** (float *buffer, int size, int offset, int length)
Relative bulk get method.
- virtual bool **hasArray** () const =0
Tells whether or not this buffer is backed by an accessible float array.
- **FloatBuffer** & **put** (**FloatBuffer** &src)
This method transfers the floats remaining in the given source buffer into this buffer.
- **FloatBuffer** & **put** (const float *buffer, int size, int offset, int length)
This method transfers floats into this buffer from the given source array.
- **FloatBuffer** & **put** (std::vector< float > &buffer)
This method transfers the entire content of the given source floats array into this buffer.
- virtual **FloatBuffer** & **put** (float value)=0
Writes the given floats into this buffer at the current position, and then increments the position.
- virtual **FloatBuffer** & **put** (int index, float value)=0

Writes the given floats into this buffer at the given index.

- virtual **FloatBuffer** * **slice** () const =0

*Creates a new **FloatBuffer** (p. 1368) whose content is a shared subsequence of this buffer's content.*

- virtual int **compareTo** (const **FloatBuffer** &value) const
- virtual bool **equals** (const **FloatBuffer** &value) const
- virtual bool **operator==** (const **FloatBuffer** &value) const
- virtual bool **operator<** (const **FloatBuffer** &value) const

Static Public Member Functions

- static **FloatBuffer** * **allocate** (int **capacity**)

Allocates a new Double buffer.

- static **FloatBuffer** * **wrap** (float ***array**, int size, int offset, int length)

*Wraps the passed buffer with a new **FloatBuffer** (p. 1368).*

- static **FloatBuffer** * **wrap** (std::vector< float > &buffer)

*Wraps the passed STL float Vector in a **FloatBuffer** (p. 1368).*

Protected Member Functions

- **FloatBuffer** (int **capacity**)

*Creates a **FloatBuffer** (p. 1368) object that has its backing array allocated internally and is then owned and deleted when this object is deleted.*

6.240.1 Detailed Description

This class defines four categories of operations upon float buffers:

o Absolute and relative get and put methods that read and write single floats; o Relative bulk get methods that transfer contiguous sequences of floats from this buffer into an array; and o Relative bulk put methods that transfer contiguous sequences of floats from a float array or some other float buffer into this buffer o Methods for compacting, duplicating, and slicing a float buffer.

Double buffers can be created either by allocation, which allocates space for the buffer's content, by wrapping an existing float array into a buffer, or by creating a view of an existing byte buffer

Methods in this class that do not otherwise have a value to return are specified to return the buffer upon which they are invoked. This allows method invocations to be chained.

6.240.2 Constructor & Destructor Documentation

6.240.2.1 decaf::nio::FloatBuffer::FloatBuffer (int *capacity*) [protected]

Creates a **FloatBuffer** (p. 1368) object that has its backing array allocated internally and is then owned and deleted when this object is deleted.

The array is initially created with all elements initialized to zero.

Parameters

<i>capacity</i>	The size and limit of the Buffer (p. 582) in floats.
-----------------	---

Exceptions

<i>IllegalArgument-Exception</i>	if capacity is negative.
----------------------------------	--------------------------

6.240.2.2 virtual decaf::nio::FloatBuffer::~FloatBuffer () [inline, virtual]

6.240.3 Member Function Documentation

6.240.3.1 static FloatBuffer* decaf::nio::FloatBuffer::allocate (int *capacity*) [static]

Allocates a new Double buffer.

The new buffer's position will be zero, its limit will be its capacity, and its mark will be undefined. It will have a backing array, and its array offset will be zero.

Parameters

<i>capacity</i>	The size of the Double buffer in floats.
-----------------	--

Returns

the **FloatBuffer** (p. 1368) that was allocated, caller owns.

6.240.3.2 virtual float* decaf::nio::FloatBuffer::array () [pure virtual]

Returns the float array that backs this buffer (optional operation).

Modifications to this buffer's content will cause the returned array's content to be modified, and vice versa.

Invoke the `hasArray` method before invoking this method in order to ensure that this buffer has an accessible backing array.

Returns

the array that backs this **Buffer** (p. 582).

Exceptions

<i>ReadOnlyBuffer-Exception</i> (p. 2244)	if this Buffer (p. 582) is read only.
<i>UnsupportedOperationException</i>	if the underlying store has no array.

Implemented in **decaf::internal::nio::FloatArrayBuffer** (p. 1362).

6.240.3.3 `virtual int decaf::nio::FloatBuffer::arrayOffset()` [pure virtual]

Returns the offset within this buffer's backing array of the first element of the buffer (optional operation).

Invoke the `hasArray` method before invoking this method in order to ensure that this buffer has an accessible backing array.

Returns

The offset into the backing array where index zero starts.

Exceptions

<i>ReadOnlyBuffer-Exception</i> (p. 2244)	if this Buffer (p. 582) is read only.
<i>UnsupportedOperationException</i>	if the underlying store has no array.

Implemented in **decaf::internal::nio::FloatArrayBuffer** (p. 1363).

6.240.3.4 `virtual FloatBuffer* decaf::nio::FloatBuffer::asReadOnlyBuffer() const` [pure virtual]

Creates a new, read-only float buffer that shares this buffer's content.

The content of the new buffer will be that of this buffer. Changes to this buffer's content will be visible in the new buffer; the new buffer itself, however, will be read-only and will not allow the shared content to be modified. The two buffers' position, limit, and mark values will be independent.

If this buffer is itself read-only then this method behaves in exactly the same way as the `duplicate` method.

The new buffer's capacity, limit, position, and mark values will be identical to those of

this buffer.

Returns

The new, read-only float buffer which the caller then owns.

Implemented in **decaf::internal::nio::FloatArrayBuffer** (p. 1363).

6.240.3.5 `virtual FloatBuffer& decaf::nio::FloatBuffer::compact () [pure virtual]`

Compacts this buffer.

The bytes between the buffer's current position and its limit, if any, are copied to the beginning of the buffer. That is, the byte at index $p = \text{position}()$ (p. 587) is copied to index zero, the byte at index $p + 1$ is copied to index one, and so forth until the byte at index $\text{limit}()$ (p. 586) - 1 is copied to index $n = \text{limit}()$ (p. 586) - 1 - p . The buffer's position is then set to $n+1$ and its limit is set to its capacity. The mark, if defined, is discarded.

The buffer's position is set to the number of bytes copied, rather than to zero, so that an invocation of this method can be followed immediately by an invocation of another relative put method.

Returns

a reference to this **FloatBuffer** (p. 1368).

Exceptions

<i>ReadOnlyBufferException</i> (p. 2244)	if this buffer is read-only
--	-----------------------------

Implemented in **decaf::internal::nio::FloatArrayBuffer** (p. 1364).

6.240.3.6 `virtual int decaf::nio::FloatBuffer::compareTo (const FloatBuffer & value) const [virtual]`

6.240.3.7 `virtual FloatBuffer* decaf::nio::FloatBuffer::duplicate () [pure virtual]`

Creates a new float buffer that shares this buffer's content.

The content of the new buffer will be that of this buffer. Changes to this buffer's content will be visible in the new buffer, and vice versa; the two buffers' position, limit, and mark values will be independent.

The new buffer's capacity, limit, position, and mark values will be identical to those of this buffer. The new buffer will be read-only if, and only if, this buffer is read-only.

Returns

a new float **Buffer** (p. 582) which the caller owns.

Implemented in **decaf::internal::nio::FloatArrayBuffer** (p. 1364).

6.240.3.8 `virtual bool decaf::nio::FloatBuffer::equals (const FloatBuffer & value)
const [virtual]`

6.240.3.9 `virtual float decaf::nio::FloatBuffer::get () [pure virtual]`

Relative get method.

Reads the value at this buffer's current position, and then increments the position.

Returns

the float at the current position.

Exceptions

<i>BufferUnderflow- Exception</i> (p. 611)	if there no more data to return.
--	----------------------------------

Implemented in **decaf::internal::nio::FloatArrayBuffer** (p. 1364).

6.240.3.10 `virtual float decaf::nio::FloatBuffer::get (int index) const [pure
virtual]`

Absolute get method.

Reads the value at the given index.

Parameters

<i>index</i>	The index in the Buffer (p. 582) where the float is to be read
--------------	---

Returns

the float that is located at the given index

Exceptions

<i>IndexOutOfBounds- Exception</i>	if index is not smaller than the buffer's limit
---	---

Implemented in **decaf::internal::nio::FloatArrayBuffer** (p. 1365).

6.240.3.11 FloatBuffer& decaf::nio::FloatBuffer::get (std::vector< float > *buffer*)

Relative bulk get method.

This method transfers values from this buffer into the given destination vector. An invocation of this method of the form `src.get(a)` behaves in exactly the same way as the invocation. The vector must be sized to the amount of data that is to be read, that is to say, the caller should call `buffer.resize(N)` before calling this get method.

Returns

a reference to this **Buffer** (p. 582).

Exceptions

BufferUnderflow-Exception (p. 611)	if there are fewer than length floats remaining in this buffer
--	--

6.240.3.12 FloatBuffer& decaf::nio::FloatBuffer::get (float * *buffer*, int *size*, int *offset*, int *length*)

Relative bulk get method.

This method transfers floats from this buffer into the given destination array. If there are fewer floats remaining in the buffer than are required to satisfy the request, that is, if `length > remaining()` (p. 588), then no bytes are transferred and a **BufferUnderflow-Exception** (p. 611) is thrown.

Otherwise, this method copies length floats from this buffer into the given array, starting at the current position of this buffer and at the given offset in the array. The position of this buffer is then incremented by length.

Parameters

<i>buffer</i>	The pointer to an allocated buffer to fill.
<i>size</i>	The size of the passed in buffer.
<i>offset</i>	The position in the buffer to start filling.
<i>length</i>	The amount of data to put in the passed buffer.

Returns

a reference to this **Buffer** (p. 582).

Exceptions

BufferUnderflow-Exception (p. 611)	if there are fewer than length floats remaining in this buffer
--	--

<i>NullPointerException</i>	if the passed buffer is null.
<i>IndexOutOfBoundsException</i>	if the preconditions of size, offset, or length are not met.

6.240.3.13 `virtual bool decaf::nio::FloatBuffer::hasArray () const` [pure virtual]

Tells whether or not this buffer is backed by an accessible float array.

If this method returns true then the `array` and `arrayOffset` methods may safely be invoked. Subclasses should override this method if they do not have a backing array as this class always returns true.

Returns

true if, and only if, this buffer is backed by an array and is not read-only

Implemented in `decaf::internal::nio::FloatArrayBuffer` (p. 1365).

6.240.3.14 `virtual bool decaf::nio::FloatBuffer::operator< (const FloatBuffer & value) const` [virtual]

6.240.3.15 `virtual bool decaf::nio::FloatBuffer::operator== (const FloatBuffer & value) const` [virtual]

6.240.3.16 `FloatBuffer& decaf::nio::FloatBuffer::put (FloatBuffer & src)`

This method transfers the floats remaining in the given source buffer into this buffer.

If there are more floats remaining in the source buffer than in this buffer, that is, if `src.remaining() > remaining()` (p. 588), then no floats are transferred and a **BufferOverflowException** (p. 609) is thrown.

Otherwise, this method copies `n = src.remaining()` floats from the given buffer into this buffer, starting at each buffer's current position. The positions of both buffers are then incremented by `n`.

Parameters

<code>src</code>	The buffer to take floats from an place in this one.
------------------	--

Returns

a reference to this buffer.

Exceptions

BufferOverflow-Exception (p. 609)	if there is insufficient space in this buffer for the remaining floats in the source buffer
<i>IllegalArgumentException</i>	if the source buffer is this buffer.
ReadOnlyBuffer-Exception (p. 2244)	if this buffer is read-only.

6.240.3.17 FloatBuffer& decaf::nio::FloatBuffer::put (const float * *buffer*, int *size*, int *offset*, int *length*)

This method transfers floats into this buffer from the given source array.

If there are more floats to be copied from the array than remain in this buffer, that is, if `length > remaining()` (p. 588), then no floats are transferred and a **BufferOverflow-Exception** (p. 609) is thrown.

Otherwise, this method copies `length` bytes from the given array into this buffer, starting at the given `offset` in the array and at the current position of this buffer. The position of this buffer is then incremented by `length`.

Parameters

<i>buffer</i>	The array from which floats are to be read.
<i>size</i>	The size of the passed in buffer.
<i>offset</i>	The offset within the array of the first float to be read.
<i>length</i>	The number of floats to be read from the given array.

Returns

a reference to this buffer.

Exceptions

BufferOverflow-Exception (p. 609)	if there is insufficient space in this buffer
ReadOnlyBuffer-Exception (p. 2244)	if this buffer is read-only
<i>NullPointerException</i>	if the passed buffer is null.
<i>IndexOutOfBoundsException</i>	if the preconditions of <code>size</code> , <code>offset</code> , or <code>length</code> are not met.

6.240.3.18 FloatBuffer& decaf::nio::FloatBuffer::put (std::vector< float > & buffer)

This method transfers the entire content of the given source floats array into this buffer.

This is the same as calling put(&buffer[0], 0, buffer.size()).

Parameters

<i>buffer</i>	The buffer whose contents are copied to this FloatBuffer (p. 1368)
---------------	---

Returns

a reference to this buffer.

Exceptions

BufferOverflow-Exception (p. 609)	if there is insufficient space in this buffer
ReadOnlyBuffer-Exception (p. 2244)	if this buffer is read-only

6.240.3.19 virtual FloatBuffer& decaf::nio::FloatBuffer::put (float value) [pure virtual]

Writes the given floats into this buffer at the current position, and then increments the position.

Parameters

<i>value</i>	The floats value to be written.
--------------	---------------------------------

Returns

a reference to this buffer.

Exceptions

BufferOverflow-Exception (p. 609)	if this buffer's current position is not smaller than its limit
ReadOnlyBuffer-Exception (p. 2244)	if this buffer is read-only

Implemented in **decaf::internal::nio::FloatArrayBuffer** (p. 1366).

6.240.3.20 `virtual FloatBuffer& decaf::nio::FloatBuffer::put (int index, float value)`
`[pure virtual]`

Writes the given floats into this buffer at the given index.

Parameters

<i>index</i>	The position in the Buffer (p. 582) to write the data.
<i>value</i>	The floats to write.

Returns

a reference to this buffer.

Exceptions

<i>IndexOutOfBoundsException</i>	if index greater than the buffer's limit minus the size of the type being written, or index is negative.
ReadOnlyBufferException (p. 2244)	if this buffer is read-only.

Implemented in **decaf::internal::nio::FloatArrayBuffer** (p. 1366).

6.240.3.21 `virtual FloatBuffer* decaf::nio::FloatBuffer::slice () const` `[pure virtual]`

Creates a new **FloatBuffer** (p. 1368) whose content is a shared subsequence of this buffer's content.

The content of the new buffer will start at this buffer's current position. Changes to this buffer's content will be visible in the new buffer, and vice versa; the two buffers' position, limit, and mark values will be independent.

The new buffer's position will be zero, its capacity and its limit will be the number of bytes remaining in this buffer, and its mark will be undefined. The new buffer will be read-only if, and only if, this buffer is read-only.

Returns

the newly create **FloatBuffer** (p. 1368) which the caller owns.

Implemented in **decaf::internal::nio::FloatArrayBuffer** (p. 1367).

6.240.3.22 `virtual std::string decaf::nio::FloatBuffer::toString () const` `[virtual]`

Returns

a std::string describing this object

6.240.3.23 **static FloatBuffer* decaf::nio::FloatBuffer::wrap (float * array, int size, int offset, int length)** [static]

Wraps the passed buffer with a new **FloatBuffer** (p. 1368).

The new buffer will be backed by the given float array; that is, modifications to the buffer will cause the array to be modified and vice versa. The new buffer's capacity will be array.length, its position will be offset, its limit will be offset + length, and its mark will be undefined. Its backing array will be the given array, and its array offset will be zero.

Parameters

<i>array</i>	The array that will back the new buffer.
<i>size</i>	The size of the array that was passed in.
<i>offset</i>	The offset of the subarray to be used.
<i>length</i>	The length of the subarray to be used.

Returns

a new **FloatBuffer** (p. 1368) that is backed by buffer, caller owns.

Exceptions

<i>NullPointerException</i>	if the array pointer is NULL.
<i>IndexOutOfBoundsException</i>	if the preconditions of size, offset, or length are not met.

6.240.3.24 **static FloatBuffer* decaf::nio::FloatBuffer::wrap (std::vector< float > & buffer)** [static]

Wraps the passed STL float Vector in a **FloatBuffer** (p. 1368).

The new buffer will be backed by the given float array; modifications to the buffer will cause the array to be modified and vice versa. The new buffer's capacity and limit will be buffer.size(), its position will be zero, and its mark will be undefined. Its backing array will be the given array, and its array offset will be zero.

Parameters

<i>buffer</i>	- The vector that will back the new buffer, the vector must have been sized to the desired size already by calling vector.resize(N).
---------------	--

Returns

a new **FloatBuffer** (p. 1368) that is backed by buffer, caller owns.

The documentation for this class was generated from the following file:

- src/main/decaf/nio/**FloatBuffer.h**

6.241 decaf::io::Flushable Class Reference

A **Flushable** (p. 1380) is a destination of data that can be flushed.

```
#include <src/main/decaf/io/Flushable.h>
```

Inheritance diagram for decaf::io::Flushable:

Public Member Functions

- virtual **~Flushable** ()
- virtual void **flush** ()=0

Flushes this stream by writing any buffered output to the underlying stream.

6.241.1 Detailed Description

A **Flushable** (p. 1380) is a destination of data that can be flushed.

The flush method is invoked to write any buffered output to the underlying stream.

Since

1.0

6.241.2 Constructor & Destructor Documentation

6.241.2.1 virtual **decaf::io::Flushable::~~Flushable** () [inline, virtual]

6.241.3 Member Function Documentation

6.241.3.1 virtual void **decaf::io::Flushable::flush** () [pure virtual]

Flushes this stream by writing any buffered output to the underlying stream.

Exceptions

<i>IOException</i> (p. 1545)	if an I/O error occurs.
--	-------------------------

Implemented in **decaf::io::BufferedOutputStream** (p. 597), **decaf::io::FilterOutputStream** (p. 1343), **decaf::io::OutputStreamWriter** (p. 2076), **decaf::io::OutputStream** (p. 2069), **decaf::internal::io::StandardErrorOutputStream** (p. 2530), and **decaf::internal::io::StandardOutputStream** (p. 2533).

The documentation for this class was generated from the following file:

- src/main/decaf/io/**Flushable.h**

6.242 activemq::commands::FlushCommand Class Reference

```
#include <src/main/activemq/commands/FlushCommand.h>
```

Inheritance diagram for activemq::commands::FlushCommand:

Public Member Functions

- **FlushCommand** ()
- virtual **~FlushCommand** ()
- virtual unsigned char **getDataStructureType** () const
*Get the **DataStructure** (p. 1133) Type as defined in CommandTypes.h.*
- virtual **FlushCommand** * **cloneDataStructure** () const
Clone this obbject and return a new instance that the caller now owns, this will be an exact copy of this one.
- virtual void **copyDataStructure** (const **DataStructure** *src)
Copy the contents of the passed object into this objects members, overwriting any existing data.
- virtual std::string **toString** () const
*Returns a string containing the information for this **DataStructure** (p. 1133) such as its type and value of its elements.*
- virtual bool **equals** (const **DataStructure** *value) const
*Compares the **DataStructure** (p. 1133) passed in to this one, and returns if they are equivalent.*
- virtual **Pointer**< **Command** > **visit** (activemq::state::CommandVisitor *visitor)
Allows a Visitor to visit this command and return a response to the command based on the command type being visited.

Static Public Attributes

- static const unsigned char **ID_FLUSHCOMMAND** = 15

6.242.1 Constructor & Destructor Documentation

6.242.1.1 **activemq::commands::FlushCommand::FlushCommand** ()

6.242.1.2 **virtual activemq::commands::FlushCommand::~~FlushCommand** ()
[virtual]

6.242.2 Member Function Documentation

6.242.2.1 `virtual FlushCommand* activemq::commands::-`
`FlushCommand::cloneDataStructure () const`
`[virtual]`

Clone this obbject and return a new instance that the caller now owns, this will be an exact copy of this one.

Returns

new copy of this object.

Implements **activemq::commands::DataStructure** (p. 1133).

6.242.2.2 `virtual void activemq::commands::FlushCommand::copyDataStructure (`
`const DataStructure * src) [virtual]`

Copy the contents of the passed object into this objects members, overwriting any existing data.

Returns

src - Source Object

Reimplemented from **activemq::commands::BaseCommand** (p. 493).

6.242.2.3 `virtual bool activemq::commands::FlushCommand::equals (const`
`DataStructure * value) const [virtual]`

Compares the **DataStructure** (p. 1133) passed in to this one, and returns if they are equivalent.

Equivalent here means that they are of the same type, and that each element of the objects are the same.

Returns

true if **DataStructure** (p. 1133)'s are Equal.

Reimplemented from **activemq::commands::BaseCommand** (p. 494).

6.242.2.4 `virtual unsigned char activemq::commands::FlushCommand::getData-`
`StructureType () const [virtual]`

Get the **DataStructure** (p. 1133) Type as defined in CommandTypes.h.

Returns

The type of the data structure

Implements **activemq::commands::DataStructure** (p. 1137).

6.242.2.5 `virtual std::string activemq::commands::FlushCommand::toString ()`
`const [virtual]`

Returns a string containing the information for this **DataStructure** (p. 1133) such as its type and value of its elements.

Returns

formatted string useful for debugging.

Reimplemented from **activemq::commands::BaseCommand** (p. 498).

6.242.2.6 `virtual Pointer<Command> activemq::commands::FlushCommand::visit (activemq::state::CommandVisitor * visitor)`
`[virtual]`

Allows a Visitor to visit this command and return a response to the command based on the command type being visited.

The command will call the proper processXXX method in the visitor.

Returns

a **Response** (p. 2298) to the visitor being called or NULL if no response.

Implements **activemq::commands::Command** (p. 871).

6.242.3 Field Documentation

6.242.3.1 `const unsigned char activemq::commands::FlushCommand::ID_FLUSHCOMMAND = 15` `[static]`

The documentation for this class was generated from the following file:

- `src/main/activemq/commands/FlushCommand.h`

6.243 **activemq::wireformat::openwire::marshal::generated::FlushCommandMarshaller Class Reference**

Marshaling code for Open Wire Format for **FlushCommandMarshaller** (p. 1383).

```
#include <src/main/activemq/wireformat/openwire/marshal/generated/FlushCommandMarshaller.h>
```

Inheritance diagram for **activemq::wireformat::openwire::marshal::generated::FlushCommandMarshaller**:

Public Member Functions

- **FlushCommandMarshaller** ()
- virtual **~FlushCommandMarshaller** ()
- virtual **commands::DataStructure * createObject** () const
Creates a new instance of the class that this class is a marshaling director for.
- virtual unsigned char **getDataStructureType** () const
Gets the DataStructureType that this class marshals/unmarshals.
- virtual void **tightUnmarshal** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataInputStream** *dataIn, **utils::BooleanStream** *bs)
Tight Un-marhsal to the given stream.
- virtual int **tightMarshal1** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **utils::BooleanStream** *bs)
Tight Marshal to the given stream.
- virtual void **tightMarshal2** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataOutputStream** *dataOut, **utils::BooleanStream** *bs)
Tight Marshal to the given stream.
- virtual void **looseUnmarshal** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataInputStream** *dataIn)
Loose Un-marhsal to the given stream.
- virtual void **looseMarshal** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataOutputStream** *dataOut)
Tight Marshal to the given stream.

6.243.1 Detailed Description

Marshaling code for Open Wire Format for **FlushCommandMarshaller** (p. 1383).

NOTE!: This file is auto generated - do not modify! if you need to make a change, please see the Java Classes in the activemq-openwire-generator module

6.243.2 Constructor & Destructor Documentation

6.243.2.1 **activemq::wireformat::openwire::marshal::generated::FlushCommandMarshaller::FlushCommandMarshaller** ()
[inline]

6.243.2.2 **virtual activemq::wireformat::openwire::marshal::generated::FlushCommandMarshaller::~~FlushCommandMarshaller** () [inline, virtual]

6.243.3 Member Function Documentation

6.243.3.1 `virtual commands::DataStructure* activemq::wireformat::openwire-
::marshal::generated::FlushCommandMarshaller::createObject ()
const [virtual]`

Creates a new instance of the class that this class is a marshaling director for.

Returns

newly allocated Command

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller**
(p. 1120).

6.243.3.2 `virtual unsigned char activemq::wireformat::openwire::marshal-
::generated::FlushCommandMarshaller::getDataStructureType () const
[virtual]`

Gets the DataStructureType that this class marshals/unmarshals.

Returns

byte Id of this classes DataStructureType

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller**
(p. 1122).

6.243.3.3 `virtual void activemq::wireformat::openwire::marshal::generated:-
FlushCommandMarshaller::looseMarshal (OpenWireFormat * format,
commands::DataStructure * command, decaf::io::DataOutputStream * ds
) [virtual]`

Tight Marshal to the given stream.

Parameters

<i>format</i>	- The OpenWireFormat properties
<i>command</i>	- the object to Marshal
<i>ds</i>	- DataOutputStream to marshal to

Exceptions

<i>IOException</i> if an error occurs.
--

Reimplemented from **activemq::wireformat::openwire::marshal::generated::Base-
CommandMarshaller** (p. 500).

6.243

activemq::wireformat::openwire::marshal::generated::FlushCommandMarshaller
Class Reference 1389

6.243.3.4 virtual void **activemq::wireformat::openwire::marshal::generated::FlushCommandMarshaller::looseUnmarshal** (**OpenWireFormat** * *format*, **commands::DataStructure** * *command*, **decaf::io::DataInputStream** * *dis*)
[virtual]

Loose Un-marhsal to the given stream.

Parameters

<i>format</i>	- The OpenWireFormat properties
<i>command</i>	- the object to Un-Marshal
<i>dis</i>	- the DataInputStream to Un-Marshal from

Exceptions

<i>IOException</i>	if an error occurs.
--------------------	---------------------

Reimplemented from **activemq::wireformat::openwire::marshal::generated::BaseCommandMarshaller** (p. 501).

6.243.3.5 virtual int **activemq::wireformat::openwire::marshal::generated::FlushCommandMarshaller::tightMarshal1** (**OpenWireFormat** * *format*, **commands::DataStructure** * *command*, **utils::BooleanStream** * *bs*)
[virtual]

Tight Marhsal to the given stream.

Parameters

<i>format</i>	The OpenWireFormat properties
<i>command</i>	The object to Marshal
<i>bs</i>	The boolean stream to marshal to.

Exceptions

<i>IOException</i>	if an error occurs.
--------------------	---------------------

Reimplemented from **activemq::wireformat::openwire::marshal::generated::BaseCommandMarshaller** (p. 503).

6.243.3.6 virtual void **activemq::wireformat::openwire::marshal::generated::FlushCommandMarshaller::tightMarshal2** (**OpenWireFormat** * *format*, **commands::DataStructure** * *command*, **decaf::io::DataOutputStream** * *ds*, **utils::BooleanStream** * *bs*) [virtual]

Tight Marhsal to the given stream.

Parameters

<i>format</i>	- The OpenwireFormat properties
<i>command</i>	- the object to Marshal
<i>ds</i>	- the DataOutputStream to Marshal to
<i>bs</i>	- boolean stream to marshal to.

Exceptions

<i>IOException</i>	if an error occurs.
--------------------	---------------------

Reimplemented from **activemq::wireformat::openwire::marshal::generated::Base-CommandMarshaller** (p. 504).

6.243.3.7 virtual void **activemq::wireformat::openwire::marshal::generated::Flush-CommandMarshaller::tightUnmarshal** (**OpenWireFormat** * *format*, **commands::DataStructure** * *command*, **decaf::io::DataInputStream** * *dis*, **utils::BooleanStream** * *bs*) [virtual]

Tight Un-marhsal to the given stream.

Parameters

<i>format</i>	- The OpenwireFormat properties
<i>command</i>	- the object to Un-Marshal
<i>dis</i>	- the DataInputStream to Un-Marshal from
<i>bs</i>	- boolean stream to unmarshal from.

Exceptions

<i>IOException</i>	if an error occurs.
--------------------	---------------------

Reimplemented from **activemq::wireformat::openwire::marshal::generated::Base-CommandMarshaller** (p. 505).

The documentation for this class was generated from the following file:

- `src/main/activemq/wireformat/openwire/marshal/generated/FlushCommand-Marshaller.h`

6.244 decaf::util::logging::Formatter Class Reference

A **Formatter** (p. 1387) provides support for formatting LogRecords.

```
#include <src/main/decaf/util/logging/Formatter.h>
```

Inheritance diagram for `decaf::util::logging::Formatter`:

Public Member Functions

- virtual `~Formatter()`
- virtual `std::string format (const LogRecord &record) const =0`
Format the given log record and return the formatted string.
- virtual `std::string formatMessage (const LogRecord &record) const`
Format the message string from a log record.
- virtual `std::string getHead (const Handler *handler DECAF_UNUSED)`
Return the header string for a set of formatted records.
- virtual `std::string getTail (const Handler *handler DECAF_UNUSED)`
Return the tail string for a set of formatted records.

6.244.1 Detailed Description

A **Formatter** (p. 1387) provides support for formatting LogRecords.

Typically each logging **Handler** (p. 1401) will have a **Formatter** (p. 1387) associated with it. The **Formatter** (p. 1387) takes a **LogRecord** (p. 1719) and converts it to a string.

Some formatters (such as the **XMLFormatter** (p.2950)) need to wrap head and tail strings around a set of formatted records. The `getHeader` and `getTail` methods can be used to obtain these strings.

6.244.2 Constructor & Destructor Documentation

6.244.2.1 virtual `decaf::util::logging::Formatter::~Formatter ()` [`inline`, `virtual`]

6.244.3 Member Function Documentation

6.244.3.1 virtual `std::string decaf::util::logging::Formatter::format (const LogRecord &record) const` [`pure virtual`]

Format the given log record and return the formatted string.

Parameters

<i>record</i>	The Log Record to Format
---------------	--------------------------

Returns

the formatted record.

Implemented in **decaf::util::logging::XMLFormatter** (p.2951), and **decaf::util::logging::SimpleFormatter** (p. 2442).

6.244.3.2 virtual std::string decaf::util::logging::Formatter::formatMessage (const LogRecord & *record*) const [virtual]

Format the message string from a log record.

Parameters

<i>record</i>	The Log Record to Format
---------------	--------------------------

Returns

the formatted message

6.244.3.3 virtual std::string decaf::util::logging::Formatter::getHead (const Handler *handler *DECAF_UNUSED*) [inline, virtual]

Return the header string for a set of formatted records.

In the default implementation this method should return empty string.

Parameters

<i>handler</i>	The target handler, can be NULL.
----------------	----------------------------------

Returns

the head string.

6.244.3.4 virtual std::string decaf::util::logging::Formatter::getTail (const Handler *handler *DECAF_UNUSED*) [inline, virtual]

Return the tail string for a set of formatted records.

In the default implementation this method should return empty string

Parameters

<i>handler</i>	the target handler, can be null
----------------	---------------------------------

Returns

the tail string

The documentation for this class was generated from the following file:

- src/main/decaf/util/logging/**Formatter.h**

6.245 decaf::util::concurrent::Future< V > Class Template - Reference

A **Future** (p. 1390) represents the result of an asynchronous computation.

```
#include <src/main/decaf/util/concurrent/Future.h>
```

Public Member Functions

- virtual **~Future** ()
- virtual bool **cancel** (bool mayInterruptIfRunning)=0
Attempts to cancel execution of this task.
- virtual bool **isCancelled** () const =0
Returns true if this task was canceled before it completed normally.
- virtual bool **isDone** () const =0
Returns true if this task completed.
- virtual V **get** ()=0
Waits if necessary for the computation to complete, and then retrieves its result.
- virtual V **get** (long long timeout, const **TimeUnit** &unit)=0
Waits if necessary for at most the given time for the computation to complete, and then retrieves its result, if available.

6.245.1 Detailed Description

```
template<typename V>class decaf::util::concurrent::Future< V >
```

A **Future** (p. 1390) represents the result of an asynchronous computation.

Methods are provided to check if the computation is complete, to wait for its completion, and to retrieve the result of the computation. The result can only be retrieved using method **get** when the computation has completed, blocking if necessary until it is ready. Cancellation is performed by the **cancel** method. Additional methods are provided to determine if the task completed normally or was canceled. Once a computation has completed, the computation cannot be canceled. If you would like to use a **Future** (p. 1390) for the sake of cancellability but not provide a usable result, you can declare types of the form **Future<void*>** and return null as a result of the underlying task.

6.245.2 Constructor & Destructor Documentation

```
6.245.2.1 template<typename V > virtual decaf::util::concurrent::Future< V  
>::~~Future ( ) [inline, virtual]
```

6.245.3 Member Function Documentation

6.245.3.1 `template<typename V > virtual bool decaf::util::concurrent::Future< V
>::cancel (bool mayInterruptIfRunning) [pure virtual]`

Attempts to cancel execution of this task.

This attempt will fail if the task has already completed, has already been canceled, or could not be canceled for some other reason. If successful, and this task has not started when cancel is called, this task should never run. If the task has already started, then the `mayInterruptIfRunning` parameter determines whether the thread executing this task should be interrupted in an attempt to stop the task.

After this method returns, subsequent calls to `isDone()` (p. 1392) will always return true. Subsequent calls to `isCancelled()` (p. 1392) will always return true if this method returned true.

Parameters

<i>may-InterruptIfRunning</i>	- true if the thread executing this task should be interrupted; otherwise, in-progress tasks are allowed to complete.
-------------------------------	---

Returns

false if the task could not be canceled, typically because it has already completed normally; true otherwise

6.245.3.2 `template<typename V > virtual V decaf::util::concurrent::Future< V >::get () [pure virtual]`

Waits if necessary for the computation to complete, and then retrieves its result.

Returns

the computed result.

Exceptions

Cancellation-Exception (p. 748)	- if the computation was canceled
ExecutionException (p. 1294)	- if the computation threw an exception
<i>InterruptedException</i>	- if the current thread was interrupted while waiting

6.245.3.3 `template<typename V > virtual V decaf::util::concurrent::Future< V >::get (long long timeout, const TimeUnit & unit) [pure virtual]`

Waits if necessary for at most the given time for the computation to complete, and then retrieves its result, if available.

Parameters

<i>timeout</i>	- the maximum time to wait
<i>unit</i>	- the time unit of the timeout argument

Returns

the computed result

Exceptions

Cancellation-Exception (p. 748)	- if the computation was canceled
ExecutionException (p. 1294)	- if the computation threw an exception
<i>InterruptedException</i>	- if the current thread was interrupted while waiting
TimeoutException (p. 2737)	- if the wait timed out

6.245.3.4 `template<typename V > virtual bool decaf::util::concurrent::Future< V >::isCancelled () const [pure virtual]`

Returns true if this task was canceled before it completed normally.

Returns

true if this task was canceled before it completed

6.245.3.5 `template<typename V > virtual bool decaf::util::concurrent::Future< V >::isDone () const [pure virtual]`

Returns true if this task completed.

Completion may be due to normal termination, an exception, or cancellation -- in all of these cases, this method will return true.

Returns

true if this task completed

The documentation for this class was generated from the following file:

- src/main/decaf/util/concurrent/**Future.h**

6.246 activemq::transport::correlator::FutureResponse Class - Reference

A container that holds a response object.

```
#include <src/main/activemq/transport/correlator/Future-Response.h>
```

Public Member Functions

- **FutureResponse ()**
- virtual **~FutureResponse ()**
- virtual const **Pointer< Response > & getResponse ()** const
Getters for the response property.
- virtual **Pointer< Response > & getResponse ()**
- virtual const **Pointer< Response > & getResponse (unsigned int timeout)** const
Getters for the response property.
- virtual **Pointer< Response > & getResponse (unsigned int timeout)**
- virtual void **setResponse (const Pointer< Response > &response)**
Setter for the response property.

6.246.1 Detailed Description

A container that holds a response object.

Callers of the `getResponse` method will block until a response has been receive unless they call the `getReponse` that takes a timeout.

6.246.2 Constructor & Destructor Documentation

6.246.2.1 **activemq::transport::correlator::FutureResponse::FutureResponse ()**
[inline]

6.246.2.2 **virtual activemq::transport::correlator::FutureResponse::~~FutureResponse ()** [inline, virtual]

6.246.3 Member Function Documentation

6.246.3.1 `virtual const Pointer<Response>& activemq::transport::correlator::FutureResponse::getResponse () const [inline, virtual]`

Getters for the response property.

Infinite Wait.

Returns

the response object for the request

6.246.3.2 `virtual Pointer<Response>& activemq::transport::correlator::FutureResponse::getResponse () [inline, virtual]`

6.246.3.3 `virtual const Pointer<Response>& activemq::transport::correlator::FutureResponse::getResponse (unsigned int timeout) const [inline, virtual]`

Getters for the response property.

Timed Wait.

Parameters

<i>timeout</i>	- time to wait in milliseconds
----------------	--------------------------------

Returns

the response object for the request

6.246.3.4 `virtual Pointer<Response>& activemq::transport::correlator::FutureResponse::getResponse (unsigned int timeout) [inline, virtual]`

6.246.3.5 `virtual void activemq::transport::correlator::FutureResponse::setResponse (const Pointer< Response > & response) [inline, virtual]`

Setter for the response property.

Parameters

<i>response</i>	the response object for the request.
-----------------	--------------------------------------

The documentation for this class was generated from the following file:

- src/main/activemq/transport/correlator/**FutureResponse.h**

6.247 decaf::security::GeneralSecurityException Class Reference

```
#include <src/main/decaf/security/GeneralSecurityException.h>
```

Inheritance diagram for decaf::security::GeneralSecurityException:

Public Member Functions

- **GeneralSecurityException** () throw ()
Default Constructor.
- **GeneralSecurityException** (const **decaf::lang::Exception** &ex) throw ()
Conversion Constructor from some other Exception.
- **GeneralSecurityException** (const **GeneralSecurityException** &ex) throw ()
Copy Constructor.
- **GeneralSecurityException** (const char *file, const int lineNumber, const std::exception *cause, const char *msg,...) throw ()
Constructor - Initializes the file name and line number where this message occurred.
- **GeneralSecurityException** (const std::exception *cause) throw ()
Constructor.
- **GeneralSecurityException** (const char *file, const int lineNumber, const char *msg,...) throw ()
Constructor - Initializes the file name and line number where this message occurred.
- virtual **GeneralSecurityException** * **clone** () const
Clones this exception.
- virtual ~**GeneralSecurityException** () throw ()

6.247.1 Constructor & Destructor Documentation

6.247.1.1 **decaf::security::GeneralSecurityException::GeneralSecurityException** () throw () [inline]

Default Constructor.

6.247.1.2 **decaf::security::GeneralSecurityException::GeneralSecurityException** (const **decaf::lang::Exception** & ex) throw () [inline]

Conversion Constructor from some other Exception.

Parameters

ex	An exception that should become this type of Exception
-----------	--

6.247.1.3 **decaf::security::GeneralSecurityException::GeneralSecurityException (**
const GeneralSecurityException & ex) throw () `[inline]`

Copy Constructor.

Parameters

<i>ex</i>	An exception that should become this type of Exception
-----------	--

6.247.1.4 **decaf::security::GeneralSecurityException::GeneralSecurityException (**
const char * file, const int lineNumber, const std::exception * cause, const char * msg, ...) throw () `[inline]`

Constructor - Initializes the file name and line number where this message occurred.

Sets the message to report, using an optional list of arguments to parse into the message

Parameters

<i>file</i>	The file name where exception occurs
<i>lineNumber</i>	The line number where the exception occurred.
<i>cause</i>	The exception that was the cause for this one to be thrown.
<i>msg</i>	The message to report
<i>...</i>	list of primitives that are formatted into the message

6.247.1.5 **decaf::security::GeneralSecurityException::GeneralSecurityException (**
const std::exception * cause) throw () `[inline]`

Constructor.

Parameters

<i>cause</i>	Pointer to the exception that caused this one to be thrown, the object is cloned caller retains ownership.
--------------	--

6.247.1.6 **decaf::security::GeneralSecurityException::GeneralSecurityException**
(const char * file, const int lineNumber, const char * msg, ...) throw ()
`[inline]`

Constructor - Initializes the file name and line number where this message occurred.

Sets the message to report, using an optional list of arguments to parse into the message

Parameters

<i>file</i>	name where exception occurs
<i>lineNumber</i>	line number where the exception occurred.
<i>msg</i>	message to report
...	list of primitives that are formatted into the message

6.247.1.7 `virtual decaf::security::GeneralSecurityException::~~GeneralSecurityException () throw () [inline, virtual]`

6.247.2 Member Function Documentation

6.247.2.1 `virtual GeneralSecurityException* decaf::security::GeneralSecurityException::clone () const [inline, virtual]`

Clones this exception.

This is useful for cases where you need to preserve the type of the original exception as well as the message. All subclasses should override.

Returns

A deep copy of this exception.

Reimplemented from `decaf::lang::Exception` (p. 1282).

Reimplemented in `decaf::security::NoSuchProviderException` (p. 1989), `decaf::security::NoSuchAlgorithmException` (p. 1983), `decaf::security::SignatureException` (p. 2440), `decaf::security::InvalidKeyException` (p. 1538), `decaf::security::KeyException` (p. 1603), `decaf::security::KeyManagementException` (p. 1605), `decaf::security::cert::CertificateExpiredException` (p. 760), `decaf::security::cert::CertificateNotYetValidException` (p. 762), `decaf::security::cert::CertificateParsingException` (p. 764), `decaf::security::cert::CertificateEncodingException` (p. 756), and `decaf::security::cert::CertificateException` (p. 758).

The documentation for this class was generated from the following file:

- `src/main/decaf/security/GeneralSecurityException.h`

6.248 decaf::internal::util::GenericResource< T > Class Template Reference

A Generic **Resource** (p. 2293) wraps some type and will delete it when the **Resource** (p. 2293) itself is deleted.

```
#include <src/main/decaf/internal/util/GenericResource.h>
```

Inheritance diagram for decaf::internal::util::GenericResource< T >:

Public Member Functions

- **GenericResource** (T *value)
- virtual **~GenericResource** ()
- T * **getManaged** () const
- void **setManaged** (T *value)

6.248.1 Detailed Description

```
template<typename T>class decaf::internal::util::GenericResource< T >
```

A Generic **Resource** (p. 2293) wraps some type and will delete it when the **Resource** (p. 2293) itself is deleted.

Since

1.0

6.248.2 Constructor & Destructor Documentation

6.248.2.1 `template<typename T> decaf::internal::util::GenericResource< T >::GenericResource (T * value) [inline, explicit]`

6.248.2.2 `template<typename T> virtual decaf::internal::util::GenericResource< T >::~~GenericResource () [inline, virtual]`

6.248.3 Member Function Documentation

6.248.3.1 `template<typename T> T* decaf::internal::util::GenericResource< T >::getManaged () const [inline]`

6.248.3.2 `template<typename T> void decaf::internal::util::GenericResource< T >::setManaged (T * value) [inline]`

The documentation for this class was generated from the following file:

- src/main/decaf/internal/util/**GenericResource.h**

6.249 gz_header_s Struct Reference

```
#include <src/main/decaf/internal/util/zip/zlib.h>
```

Data Fields

- int **text**
- uLong **time**
- int **xflags**
- int **os**
- Bytef * **extra**
- ulInt **extra_len**
- ulInt **extra_max**
- Bytef * **name**
- ulInt **name_max**
- Bytef * **comment**
- ulInt **comm_max**
- int **hcrc**
- int **done**

6.249.1 Field Documentation

6.249.1.1 ulInt gz_header_s::comm_max

6.249.1.2 Bytef* gz_header_s::comment

6.249.1.3 int gz_header_s::done

6.249.1.4 Bytef* gz_header_s::extra

6.249.1.5 ulInt gz_header_s::extra_len

6.249.1.6 ulInt gz_header_s::extra_max

6.249.1.7 int gz_header_s::hcrc

6.249.1.8 Bytef* gz_header_s::name

6.249.1.9 ulInt gz_header_s::name_max

6.249.1.10 int gz_header_s::os

6.249.1.11 int gz_header_s::text

6.249.1.12 uLong gz_header_s::time

6.249.1.13 int gz_header_s::xflags

The documentation for this struct was generated from the following file:

- src/main/decaf/internal/util/zip/**zlib.h**

6.250 gz_state Struct Reference

```
#include <src/main/decaf/internal/util/zip/gzguts.h>
```

Data Fields

- int **mode**
- int **fd**
- char * **path**
- **z_off64_t** **pos**
- unsigned **size**
- unsigned **want**
- unsigned char * **in**
- unsigned char * **out**
- unsigned char * **next**
- unsigned **have**
- int **eof**
- **z_off64_t** **start**
- **z_off64_t** **raw**
- int **how**
- int **direct**
- int **level**
- int **strategy**
- **z_off64_t** **skip**
- int **seek**
- int **err**
- char * **msg**
- **z_stream** **strm**

6.250.1 Field Documentation

6.250.1.1 int gz_state::direct

6.250.1.2 int gz_state::eof

6.250.1.3 int gz_state::err

6.250.1.4 int gz_state::fd

6.250.1.5 unsigned gz_state::have

6.250.1.6 int gz_state::how

6.250.1.7 unsigned char* gz_state::in

- 6.250.1.8 `int gz_state::level`
- 6.250.1.9 `int gz_state::mode`
- 6.250.1.10 `char* gz_state::msg`
- 6.250.1.11 `unsigned char* gz_state::next`
- 6.250.1.12 `unsigned char* gz_state::out`
- 6.250.1.13 `char* gz_state::path`
- 6.250.1.14 `z_off64_t gz_state::pos`
- 6.250.1.15 `z_off64_t gz_state::raw`
- 6.250.1.16 `int gz_state::seek`
- 6.250.1.17 `unsigned gz_state::size`
- 6.250.1.18 `z_off64_t gz_state::skip`
- 6.250.1.19 `z_off64_t gz_state::start`
- 6.250.1.20 `int gz_state::strategy`
- 6.250.1.21 `z_stream gz_state::strm`
- 6.250.1.22 `unsigned gz_state::want`

The documentation for this struct was generated from the following file:

- `src/main/decaf/internal/util/zip/gzguts.h`

6.251 decaf::util::logging::Handler Class Reference

A **Handler** (p. 1401) object takes log messages from a **Logger** (p. 1693) and exports them.

```
#include <src/main/decaf/util/logging/Handler.h>
```

Inheritance diagram for `decaf::util::logging::Handler`:

Public Member Functions

- **Handler** ()

- virtual `~Handler ()`
- virtual void `flush ()=0`
*Flush the **Handler** (p. 1401)'s output, clears any buffers.*
- virtual void `publish (const LogRecord &record)=0`
*Publish the Log Record to this **Handler** (p. 1401).*
- virtual bool `isLoggable (const LogRecord &record) const`
*Check if this **Handler** (p. 1401) would actually log a given **LogRecord** (p. 1719).*
- virtual void `setFilter (Filter *filter)`
*Sets the **Filter** (p. 1333) that this **Handler** (p. 1401) uses to filter Log Records.*
- virtual `Filter * getFilter ()`
*Gets the **Filter** (p. 1333) that this **Handler** (p. 1401) uses to filter Log Records.*
- virtual void `setLevel (const Level &value)`
***Set** (p. 2397) the log level specifying which message levels will be logged by this - **Handler** (p. 1401).*
- virtual `Level getLevel ()`
*Get the log level specifying which message levels will be logged by this **Handler** (p. 1401).*
- virtual void `setFormatter (Formatter *formatter)`
*Sets the **Formatter** (p. 1387) used by this **Handler** (p. 1401).*
- virtual `Formatter * getFormatter ()`
*Gets the **Formatter** (p. 1387) used by this **Handler** (p. 1401).*
- virtual void `setErrorManager (ErrorManager *errorManager)`
*Sets the **Formatter** (p. 1387) used by this **Handler** (p. 1401).*
- virtual `ErrorManager * getErrorManager ()`
*Gets the **ErrorManager** (p. 1277) used by this **Handler** (p. 1401).*

Protected Member Functions

- void `reportError (const std::string &message, decaf::lang::Exception *ex, int code)`
*Protected convenience method to report an error to this **Handler** (p. 1401)'s **ErrorManager** (p. 1277).*

6.251.1 Detailed Description

A **Handler** (p. 1401) object takes log messages from a **Logger** (p. 1693) and exports them.

It might for example, write them to a console or write them to a file, or send them to a network logging service, or forward them to an OS log, or whatever.

A **Handler** (p. 1401) can be disabled by doing a `setLevel(Level.OFF (p. 1620))` and can be re-enabled by doing a `setLevel` with an appropriate level.

Handler (p. 1401) classes typically use **LogManager** (p. 1712) properties to set default values for the **Handler** (p. 1401)'s **Filter** (p. 1333), **Formatter** (p. 1387), and **Level** (p. 1616). See the specific documentation for each concrete **Handler** (p. 1401) class.

6.251.2 Constructor & Destructor Documentation

6.251.2.1 `decaf::util::logging::Handler::Handler ()`

6.251.2.2 `virtual decaf::util::logging::Handler::~~Handler ()` [virtual]

6.251.3 Member Function Documentation

6.251.3.1 `virtual void decaf::util::logging::Handler::flush ()` [pure virtual]

Flush the **Handler** (p. 1401)'s output, clears any buffers.

Implemented in **decaf::util::logging::StreamHandler** (p. 2605).

6.251.3.2 `virtual ErrorManager* decaf::util::logging::Handler::getErrorManager ()`
[inline, virtual]

Gets the **ErrorManager** (p. 1277) used by this **Handler** (p. 1401).

Returns

ErrorManager (p. 1277) derived pointer or NULL.

6.251.3.3 `virtual Filter* decaf::util::logging::Handler::getFilter ()` [inline, virtual]

Gets the **Filter** (p. 1333) that this **Handler** (p. 1401) uses to filter Log Records.

Returns

Filter (p. 1333) derived instance

6.251.3.4 `virtual Formatter* decaf::util::logging::Handler::getFormatter ()`
[inline, virtual]

Gets the **Formatter** (p. 1387) used by this **Handler** (p. 1401).

Returns

Filter (p. 1333) derived instance

6.251.3.5 `virtual Level decaf::util::logging::Handler::getLevel ()` [inline, virtual]

Get the log level specifying which message levels will be logged by this **Handler** (p. 1401).

Returns

Level (p. 1616) enumeration value

6.251.3.6 virtual bool **decaf::util::logging::Handler::isLoggable** (const **LogRecord** & *record*) const [virtual]

Check if this **Handler** (p. 1401) would actually log a given **LogRecord** (p. 1719).

This method checks if the **LogRecord** (p. 1719) has an appropriate **Level** (p. 1616) and whether it satisfies any **Filter** (p. 1333). It also may make other **Handler** (p. 1401) specific checks that might prevent a handler from logging the **LogRecord** (p. 1719).

Parameters

<i>record</i>	LogRecord (p. 1719) to check
---------------	-------------------------------------

Reimplemented in **decaf::util::logging::StreamHandler** (p. 2605).

6.251.3.7 virtual void **decaf::util::logging::Handler::publish** (const **LogRecord** & *record*) [pure virtual]

Publish the Log Record to this **Handler** (p. 1401).

Parameters

<i>record</i>	The Log Record to Publish
---------------	---------------------------

Implemented in **decaf::util::logging::StreamHandler** (p. 2605), and **decaf::util::logging::ConsoleHandler** (p. 991).

6.251.3.8 void **decaf::util::logging::Handler::reportError** (const std::string & *message*, **decaf::lang::Exception** * *ex*, int *code*) [protected]

Protected convenience method to report an error to this **Handler** (p. 1401)'s **ErrorManager** (p. 1277).

Parameters

<i>message</i>	- a descriptive string (may be empty)
<i>ex</i>	- an exception (may be NULL)
<i>code</i>	- an error code defined in EventManager (p. 1277)

6.251.3.9 virtual void **decaf::util::logging::Handler::setErrorHandler** (**EventManager** * *errorManager*) [virtual]

Sets the **Formatter** (p. 1387) used by this **Handler** (p. 1401).

The **ErrorManager** (p. 1277)'s "error" method will be invoked if any errors occur while using this **Handler** (p. 1401).

Parameters

<i>error-Manager</i>	ErrorManager (p. 1277) derived instance
----------------------	--

6.251.3.10 virtual void **decaf::util::logging::Handler::setFilter** (**Filter** * *filter*)
[inline, virtual]

Sets the **Filter** (p. 1333) that this **Handler** (p. 1401) uses to filter Log Records.

For each call of publish the **Handler** (p. 1401) will call this **Filter** (p. 1333) (if it is non-null) to check if the **LogRecord** (p. 1719) should be published or discarded.

Parameters

<i>filter</i>	Filter (p. 1333) derived instance
---------------	--

6.251.3.11 virtual void **decaf::util::logging::Handler::setFormatter** (**Formatter** * *formatter*) [virtual]

Sets the **Formatter** (p. 1387) used by this **Handler** (p. 1401).

Some Handlers may not use Formatters, in which case the **Formatter** (p. 1387) will be remembered, but not used.

Parameters

<i>formatter</i>	Filter (p. 1333) derived instance
------------------	--

6.251.3.12 virtual void **decaf::util::logging::Handler::setLevel** (const **Level** & *value*)
[inline, virtual]

Set (p. 2397) the log level specifying which message levels will be logged by this - **Handler** (p. 1401).

The intention is to allow developers to turn on voluminous logging, but to limit the messages that are sent to certain Handlers.

Parameters

<i>value</i>	Level (p. 1616) enumeration value
--------------	--

The documentation for this class was generated from the following file:

- src/main/decaf/util/logging/**Handler.h**

6.252 decaf::internal::util::HexStringParser Class Reference

```
#include <src/main/decaf/internal/util/HexStringParser.h>
```

Public Member Functions

- **HexStringParser** (int exponentWidth, int mantissaWidth)
Create a new HexParser.
- virtual **~HexStringParser** ()
- long long **parse** (const std::string &hexString)
Parses a hex string using the specs given in the constructor and returns a long long with the bits of the parsed string, the caller can then convert those to a float or double as needed.

Static Public Member Functions

- static double **parseDouble** (const std::string &hexString)
- static float **parseFloat** (const std::string &hexString)

6.252.1 Constructor & Destructor Documentation

6.252.1.1 decaf::internal::util::HexStringParser::HexStringParser (int exponentWidth, int mantissaWidth)

Create a new HexParser.

Parameters

<i>exponent-Width</i>	- Width of the exponent for the type to parse
<i>mantissa-Width</i>	- Width of the mantissa for the type to parse

6.252.1.2 virtual decaf::internal::util::HexStringParser::~~HexStringParser () [inline, virtual]

6.252.2 Member Function Documentation

6.252.2.1 long long decaf::internal::util::HexStringParser::parse (const std::string &hexString)

Parses a hex string using the specs given in the constructor and returns a long long with the bits of the parsed string, the caller can then convert those to a float or double as needed.

Parameters

<i>hexString</i>	- string to parse
------------------	-------------------

Returns

the bits parsed from the string

6.252.2.2 static double decaf::internal::util::HexStringParser::parseDouble (const std::string & *hexString*) [static]

6.252.2.3 static float decaf::internal::util::HexStringParser::parseFloat (const std::string & *hexString*) [static]

The documentation for this class was generated from the following file:

- src/main/decaf/internal/util/**HexStringParser.h**

6.253 activemq::wireformat::openwire::utils::HexTable Class - Reference

The **HexTable** (p. 1407) class maps hexadecimal strings to the value of an index into the table, i.e.

```
#include <src/main/activemq/wireformat/openwire/utils/-
HexTable.h>
```

Public Member Functions

- **HexTable** ()
- virtual ~**HexTable** ()
- virtual const std::string & **operator[]** (std::size_t index)
Index operator for this Table, will throw an exception if the index requested is out of bounds for this table.
- virtual const std::string & **operator[]** (std::size_t index) const
- virtual std::size_t **size** () const
Returns the max size of this Table.

6.253.1 Detailed Description

The **HexTable** (p. 1407) class maps hexadecimal strings to the value of an index into the table, i.e.

the class will return "FF" for the index 255 in the table.

6.253.2 Constructor & Destructor Documentation

6.253.2.1 `activemq::wireformat::openwire::utils::HexTable::HexTable ()`

6.253.2.2 `virtual activemq::wireformat::openwire::utils::HexTable::~~HexTable ()`
`[inline, virtual]`

6.253.3 Member Function Documentation

6.253.3.1 `virtual const std::string& activemq::wireformat::openwire::utils::HexTable::operator[] (`
`std::size_t index)` `[virtual]`

Index operator for this Table, will throw an exception if the index requested is out of bounds for this table.

Parameters

<i>index</i>	The index of the value in the table to fetch.
--------------	---

Returns

string containing the hex value if the index

Exceptions

<i>IndexOutOfBounds-Exception</i>	if the index exceeds the table size.
-----------------------------------	--------------------------------------

6.253.3.2 `virtual const std::string& activemq::wireformat::openwire::utils::HexTable::operator[] (`
`std::size_t index) const` `[virtual]`

6.253.3.3 `virtual std::size_t activemq::wireformat::openwire::utils::HexTable::size (`
`) const` `[inline, virtual]`

Returns the max size of this Table.

Returns

an integer size value for the table.

The documentation for this class was generated from the following file:

- `src/main/activemq/wireformat/openwire/utils/HexTable.h`

6.254 decaf::net::HttpRetryException Class Reference

```
#include <src/main/decaf/net/HttpRetryException.h>
```

Inheritance diagram for decaf::net::HttpRetryException:

Public Member Functions

- **HttpRetryException** () throw ()
Default Constructor.
- **HttpRetryException** (const **Exception** &ex) throw ()
Conversion Constructor from some other Exception.
- **HttpRetryException** (const **HttpRetryException** &ex) throw ()
Copy Constructor.
- **HttpRetryException** (const char *file, const int lineNumber, const std::exception *cause, const char *msg,...) throw ()
Constructor - Initializes the file name and line number where this message occurred.
- **HttpRetryException** (const std::exception *cause) throw ()
Constructor.
- **HttpRetryException** (const char *file, const int lineNumber, const char *msg,...) throw ()
Constructor - Initializes the file name and line number where this message occurred.
- virtual **HttpRetryException** * **clone** () const
Clones this exception.
- virtual ~**HttpRetryException** () throw ()

6.254.1 Constructor & Destructor Documentation

6.254.1.1 **decaf::net::HttpRetryException::HttpRetryException** () throw ()
[inline]

Default Constructor.

6.254.1.2 **decaf::net::HttpRetryException::HttpRetryException** (const **Exception** &ex) throw () [inline]

Conversion Constructor from some other Exception.

Parameters

ex	An exception that should become this type of Exception
----	--

6.254.1.3 **decaf::net::HttpRetryException::HttpRetryException** (const **HttpRetryException** &ex) throw () [inline]

Copy Constructor.

Parameters

<i>ex</i>	An exception that should become this type of Exception
-----------	--

6.254.1.4 decaf::net::HttpRetryException::HttpRetryException (const char * *file*, const int *lineNumber*, const std::exception * *cause*, const char * *msg*, ...) throw () `[inline]`

Constructor - Initializes the file name and line number where this message occurred.

Sets the message to report, using an optional list of arguments to parse into the message

Parameters

<i>file</i>	The file name where exception occurs
<i>lineNumber</i>	The line number where the exception occurred.
<i>cause</i>	The exception that was the cause for this one to be thrown.
<i>msg</i>	The message to report
...	list of primitives that are formatted into the message

6.254.1.5 decaf::net::HttpRetryException::HttpRetryException (const std::exception * *cause*) throw () `[inline]`

Constructor.

Parameters

<i>cause</i>	Pointer to the exception that caused this one to be thrown, the object is cloned caller retains ownership.
--------------	--

6.254.1.6 decaf::net::HttpRetryException::HttpRetryException (const char * *file*, const int *lineNumber*, const char * *msg*, ...) throw () `[inline]`

Constructor - Initializes the file name and line number where this message occurred.

Sets the message to report, using an optional list of arguments to parse into the message

Parameters

<i>file</i>	The file name where exception occurs
<i>lineNumber</i>	The line number where the exception occurred.
<i>msg</i>	The message to report
...	list of primitives that are formatted into the message

6.254.1.7 `virtual decaf::net::HttpRetryException::~~HttpRetryException () throw ()`
`[inline, virtual]`

6.254.2 Member Function Documentation

6.254.2.1 `virtual HttpRetryException* decaf::net::HttpRetryException::clone ()`
`const [inline, virtual]`

Clones this exception.

This is useful for cases where you need to preserve the type of the original exception as well as the message. All subclasses should override.

Returns

a new Exception instance that is a copy of this Exception object.

Reimplemented from **decaf::io::IOException** (p. 1547).

The documentation for this class was generated from the following file:

- `src/main/decaf/net/HttpRetryException.h`

6.255 activemq::util::IdGenerator Class Reference

```
#include <src/main/activemq/util/IdGenerator.h>
```

Public Member Functions

- **IdGenerator** ()
- **IdGenerator** (const std::string &prefix)
- virtual **~IdGenerator** ()
- std::string **generateId** () const

Static Public Member Functions

- static std::string **getHostname** ()
Since the initialization of this object results in the retrieval of the machine's host name we can quickly return it here.
- static std::string **getSeedFromId** (const std::string &id)
Gets the seed value from a Generated Id, the count portion is removed.
- static long long **getSequenceFromId** (const std::string &id)
Gets the count value from a Generated Id, the seed portion is removed.
- static int **compare** (const std::string &id1, const std::string &id2)
Compares two generated id values.

Friends

- class **activemq::library::ActiveMQCPP**

6.255.1 Constructor & Destructor Documentation

6.255.1.1 **activemq::util::IdGenerator::IdGenerator ()**

6.255.1.2 **activemq::util::IdGenerator::IdGenerator (const std::string & *prefix*)**

6.255.1.3 **virtual activemq::util::IdGenerator::~~IdGenerator ()** [virtual]

6.255.2 Member Function Documentation

6.255.2.1 **static int activemq::util::IdGenerator::compare (const std::string & *id1*, const std::string & *id2*)** [static]

Compares two generated id values.

Parameters

<i>id1</i>	The first id to compare, or left hand side.
<i>id2</i>	The second id to compare, or right hand side.

Returns

zero if ids are equal or positove if *id1* > *id2*...

6.255.2.2 **std::string activemq::util::IdGenerator::generateId ()** const

Returns

a newly generated unique id.

6.255.2.3 **static std::string activemq::util::IdGenerator::getHostname ()**
[static]

Since the initialization of this object results in the retrieval of the machine's host name we can quickly return it here.

Returns

the previously retrieved host name.

6.255.2.4 `static std::string activemq::util::IdGenerator::getSeedFromId (const std::string & id) [static]`

Gets the seed value from a Generated Id, the count portion is removed.

Returns

the seed portion of the Id, minus the count value.

6.255.2.5 `static long long activemq::util::IdGenerator::getSequenceFromId (const std::string & id) [static]`

Gets the count value from a Generated Id, the seed portion is removed.

Returns

the sequence count portion of the id, minus the seed value.

6.255.3 Friends And Related Function Documentation

6.255.3.1 `friend class activemq::library::ActiveMQCPP [friend]`

The documentation for this class was generated from the following file:

- `src/main/activemq/util/IdGenerator.h`

6.256 decaf::lang::exceptions::IllegalArgumentException Class - Reference

```
#include <src/main/decaf/lang/exceptions/IllegalArgument-Exception.h>
```

Inheritance diagram for `decaf::lang::exceptions::IllegalArgumentException`:

Public Member Functions

- **IllegalArgumentException** () throw ()
Default Constructor.
- **IllegalArgumentException** (const **Exception** &ex) throw ()
*Conversion Constructor from some other **Exception** (p. 1279).*
- **IllegalArgumentException** (const **IllegalArgumentException** &ex) throw ()
Copy Constructor.

- **IllegalArgumentException** (const std::exception ***cause**) throw ()
Constructor.
- **IllegalArgumentException** (const char *file, const int lineNumber, const char *msg,...) throw ()
Constructor - Initializes the file name and line number where this message occurred.
- **IllegalArgumentException** (const char *file, const int lineNumber, const std::exception ***cause**, const char *msg,...) throw ()
Constructor - Initializes the file name and line number where this message occurred.
- virtual **IllegalArgumentException** * **clone** () const
Clones this exception.
- virtual ~**IllegalArgumentException** () throw ()

6.256.1 Constructor & Destructor Documentation

6.256.1.1 decaf::lang::exceptions::IllegalArgumentException::IllegalArgumentException () throw () [inline]

Default Constructor.

6.256.1.2 decaf::lang::exceptions::IllegalArgumentException::IllegalArgumentException (const Exception & ex) throw () [inline]

Conversion Constructor from some other **Exception** (p. 1279).

Parameters

ex	The Exception (p. 1279) whose data is to be copied into this one.
-----------	--

6.256.1.3 decaf::lang::exceptions::IllegalArgumentException::IllegalArgumentException (const IllegalArgumentException & ex) throw () [inline]

Copy Constructor.

Parameters

ex	The Exception (p. 1279) whose data is to be copied into this one.
-----------	--

6.256.1.4 decaf::lang::exceptions::IllegalArgumentException::IllegalArgumentException (const std::exception * cause) throw () [inline]

Constructor.

Parameters

<i>cause</i>	Pointer (p. 2083) to the exception that caused this one to be thrown, the object is cloned caller retains ownership.
--------------	---

6.256.1.5 `decaf::lang::exceptions::IllegalArgumentException::IllegalArgument-Exception (const char * file, const int lineNumber, const char * msg, ...) throw () [inline]`

Constructor - Initializes the file name and line number where this message occurred.

Sets the message to report, using an optional list of arguments to parse into the message

Parameters

<i>file</i>	The file name where exception occurs
<i>lineNumber</i>	The line number where the exception occurred.
<i>msg</i>	The message to report
<i>...</i>	list of primitives that are formatted into the message

6.256.1.6 `decaf::lang::exceptions::IllegalArgumentException::IllegalArgument-Exception (const char * file, const int lineNumber, const std::exception * cause, const char * msg, ...) throw () [inline]`

Constructor - Initializes the file name and line number where this message occurred.

Sets the message to report, using an optional list of arguments to parse into the message

Parameters

<i>file</i>	The file name where exception occurs
<i>lineNumber</i>	The line number where the exception occurred.
<i>cause</i>	The exception that was the cause for this one to be thrown.
<i>msg</i>	The message to report
<i>...</i>	list of primitives that are formatted into the message

6.256.1.7 `virtual decaf::lang::exceptions::IllegalArgumentException::~IllegalArgumentException () throw () [inline, virtual]`

6.256.2 Member Function Documentation

6.257 decaf::lang::exceptions::IllegalMonitorStateException Class Reference 419

6.256.2.1 virtual **IllegalArgumentException*** decaf::lang::exceptions-
::**IllegalArgumentException::clone** () const [inline,
virtual]

Clones this exception.

This is useful for cases where you need to preserve the type of the original exception as well as the message. All subclasses should override.

Returns

an new **Exception** (p. 1279) instance that is a copy of this one.

Reimplemented from **decaf::lang::Exception** (p. 1282).

The documentation for this class was generated from the following file:

- src/main/decaf/lang/exceptions/**IllegalArgumentException.h**

6.257 decaf::lang::exceptions::IllegalMonitorStateException Class Reference

```
#include <src/main/decaf/lang/exceptions/IllegalMonitor-  
StateException.h>
```

Inheritance diagram for decaf::lang::exceptions::IllegalMonitorStateException:

Public Member Functions

- **IllegalMonitorStateException** () throw ()
Default Constructor.
- **IllegalMonitorStateException** (const **Exception** &ex) throw ()
*Conversion Constructor from some other **Exception** (p. 1279).*
- **IllegalMonitorStateException** (const **IllegalMonitorStateException** &ex) throw ()
Copy Constructor.
- **IllegalMonitorStateException** (const char *file, const int lineNumber, const char *msg,...) throw ()
Constructor - Initializes the file name and line number where this message occurred.
- **IllegalMonitorStateException** (const char *file, const int lineNumber, const std::exception *cause, const char *msg,...) throw ()
Constructor - Initializes the file name and line number where this message occurred.
- **IllegalMonitorStateException** (const std::exception *cause) throw ()
Constructor.

- virtual **IllegalMonitorStateException** * **clone** () const
Clones this exception.
- virtual ~**IllegalMonitorStateException** () throw ()

6.257.1 Constructor & Destructor Documentation

6.257.1.1 **decaf::lang::exceptions::IllegalMonitorStateException::IllegalMonitorStateException** () throw ()
[inline]

Default Constructor.

6.257.1.2 **decaf::lang::exceptions::IllegalMonitorStateException::IllegalMonitorStateException** (const **Exception** & *ex*) throw ()
[inline]

Conversion Constructor from some other **Exception** (p. 1279).

Parameters

<i>ex</i>	The Exception (p. 1279) whose data is to be copied into this one.
-----------	--

6.257.1.3 **decaf::lang::exceptions::IllegalMonitorStateException::IllegalMonitorStateException** (const **IllegalMonitorStateException** & *ex*) throw ()
[inline]

Copy Constructor.

Parameters

<i>ex</i>	The Exception (p. 1279) whose data is to be copied into this one.
-----------	--

6.257.1.4 **decaf::lang::exceptions::IllegalMonitorStateException::IllegalMonitorStateException** (const char * *file*, const int *lineNumber*, const char * *msg*, ...) throw () [inline]

Constructor - Initializes the file name and line number where this message occurred.

Sets the message to report, using an optional list of arguments to parse into the message

Parameters

<i>file</i>	The file name where exception occurs
<i>lineNumber</i>	The line number where the exception occurred.
<i>msg</i>	The message to report
...	list of primitives that are formatted into the message

6.257 decaf::lang::exceptions::IllegalMonitorStateException Class Reference ¶ 421

6.257.1.5 **decaf::lang::exceptions::IllegalMonitorStateException::IllegalMonitorStateException** (const char * *file*, const int *lineNumber*, const std::exception * *cause*, const char * *msg*, ...) throw () [inline]

Constructor - Initializes the file name and line number where this message occurred.

Sets the message to report, using an optional list of arguments to parse into the message

Parameters

<i>file</i>	The file name where exception occurs
<i>lineNumber</i>	The line number where the exception occurred.
<i>cause</i>	The exception that was the cause for this one to be thrown.
<i>msg</i>	The message to report
...	list of primitives that are formatted into the message

6.257.1.6 **decaf::lang::exceptions::IllegalMonitorStateException::IllegalMonitorStateException** (const std::exception * *cause*) throw () [inline]

Constructor.

Parameters

<i>cause</i>	Pointer (p. 2083) to the exception that caused this one to be thrown, the object is cloned caller retains ownership.
--------------	---

6.257.1.7 **virtual decaf::lang::exceptions::IllegalMonitorStateException::~~IllegalMonitorStateException** () throw () [inline, virtual]

6.257.2 Member Function Documentation

6.257.2.1 **virtual IllegalMonitorStateException* decaf::lang::exceptions::IllegalMonitorStateException::clone** () const [inline, virtual]

Clones this exception.

This is useful for cases where you need to preserve the type of the original exception as well as the message. All subclasses should override.

Returns

an new **Exception** (p. 1279) instance that is a copy of this one.

Reimplemented from **decaf::lang::Exception** (p. 1282).

The documentation for this class was generated from the following file:

- `src/main/decaf/lang/exceptions/IllegalMonitorStateException.h`

6.258 cms::IllegalStateException Class Reference

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

```
#include <src/main/cms/IllegalStateException.h>
```

Inheritance diagram for cms::IllegalStateException:

Public Member Functions

- **IllegalStateException** ()
- **IllegalStateException** (const **IllegalStateException** &ex)
- **IllegalStateException** (const std::string &message)
- **IllegalStateException** (const std::string &message, const std::exception *cause)
- **IllegalStateException** (const std::string &message, const std::exception *cause, const std::vector< std::pair< std::string, int > > &stackTrace)
- virtual ~**IllegalStateException** () throw ()

6.258.1 Detailed Description

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

For example, this exception must be thrown if **Session.commit** (p. 2366) is called on a non-transacted session.

Since

1.3

6.258.2 Constructor & Destructor Documentation

6.258.2.1 cms::IllegalStateException::IllegalStateException ()

6.258.2.2 cms::IllegalStateException::IllegalStateException (const
IllegalStateException & ex)

6.258.2.3 cms::IllegalStateException::IllegalStateException (const std::string &
message)

- 6.258.2.4 `cms::IllegalStateException::IllegalStateException (const std::string & message, const std::exception * cause)`
- 6.258.2.5 `cms::IllegalStateException::IllegalStateException (const std::string & message, const std::exception * cause, const std::vector< std::pair< std::string, int > > & stackTrace)`
- 6.258.2.6 `virtual cms::IllegalStateException::~~IllegalStateException () throw ()`
[virtual]

The documentation for this class was generated from the following file:

- `src/main/cms/IllegalStateException.h`

6.259 decaf::lang::exceptions::IllegalStateException Class Reference

```
#include <src/main/decaf/lang/exceptions/IllegalStateException.h>
```

Inheritance diagram for `decaf::lang::exceptions::IllegalStateException`:

Public Member Functions

- **IllegalStateException** () throw ()
Default Constructor.
- **IllegalStateException** (const **Exception** &ex) throw ()
*Conversion Constructor from some other **Exception** (p. 1279).*
- **IllegalStateException** (const **IllegalStateException** &ex) throw ()
Copy Constructor.
- **IllegalStateException** (const char *file, const int lineNumber, const char *msg,...) throw ()
Constructor - Initializes the file name and line number where this message occurred.
- **IllegalStateException** (const char *file, const int lineNumber, const std::exception *cause, const char *msg,...) throw ()
Constructor - Initializes the file name and line number where this message occurred.
- **IllegalStateException** (const std::exception *cause) throw ()
Constructor.
- virtual **IllegalStateException** * clone () const
Clones this exception.
- virtual ~**IllegalStateException** () throw ()

6.259.1 Constructor & Destructor Documentation

6.259.1.1 `decaf::lang::exceptions::IllegalStateException::IllegalStateException () throw () [inline]`

Default Constructor.

6.259.1.2 `decaf::lang::exceptions::IllegalStateException::IllegalStateException (const Exception & ex) throw () [inline]`

Conversion Constructor from some other **Exception** (p. 1279).

Parameters

<i>ex</i>	The Exception (p. 1279) whose data is to be copied into this one.
-----------	--

6.259.1.3 `decaf::lang::exceptions::IllegalStateException::IllegalStateException (const IllegalStateException & ex) throw () [inline]`

Copy Constructor.

Parameters

<i>ex</i>	The Exception (p. 1279) whose data is to be copied into this one.
-----------	--

6.259.1.4 `decaf::lang::exceptions::IllegalStateException::IllegalStateException (const char * file, const int lineNumber, const char * msg, ...) throw () [inline]`

Constructor - Initializes the file name and line number where this message occurred.

Sets the message to report, using an optional list of arguments to parse into the message

Parameters

<i>file</i>	The file name where exception occurs
<i>lineNumber</i>	The line number where the exception occurred.
<i>msg</i>	The message to report
<i>...</i>	list of primitives that are formatted into the message

6.259.1.5 `decaf::lang::exceptions::IllegalStateException::IllegalStateException (const char * file, const int lineNumber, const std::exception * cause, const char * msg, ...) throw () [inline]`

Constructor - Initializes the file name and line number where this message occurred.

Sets the message to report, using an optional list of arguments to parse into the message

Parameters

<i>file</i>	The file name where exception occurs
<i>lineNumber</i>	The line number where the exception occurred.
<i>cause</i>	The exception that was the cause for this one to be thrown.
<i>msg</i>	The message to report
<i>...</i>	list of primitives that are formatted into the message

6.259.1.6 `decaf::lang::exceptions::IllegalStateException::IllegalStateException (const std::exception * cause) throw () [inline]`

Constructor.

Parameters

<i>cause</i>	Pointer (p. 2083) to the exception that caused this one to be thrown, the object is cloned caller retains ownership.
--------------	---

6.259.1.7 `virtual decaf::lang::exceptions::IllegalStateException::~~IllegalStateException () throw () [inline, virtual]`

6.259.2 Member Function Documentation

6.259.2.1 `virtual IllegalStateException* decaf::lang::exceptions::IllegalStateException::clone () const [inline, virtual]`

Clones this exception.

This is useful for cases where you need to preserve the type of the original exception as well as the message. All subclasses should override.

Returns

an new **Exception** (p. 1279) instance that is a copy of this one.

Reimplemented from **decaf::lang::Exception** (p. 1282).

Reimplemented in **decaf::nio::InvalidMarkException** (p. 1541).

The documentation for this class was generated from the following file:

- `src/main/decaf/lang/exceptions/IllegalStateException.h`

6.260 decaf::lang::exceptions::IllegalThreadStateException Class - Reference

```
#include <src/main/decaf/lang/exceptions/IllegalThread-
StateException.h>
```

Inheritance diagram for decaf::lang::exceptions::IllegalThreadStateException:

Public Member Functions

- **IllegalThreadStateException** () throw ()
Default Constructor.
- **IllegalThreadStateException** (const **Exception** &ex) throw ()
*Conversion Constructor from some other **Exception** (p. 1279).*
- **IllegalThreadStateException** (const **IllegalThreadStateException** &ex) throw ()
Copy Constructor.
- **IllegalThreadStateException** (const char *file, const int lineNumber, const char *msg,...) throw ()
Constructor - Initializes the file name and line number where this message occurred.
- **IllegalThreadStateException** (const char *file, const int lineNumber, const std::exception *cause, const char *msg,...) throw ()
Constructor - Initializes the file name and line number where this message occurred.
- **IllegalThreadStateException** (const std::exception *cause) throw ()
Constructor.
- virtual **IllegalThreadStateException** * **clone** () const
Clones this exception.
- virtual ~**IllegalThreadStateException** () throw ()

6.260.1 Constructor & Destructor Documentation

6.260.1.1 **decaf::lang::exceptions::IllegalThreadStateException::IllegalThreadStateException () throw ()** [inline]

Default Constructor.

6.260.1.2 **decaf::lang::exceptions::IllegalThreadStateException::IllegalThreadStateException (const Exception & ex) throw ()** [inline]

Conversion Constructor from some other **Exception** (p. 1279).

6.260 decaf::lang::exceptions::IllegalThreadStateException Class Reference 1427

Parameters

<i>ex</i>	The Exception (p. 1279) whose data is to be copied into this one.
-----------	--

6.260.1.3 **decaf::lang::exceptions::IllegalThreadStateException::IllegalThreadStateException** (const **IllegalThreadStateException** & *ex*) throw ()
[inline]

Copy Constructor.

Parameters

<i>ex</i>	The Exception (p. 1279) whose data is to be copied into this one.
-----------	--

6.260.1.4 **decaf::lang::exceptions::IllegalThreadStateException::IllegalThreadStateException** (const char * *file*, const int *lineNumber*, const char * *msg*, ...)
throw () [inline]

Constructor - Initializes the file name and line number where this message occurred.

Sets the message to report, using an optional list of arguments to parse into the message

Parameters

<i>file</i>	The file name where exception occurs
<i>lineNumber</i>	The line number where the exception occurred.
<i>msg</i>	The message to report
...	list of primitives that are formatted into the message

6.260.1.5 **decaf::lang::exceptions::IllegalThreadStateException::IllegalThreadStateException** (const char * *file*, const int *lineNumber*, const std::exception * *cause*, const char * *msg*, ...) throw () [inline]

Constructor - Initializes the file name and line number where this message occurred.

Sets the message to report, using an optional list of arguments to parse into the message

Parameters

<i>file</i>	The file name where exception occurs
<i>lineNumber</i>	The line number where the exception occurred.
<i>cause</i>	The exception that was the cause for this one to be thrown.
<i>msg</i>	The message to report
...	list of primitives that are formatted into the message

6.260.1.6 **decaf::lang::exceptions::IllegalThreadStateException::IllegalThreadStateException** (*const std::exception * cause*) throw ()
[inline]

Constructor.

Parameters

<i>cause</i>	Pointer (p. 2083) to the exception that caused this one to be thrown, the object is cloned caller retains ownership.
--------------	---

6.260.1.7 **virtual decaf::lang::exceptions::IllegalThreadStateException::~IllegalThreadStateException** () throw () [inline, virtual]

6.260.2 Member Function Documentation

6.260.2.1 **virtual IllegalThreadStateException* decaf::lang::exceptions::IllegalThreadStateException::clone** () const [inline, virtual]

Clones this exception.

This is useful for cases where you need to preserve the type of the original exception as well as the message. All subclasses should override.

Returns

an new **Exception** (p. 1279) instance that is a copy of this one.

Reimplemented from **decaf::lang::Exception** (p. 1282).

The documentation for this class was generated from the following file:

- `src/main/decaf/lang/exceptions/IllegalThreadStateException.h`

6.261 activemq::transport::inactivity::InactivityMonitor Class - Reference

```
#include <src/main/activemq/transport/inactivity/InactivityMonitor.h>
```

Inheritance diagram for `activemq::transport::inactivity::InactivityMonitor`:

Public Member Functions

- **InactivityMonitor** (const **Pointer**< **Transport** > &next, const **Pointer**< **wireformat::WireFormat** > &wireFormat)
- **InactivityMonitor** (const **Pointer**< **Transport** > &next, const **decaf::util::Properties** &properties, const **Pointer**< **wireformat::WireFormat** > &wireFormat)
- virtual **~InactivityMonitor** ()
- virtual void **close** ()
Closes this object and deallocates the appropriate resources.
- virtual void **onException** (const **decaf::lang::Exception** &ex)
Event handler for an exception from a command transport.
- virtual void **onCommand** (const **Pointer**< **Command** > &command)
Event handler for the receipt of a command.
- virtual void **oneway** (const **Pointer**< **Command** > &command)
Sends a one-way command.
- bool **isKeepAliveResponseRequired** () const
- void **setKeepAliveResponseRequired** (bool value)
- long long **getReadCheckTime** () const
- void **setReadCheckTime** (long long value)
- long long **getWriteCheckTime** () const
- void **setWriteCheckTime** (long long value)
- long long **getInitialDelayTime** () const
- void **setInitialDelayTime** (long long value) const

Friends

- class **ReadChecker**
- class **AsyncSignalReadErrorTask**
- class **WriteChecker**
- class **AsyncWriteTask**

6.261.1 Constructor & Destructor Documentation

- 6.261.1.1 **activemq::transport::inactivity::InactivityMonitor::InactivityMonitor** (const **Pointer**< **Transport** > & next, const **Pointer**< **wireformat::WireFormat** > & wireFormat)
- 6.261.1.2 **activemq::transport::inactivity::InactivityMonitor::InactivityMonitor** (const **Pointer**< **Transport** > & next, const **decaf::util::Properties** & properties, const **Pointer**< **wireformat::WireFormat** > & wireFormat)
- 6.261.1.3 **virtual activemq::transport::inactivity::InactivityMonitor::~~InactivityMonitor** () [virtual]

6.261.2 Member Function Documentation

6.261.2.1 `virtual void activemq::transport::inactivity::InactivityMonitor::close ()`
`[virtual]`

Closes this object and deallocates the appropriate resources.

The object is generally no longer usable after calling close.

Exceptions

<i>IOException</i> if an error occurs while closing.
--

Reimplemented from `activemq::transport::TransportFilter` (p. 2802).

6.261.2.2 `long long activemq::transport::inactivity::InactivityMonitor::getInitialDelayTime () const`

6.261.2.3 `long long activemq::transport::inactivity::InactivityMonitor::getReadCheckTime () const`

6.261.2.4 `long long activemq::transport::inactivity::InactivityMonitor::getWriteCheckTime () const`

6.261.2.5 `bool activemq::transport::inactivity::InactivityMonitor::isKeepAliveResponseRequired () const`

6.261.2.6 `virtual void activemq::transport::inactivity::InactivityMonitor::onCommand (const Pointer< Command > & command)`
`[virtual]`

Event handler for the receipt of a command.

Parameters

<i>command</i>	- the received command object.
----------------	--------------------------------

Reimplemented from `activemq::transport::TransportFilter` (p. 2805).

6.261.2.7 `virtual void activemq::transport::inactivity::InactivityMonitor::oneway (const Pointer< Command > & command)` `[virtual]`

Sends a one-way command.

Does not wait for any response from the broker.

Parameters

<i>command</i>	The command to be sent.
----------------	-------------------------

Exceptions

<i>IOException</i>	if an exception occurs during writing of the command.
<i>Unsupported-Operation-Exception</i>	if this method is not implemented by this transport.

Reimplemented from **activemq::transport::TransportFilter** (p. 2806).

6.261.2.8 `virtual void activemq::transport::inactivity::Inactivity-Monitor::onException (const decaf::lang::Exception & ex)`
[virtual]

Event handler for an exception from a command transport.

Parameters

<i>ex</i>	The exception to handle.
-----------	--------------------------

Reimplemented from **activemq::transport::TransportFilter** (p. 2806).

6.261.2.9 `void activemq::transport::inactivity::InactivityMonitor::setInitialDelay-Time (long long value) const`

6.261.2.10 `void activemq::transport::inactivity::Inactivity-Monitor::setKeepAliveResponseRequired (bool value)`

6.261.2.11 `void activemq::transport::inactivity::InactivityMonitor::setReadCheck-Time (long long value)`

6.261.2.12 `void activemq::transport::inactivity::InactivityMonitor::setWriteCheck-Time (long long value)`

6.261.3 Friends And Related Function Documentation

6.261.3.1 `friend class AsyncSignalReadErrorTask` [friend]

6.261.3.2 `friend class AsyncWriteTask` [friend]

6.261.3.3 `friend class ReadChecker` [friend]

6.261.3.4 `friend class WriteChecker` [friend]

The documentation for this class was generated from the following file:

- `src/main/activemq/transport/inactivity/InactivityMonitor.h`

6.262 decaf::lang::exceptions::IndexOutOfBoundsException Class Reference

```
#include <src/main/decaf/lang/exceptions/IndexOutOfBoundsException.h>
```

Inheritance diagram for decaf::lang::exceptions::IndexOutOfBoundsException:

Public Member Functions

- **IndexOutOfBoundsException** () throw ()
Default Constructor.
- **IndexOutOfBoundsException** (const **Exception** &ex) throw ()
*Conversion Constructor from some other **Exception** (p. 1279).*
- **IndexOutOfBoundsException** (const **IndexOutOfBoundsException** &ex) throw ()
Copy Constructor.
- **IndexOutOfBoundsException** (const char *file, const int lineNumber, const char *msg,...) throw ()
Constructor - Initializes the file name and line number where this message occurred.
- **IndexOutOfBoundsException** (const char *file, const int lineNumber, const std::exception *cause, const char *msg,...) throw ()
Constructor - Initializes the file name and line number where this message occurred.
- **IndexOutOfBoundsException** (const std::exception *cause) throw ()
Constructor.
- virtual **IndexOutOfBoundsException** * clone () const
Clones this exception.
- virtual ~**IndexOutOfBoundsException** () throw ()

6.262.1 Constructor & Destructor Documentation

6.262.1.1 decaf::lang::exceptions::IndexOutOfBoundsException::IndexOutOfBoundsException () throw ()
[inline]

Default Constructor.

6.262.1.2 decaf::lang::exceptions::IndexOutOfBoundsException::~IndexOutOfBoundsException (const Exception & ex) throw ()
[inline]

Conversion Constructor from some other **Exception** (p. 1279).

6.262 decaf::lang::exceptions::IndexOutOfBoundsException Class Reference 433

Parameters

<i>ex</i>	The Exception (p. 1279) whose data is to be copied into this one.
-----------	--

6.262.1.3 **decaf::lang::exceptions::IndexOutOfBoundsException::IndexOutOfBoundsException** (**const** **IndexOutOfBoundsException** & *ex*) **throw** ()
[*inline*]

Copy Constructor.

Parameters

<i>ex</i>	The Exception (p. 1279) whose data is to be copied into this one.
-----------	--

6.262.1.4 **decaf::lang::exceptions::IndexOutOfBoundsException::IndexOutOfBoundsException** (**const** **char** * *file*, **const** **int** *lineNumber*, **const** **char** * *msg*, ...) **throw** () [*inline*]

Constructor - Initializes the file name and line number where this message occurred.

Sets the message to report, using an optional list of arguments to parse into the message

Parameters

<i>file</i>	The file name where exception occurs
<i>lineNumber</i>	The line number where the exception occurred.
<i>msg</i>	The message to report
...	list of primitives that are formatted into the message

6.262.1.5 **decaf::lang::exceptions::IndexOutOfBoundsException::IndexOutOfBoundsException** (**const** **char** * *file*, **const** **int** *lineNumber*, **const** **std::exception** * *cause*, **const** **char** * *msg*, ...) **throw** () [*inline*]

Constructor - Initializes the file name and line number where this message occurred.

Sets the message to report, using an optional list of arguments to parse into the message

Parameters

<i>file</i>	The file name where exception occurs
<i>lineNumber</i>	The line number where the exception occurred.
<i>cause</i>	The exception that was the cause for this one to be thrown.
<i>msg</i>	The message to report
...	list of primitives that are formatted into the message

6.262.1.6 `decaf::lang::exceptions::IndexOutOfBoundsException::IndexOutOfBoundsException (const std::exception * cause) throw ()`
`[inline]`

Constructor.

Parameters

<i>cause</i>	Pointer (p.2083) to the exception that caused this one to be thrown, the object is cloned caller retains ownership.
--------------	--

6.262.1.7 `virtual decaf::lang::exceptions::IndexOutOfBoundsException::~~IndexOutOfBoundsException () throw ()` `[inline, virtual]`

6.262.2 Member Function Documentation

6.262.2.1 `virtual IndexOutOfBoundsException* decaf::lang::exceptions::IndexOutOfBoundsException::clone () const` `[inline, virtual]`

Clones this exception.

This is useful for cases where you need to preserve the type of the original exception as well as the message. All subclasses should override.

Returns

an new **Exception** (p. 1279) instance that is a copy of this one.

Reimplemented from **decaf::lang::Exception** (p. 1282).

The documentation for this class was generated from the following file:

- `src/main/decaf/lang/exceptions/IndexOutOfBoundsException.h`

6.263 decaf::net::Inet4Address Class Reference

```
#include <src/main/decaf/net/Inet4Address.h>
```

Inheritance diagram for `decaf::net::Inet4Address`:

Public Member Functions

- `virtual ~Inet4Address ()`

- virtual **InetAddress** * **clone** () const
*Returns a newly allocated copy of this **InetAddress** (p. 1437).*
- virtual bool **isAnyLocalAddress** () const
*Check if this **InetAddress** (p. 1437) is a valid wildcard address.*
- virtual bool **isLoopbackAddress** () const
*Check if this **InetAddress** (p. 1437) is a valid loopback address.*
- virtual bool **isMulticastAddress** () const
*Check if this **InetAddress** (p. 1437) is a valid Multicast address.*
- virtual bool **isLinkLocalAddress** () const
*Check if this **InetAddress** (p. 1437) is a valid link local address.*
- virtual bool **isSiteLocalAddress** () const
*Check if this **InetAddress** (p. 1437) is a valid site local address.*
- virtual bool **isMCGlobal** () const
*Check if this **InetAddress** (p. 1437) is Multicast and has Global scope.*
- virtual bool **isMCNodeLocal** () const
*Check if this **InetAddress** (p. 1437) is Multicast and has Node Local scope.*
- virtual bool **isMCLinkLocal** () const
*Check if this **InetAddress** (p. 1437) is Multicast and has Link Local scope.*
- virtual bool **isMCSiteLocal** () const
*Check if this **InetAddress** (p. 1437) is Multicast and has Site Local scope.*
- virtual bool **isMCOrgLocal** () const
*Check if this **InetAddress** (p. 1437) is Multicast and has Organization scope.*

Protected Member Functions

- **Inet4Address** ()
- **Inet4Address** (const unsigned char *ipAddress, int numBytes)
- **Inet4Address** (const std::string &hostname, const unsigned char *ipAddress, int numBytes)

Friends

- class **InetAddress**

6.263.1 Constructor & Destructor Documentation

6.263.1.1 **decaf::net::Inet4Address::Inet4Address** () [protected]

6.263.1.2 **decaf::net::Inet4Address::Inet4Address** (const unsigned char * *ipAddress*, int *numBytes*) [protected]

6.263.1.3 **decaf::net::Inet4Address::Inet4Address** (const std::string & *hostname*, const unsigned char * *ipAddress*, int *numBytes*) [protected]

6.263.1.4 `virtual decaf::net::Inet4Address::~~Inet4Address () [virtual]`

6.263.2 Member Function Documentation

6.263.2.1 `virtual InetAddress* decaf::net::Inet4Address::clone () const [virtual]`

Returns a newly allocated copy of this **InetAddress** (p. 1437).

The caller owns the resulting copy and must delete it.

Returns

a new **InetAddress** (p. 1437) instance that is a copy of this one, caller owns.

Reimplemented from **decaf::net::InetAddress** (p. 1439).

6.263.2.2 `virtual bool decaf::net::Inet4Address::isAnyLocalAddress () const [virtual]`

Check if this **InetAddress** (p. 1437) is a valid wildcard address.

Returns

true if the address is a wildcard address.

Reimplemented from **decaf::net::InetAddress** (p. 1442).

6.263.2.3 `virtual bool decaf::net::Inet4Address::isLinkLocalAddress () const [virtual]`

Check if this **InetAddress** (p. 1437) is a valid link local address.

Returns

true if the address is a link local address.

Reimplemented from **decaf::net::InetAddress** (p. 1442).

6.263.2.4 `virtual bool decaf::net::Inet4Address::isLoopbackAddress () const [virtual]`

Check if this **InetAddress** (p. 1437) is a valid loopback address.

Returns

true if the address is a loopback address.

Reimplemented from **decaf::net::InetAddress** (p. 1442).

6.263.2.5 `virtual bool decaf::net::Inet4Address::isMCGlobal () const [virtual]`

Check if this **InetAddress** (p. 1437) is Multicast and has Global scope.

Returns

true if the address is Multicast and has Global scope.

Reimplemented from **decaf::net::InetAddress** (p. 1442).

6.263.2.6 `virtual bool decaf::net::Inet4Address::isMCLinkLocal () const [virtual]`

Check if this **InetAddress** (p. 1437) is Multicast and has Link Local scope.

Returns

true if the address is Multicast and has Link Local scope.

Reimplemented from **decaf::net::InetAddress** (p. 1443).

6.263.2.7 `virtual bool decaf::net::Inet4Address::isMCNodeLocal () const [virtual]`

Check if this **InetAddress** (p. 1437) is Multicast and has Node Local scope.

Returns

true if the address is Multicast and has Node Local scope.

Reimplemented from **decaf::net::InetAddress** (p. 1443).

6.263.2.8 `virtual bool decaf::net::Inet4Address::isMCOrgLocal () const [virtual]`

Check if this **InetAddress** (p. 1437) is Multicast and has Organization scope.

Returns

true if the address is Multicast and has Organization scope.

Reimplemented from **decaf::net::InetAddress** (p. 1443).

6.263.2.9 `virtual bool decaf::net::Inet4Address::isMCSiteLocal () const [virtual]`

Check if this **InetAddress** (p. 1437) is Multicast and has Site Local scope.

Returns

true if the address is Multicast and has Site Local scope.

Reimplemented from **decaf::net::InetAddress** (p. 1443).

6.263.2.10 `virtual bool decaf::net::Inet4Address::isMulticastAddress () const`
`[virtual]`

Check if this **InetAddress** (p. 1437) is a valid Multicast address.

Returns

true if the address is a Multicast address.

Reimplemented from **decaf::net::InetAddress** (p. 1444).

6.263.2.11 `virtual bool decaf::net::Inet4Address::isSiteLocalAddress () const`
`[virtual]`

Check if this **InetAddress** (p. 1437) is a valid site local address.

Returns

true if the address is a site local address.

Reimplemented from **decaf::net::InetAddress** (p. 1444).

6.263.3 Friends And Related Function Documentation

6.263.3.1 `friend class InetAddress` `[friend]`

The documentation for this class was generated from the following file:

- `src/main/decaf/net/Inet4Address.h`

6.264 decaf::net::Inet6Address Class Reference

```
#include <src/main/decaf/net/Inet6Address.h>
```

Inheritance diagram for `decaf::net::Inet6Address`:

Public Member Functions

- virtual `~Inet6Address()`
- virtual `InetAddress * clone() const`

*Returns a newly allocated copy of this **InetAddress** (p. 1437).*

Protected Member Functions

- `Inet6Address()`
- `Inet6Address(const unsigned char *ipAddress, int numBytes)`
- `Inet6Address(const std::string &hostname, const unsigned char *ipAddress, int numBytes)`

Friends

- class `InetAddress`

6.264.1 Constructor & Destructor Documentation

6.264.1.1 `decaf::net::Inet6Address::Inet6Address()` [protected]

6.264.1.2 `decaf::net::Inet6Address::Inet6Address(const unsigned char *ipAddress, int numBytes)` [protected]

6.264.1.3 `decaf::net::Inet6Address::Inet6Address(const std::string &hostname, const unsigned char *ipAddress, int numBytes)` [protected]

6.264.1.4 `virtual decaf::net::Inet6Address::~~Inet6Address()` [virtual]

6.264.2 Member Function Documentation

6.264.2.1 `virtual InetAddress* decaf::net::Inet6Address::clone() const`
[virtual]

Returns a newly allocated copy of this **InetAddress** (p. 1437).

The caller owns the resulting copy and must delete it.

Returns

a new **InetAddress** (p. 1437) instance that is a copy of this one, caller owns.

Reimplemented from **decaf::net::InetAddress** (p. 1439).

6.264.3 Friends And Related Function Documentation

6.264.3.1 friend class `InetAddress` [`friend`]

The documentation for this class was generated from the following file:

- `src/main/decaf/net/Inet6Address.h`

6.265 `decaf::net::InetAddress` Class Reference

Represents an IP address.

```
#include <src/main/decaf/net/InetAddress.h>
```

Inheritance diagram for `decaf::net::InetAddress`:

Public Member Functions

- virtual `~InetAddress ()`
- virtual `decaf::lang::ArrayPointer < unsigned char > getAddress () const`
Returns the Raw IP address in Network byte order.
- virtual `std::string getHostAddress () const`
Returns a textual representation of the IP Address.
- virtual `std::string getHostName () const`
*Get the host name associated with this **InetAddress** (p. 1437) instance.*
- virtual `std::string toString () const`
*Returns a string representation of the **InetAddress** (p. 1437) in the form 'hostname / ipaddress'.*
- virtual `InetAddress * clone () const`
*Returns a newly allocated copy of this **InetAddress** (p. 1437).*
- virtual `bool isAnyLocalAddress () const`
*Check if this **InetAddress** (p. 1437) is a valid wildcard address.*
- virtual `bool isLoopbackAddress () const`
*Check if this **InetAddress** (p. 1437) is a valid loopback address.*
- virtual `bool isMulticastAddress () const`
*Check if this **InetAddress** (p. 1437) is a valid Multicast address.*
- virtual `bool isLinkLocalAddress () const`
*Check if this **InetAddress** (p. 1437) is a valid link local address.*
- virtual `bool isSiteLocalAddress () const`
*Check if this **InetAddress** (p. 1437) is a valid site local address.*
- virtual `bool isMCGlobal () const`
*Check if this **InetAddress** (p. 1437) is Multicast and has Global scope.*

- virtual bool **isMCNodeLocal** () const
*Check if this **InetAddress** (p. 1437) is Multicast and has Node Local scope.*
- virtual bool **isMCLinkLocal** () const
*Check if this **InetAddress** (p. 1437) is Multicast and has Link Local scope.*
- virtual bool **isMCSiteLocal** () const
*Check if this **InetAddress** (p. 1437) is Multicast and has Site Local scope.*
- virtual bool **isMCOrgLocal** () const
*Check if this **InetAddress** (p. 1437) is Multicast and has Organization scope.*

Static Public Member Functions

- static **InetAddress** **getByAddress** (const unsigned char *bytes, int numBytes)
*Given a raw IP Address in byte array form, create and return a new **InetAddress** (p. 1437) instance.*
- static **InetAddress** **getByAddress** (const std::string &hostname, const unsigned char *bytes, int numBytes)
*Given a host name and IPAddress return a new **InetAddress** (p. 1437).*
- static **InetAddress** **getLocalHost** ()
*Gets an **InetAddress** (p. 1437) that is the local host address.*

Protected Member Functions

- **InetAddress** ()
- **InetAddress** (const unsigned char *ipAddress, int numBytes)
- **InetAddress** (const std::string &hostname, const unsigned char *ipAddress, int numBytes)

Static Protected Member Functions

- static unsigned int **bytesToInt** (const unsigned char *bytes, int start)
Converts the bytes in an address array to an int starting from the value start treating the start value as the high order byte.
- static **InetAddress** **getAnyAddress** ()
- static **InetAddress** **getLoopbackAddress** ()

Protected Attributes

- std::string **hostname**
- bool **reached**
- decaf::lang::ArrayPointer < unsigned char > **addressBytes**

Static Protected Attributes

- static const unsigned char **loopbackBytes** [4]
- static const unsigned char **anyBytes** [4]

6.265.1 Detailed Description

Represents an IP address.

Since

1.0

6.265.2 Constructor & Destructor Documentation

6.265.2.1 **decaf::net::InetAddress::InetAddress ()** [protected]

6.265.2.2 **decaf::net::InetAddress::InetAddress (const unsigned char * *ipAddress*, int *numBytes*)** [protected]

6.265.2.3 **decaf::net::InetAddress::InetAddress (const std::string & *hostname*, const unsigned char * *ipAddress*, int *numBytes*)** [protected]

6.265.2.4 **virtual decaf::net::InetAddress::~~InetAddress ()** [virtual]

6.265.3 Member Function Documentation

6.265.3.1 **static unsigned int decaf::net::InetAddress::bytesToInt (const unsigned char * *bytes*, int *start*)** [static, protected]

Converts the bytes in an address array to an int starting from the value start treating the start value as the high order byte.

Parameters

<i>bytes</i>	The array of bytes to convert to an int.
<i>start</i>	The index in the array to treat as the high order byte.

Returns

an unsigned int that represents the address value.

6.265.3.2 **virtual InetAddress* decaf::net::InetAddress::clone ()** const [virtual]

Returns a newly allocated copy of this **InetAddress** (p. 1437).

The caller owns the resulting copy and must delete it.

Returns

a new **InetAddress** (p. 1437) instance that is a copy of this one, caller owns.

Reimplemented in **decaf::net::Inet4Address** (p. 1433), and **decaf::net::Inet6Address** (p. 1436).

6.265.3.3 **virtual decaf::lang::ArrayPointer<unsigned char>**
decaf::net::InetAddress::getAddress () const [virtual]

Returns the Raw IP address in Network byte order.

The returned address is a copy of the bytes contained in this **InetAddress** (p. 1437).

Returns

and ArrayPointer containing the raw bytes of the network address.

6.265.3.4 **static InetAddress decaf::net::InetAddress::getAnyAddress ()**
[static, protected]

6.265.3.5 **static InetAddress decaf::net::InetAddress::getByAddress (const**
unsigned char * bytes, int numBytes) [static]

Given a raw IP Address in byte array form, create and return a new **InetAddress** (p. 1437) instance.

An IPV4 address must be only four bytes in length and an IPV6 address must be 16 bytes in length.

Returns

a copy of an **InetAddress** (p. 1437) that represents the given byte array address.

Exceptions

UnknownHostException (p. 2816)	if the address array length is invalid.
--	---

6.265.3.6 **static InetAddress decaf::net::InetAddress::getByAddress (const**
std::string & hostname, const unsigned char * bytes, int numBytes) [static]

Given a host name and IP Address return a new **InetAddress** (p. 1437).

There is no name service checking or address validation done on the provided host name. The host name can either be machine name or the text based representation of the IP Address.

An IPV4 address must be only four bytes in length and an IPV6 address must be 16 bytes in length.

Returns

a copy of an **InetAddress** (p. 1437) that represents the given byte array address.

Exceptions

UnknownHostException (p. 2816)	if the address array length is invalid.
--	---

6.265.3.7 `virtual std::string decaf::net::InetAddress::getHostAddress () const`
[virtual]

Returns a textual representation of the IP Address.

Returns

the string form of the IP Address.

6.265.3.8 `virtual std::string decaf::net::InetAddress::getHostName () const`
[virtual]

Get the host name associated with this **InetAddress** (p. 1437) instance.

If a host name was set upon construction then that value is returned, otherwise a reverse name lookup will be performed to attempt to get the host name associated with the set IP Address. If the host name cannot be resolved the textual representation of the IP Address is returned instead.

Returns

the name of the host associated with this set IP Address.

6.265.3.9 `static InetAddress decaf::net::InetAddress::getLocalHost ()`
[static]

Gets an **InetAddress** (p. 1437) that is the local host address.

If the localhost value cannot be resolved then the **InetAddress** (p. 1437) for Loopback is returned.

Returns

a new **InetAddress** (p. 1437) object that contains the local host address.

Exceptions

<i>UnknownHostException</i> (p. 2816)	if the address for local host is not found.
---	---

6.265.3.10 **static InetAddress decaf::net::InetAddress::getLoopbackAddress ()**
[static, protected]

6.265.3.11 **virtual bool decaf::net::InetAddress::isAnyLocalAddress () const**
[inline, virtual]

Check if this **InetAddress** (p. 1437) is a valid wildcard address.

Returns

true if the address is a wildcard address.

Reimplemented in **decaf::net::Inet4Address** (p. 1433).

6.265.3.12 **virtual bool decaf::net::InetAddress::isLinkLocalAddress () const**
[inline, virtual]

Check if this **InetAddress** (p. 1437) is a valid link local address.

Returns

true if the address is a link local address.

Reimplemented in **decaf::net::Inet4Address** (p. 1433).

6.265.3.13 **virtual bool decaf::net::InetAddress::isLoopbackAddress () const**
[inline, virtual]

Check if this **InetAddress** (p. 1437) is a valid loopback address.

Returns

true if the address is a loopback address.

Reimplemented in **decaf::net::Inet4Address** (p. 1433).

6.265.3.14 **virtual bool decaf::net::InetAddress::isMCGlobal () const** [inline, virtual]

Check if this **InetAddress** (p. 1437) is Multicast and has Global scope.

Returns

true if the address is Multicast and has Global scope.

Reimplemented in **decaf::net::Inet4Address** (p. 1434).

6.265.3.15 `virtual bool decaf::net::InetAddress::isMCLinkLocal () const`
`[inline, virtual]`

Check if this **InetAddress** (p. 1437) is Multicast and has Link Local scope.

Returns

true if the address is Multicast and has Link Local scope.

Reimplemented in **decaf::net::Inet4Address** (p. 1434).

6.265.3.16 `virtual bool decaf::net::InetAddress::isMCNodeLocal () const`
`[inline, virtual]`

Check if this **InetAddress** (p. 1437) is Multicast and has Node Local scope.

Returns

true if the address is Multicast and has Node Local scope.

Reimplemented in **decaf::net::Inet4Address** (p. 1434).

6.265.3.17 `virtual bool decaf::net::InetAddress::isMCOrgLocal () const`
`[inline, virtual]`

Check if this **InetAddress** (p. 1437) is Multicast and has Organization scope.

Returns

true if the address is Multicast and has Organization scope.

Reimplemented in **decaf::net::Inet4Address** (p. 1434).

6.265.3.18 `virtual bool decaf::net::InetAddress::isMCSiteLocal () const`
`[inline, virtual]`

Check if this **InetAddress** (p. 1437) is Multicast and has Site Local scope.

Returns

true if the address is Multicast and has Site Local scope.

Reimplemented in **decaf::net::Inet4Address** (p. 1434).

6.265.3.19 `virtual bool decaf::net::InetAddress::isMulticastAddress () const`
[inline, virtual]

Check if this **InetAddress** (p. 1437) is a valid Multicast address.

Returns

true if the address is a Multicast address.

Reimplemented in **decaf::net::Inet4Address** (p. 1435).

6.265.3.20 `virtual bool decaf::net::InetAddress::isSiteLocalAddress () const`
[inline, virtual]

Check if this **InetAddress** (p. 1437) is a valid site local address.

Returns

true if the address is a site local address.

Reimplemented in **decaf::net::Inet4Address** (p. 1435).

6.265.3.21 `virtual std::string decaf::net::InetAddress::toString () const`
[virtual]

Returns a string representation of the **InetAddress** (p. 1437) in the form 'hostname / ipaddress'.

If the hostname is not resolved than it appears as empty.

Returns

string value of this **InetAddress** (p. 1437).

6.265.4 Field Documentation

6.265.4.1 `decaf::lang::ArrayPointer<unsigned char> decaf-`
`::net::InetAddress::addressBytes` [mutable,
protected]

6.265.4.2 `const unsigned char decaf::net::InetAddress::anyBytes[4]` [static,
protected]

6.265.4.3 `std::string decaf::net::InetAddress::hostname` [mutable,
protected]

6.265.4.4 `const unsigned char decaf::net::InetAddress::loopbackBytes[4]`
[static, protected]

6.265.4.5 `bool decaf::net::InetAddress::reached` [mutable, protected]

The documentation for this class was generated from the following file:

- `src/main/decaf/net/InetAddress.h`

6.266 `decaf::net::InetSocketAddress` Class Reference

```
#include <src/main/decaf/net/InetSocketAddress.h>
```

Inheritance diagram for `decaf::net::InetSocketAddress`:

Public Member Functions

- `InetSocketAddress ()`
- `virtual ~InetSocketAddress ()`

6.266.1 Constructor & Destructor Documentation

6.266.1.1 `decaf::net::InetSocketAddress::InetSocketAddress ()`

6.266.1.2 `virtual decaf::net::InetSocketAddress::~~InetSocketAddress ()` [virtual]

The documentation for this class was generated from the following file:

- `src/main/decaf/net/InetSocketAddress.h`

6.267 `inflate_state` Struct Reference

```
#include <src/main/decaf/internal/util/zip/inflate.h>
```

Data Fields

- `inflate_mode mode`
- `int last`
- `int wrap`
- `int havedict`
- `int flags`
- `unsigned dmax`
- `unsigned long check`

- unsigned long **total**
- **gz_headerp** head
- unsigned **wbits**
- unsigned **wsiz**e
- unsigned **whave**
- unsigned **wnext**
- unsigned char FAR * **window**
- unsigned long **hold**
- unsigned **bits**
- unsigned **length**
- unsigned **offset**
- unsigned **extra**
- **code** const FAR * **lencode**
- **code** const FAR * **distcode**
- unsigned **lenbits**
- unsigned **distbits**
- unsigned **ncode**
- unsigned **nlen**
- unsigned **ndist**
- unsigned **have**
- **code** FAR * **next**
- unsigned short **lens** [320]
- unsigned short **work** [288]
- **code** **codes** [ENOUGH]
- int **sane**
- int **back**
- unsigned **was**

6.267.1 Field Documentation

6.267.1.1 int inflate_state::back

6.267.1.2 unsigned inflate_state::bits

6.267.1.3 unsigned long inflate_state::check

6.267.1.4 code inflate_state::codes[ENOUGH]

6.267.1.5 unsigned inflate_state::distbits

6.267.1.6 code const FAR* inflate_state::distcode

6.267.1.7 unsigned inflate_state::dmax

6.267.1.8 unsigned inflate_state::extra

- 6.267.1.9 `int inflate_state::flags`
- 6.267.1.10 `unsigned inflate_state::have`
- 6.267.1.11 `int inflate_state::havedict`
- 6.267.1.12 `gz_headerp inflate_state::head`
- 6.267.1.13 `unsigned long inflate_state::hold`
- 6.267.1.14 `int inflate_state::last`
- 6.267.1.15 `unsigned inflate_state::lenbits`
- 6.267.1.16 `code const FAR* inflate_state::lencode`
- 6.267.1.17 `unsigned inflate_state::length`
- 6.267.1.18 `unsigned short inflate_state::lens[320]`
- 6.267.1.19 `inflate_mode inflate_state::mode`
- 6.267.1.20 `unsigned inflate_state::ncode`
- 6.267.1.21 `unsigned inflate_state::ndist`
- 6.267.1.22 `code FAR* inflate_state::next`
- 6.267.1.23 `unsigned inflate_state::nlen`
- 6.267.1.24 `unsigned inflate_state::offset`
- 6.267.1.25 `int inflate_state::sane`
- 6.267.1.26 `unsigned long inflate_state::total`
- 6.267.1.27 `unsigned inflate_state::was`
- 6.267.1.28 `unsigned inflate_state::wbits`
- 6.267.1.29 `unsigned inflate_state::whave`
- 6.267.1.30 `unsigned char FAR* inflate_state::window`
- 6.267.1.31 `unsigned inflate_state::wnext`
- 6.267.1.32 `unsigned short inflate_state::work[288]`

6.267.1.33 int inflate_state::wrap

6.267.1.34 unsigned inflate_state::wsize

The documentation for this struct was generated from the following file:

- src/main/decaf/internal/util/zip/inflate.h

6.268 decaf::util::zip::Inflater Class Reference

This class uncompresses data that was compressed using the *DEFLATE* algorithm (see specification).

```
#include <src/main/decaf/util/zip/Inflater.h>
```

Public Member Functions

- **Inflater** ()
Creates a new decompressor.
- **Inflater** (bool nowrap)
Creates a new decompressor.
- virtual ~**Inflater** ()
- void **setInput** (const unsigned char *buffer, int size, int offset, int length)
Sets input data for decompression.
- void **setInput** (const std::vector< unsigned char > &buffer, int offset, int length)
Sets input data for decompression.
- void **setInput** (const std::vector< unsigned char > &buffer)
Sets input data for decompression.
- int **getRemaining** () const
Returns the total number of bytes remaining in the input buffer.
- void **setDictionary** (const unsigned char *buffer, int size, int offset, int length)
Sets the preset dictionary to the given array of bytes.
- void **setDictionary** (const std::vector< unsigned char > &buffer, int offset, int length)
Sets the preset dictionary to the given array of bytes.
- void **setDictionary** (const std::vector< unsigned char > &buffer)
Sets the preset dictionary to the given array of bytes.
- bool **needsInput** () const
- bool **needsDictionary** () const
- void **finish** ()
When called, indicates that decompression should end with the current contents of the input buffer.
- bool **finished** () const
- int **inflate** (unsigned char *buffer, int size, int offset, int length)

Uncompresses bytes into specified buffer.

- int **inflate** (std::vector< unsigned char > &buffer, int offset, int length)

Uncompresses bytes into specified buffer.

- int **inflate** (std::vector< unsigned char > &buffer)

Uncompresses bytes into specified buffer.

- long long **getAdler** () const
- long long **getBytesRead** () const
- long long **getBytesWritten** () const
- void **reset** ()

Resets deflater so that a new set of input data can be processed.

- void **end** ()

Closes the decompressor and discards any unprocessed input.

6.268.1 Detailed Description

This class uncompresses data that was compressed using the *DEFLATE* algorithm (see *specification*).

Basically this class is part of the API to the stream based ZLIB compression library and is used as such by **InflaterInputStream** (p. 1456) and its descendants.

The typical usage of a **Inflater** (p. 1448) outside this package consists of a specific call to one of its constructors before being passed to an instance of **InflaterInputStream** (p. 1456).

See also

InflaterInputStream (p. 1456)

Deflater (p. 1180)

Since

1.0

6.268.2 Constructor & Destructor Documentation

6.268.2.1 decaf::util::zip::Inflater::Inflater ()

Creates a new decompressor.

This constructor defaults the inflater to use the ZLIB header and checksum fields.

6.268.2.2 decaf::util::zip::Inflater::Inflater (bool nowrap)

Creates a new decompressor.

If the parameter 'nowrap' is true then the ZLIB header and checksum fields will not be used. This provides compatibility with the compression format used by both GZIP and PKZIP.

Note: When using the 'nowrap' option it is also necessary to provide an extra "dummy" byte as input. This is required by the ZLIB native library in order to support certain optimizations.

6.268.2.3 virtual **decaf::util::zip::Inflater::~~Inflater** () [virtual]

6.268.3 Member Function Documentation

6.268.3.1 void **decaf::util::zip::Inflater::end** ()

Closes the decompressor and discards any unprocessed input.

This method should be called when the decompressor is no longer being used, but will also be called automatically by the destructor. Once this method is called, the behavior of the **Inflater** (p. 1448) object is undefined.

6.268.3.2 void **decaf::util::zip::Inflater::finish** ()

When called, indicates that decompression should end with the current contents of the input buffer.

6.268.3.3 bool **decaf::util::zip::Inflater::finished** () const

Returns

true if the end of the compressed data output stream has been reached.

6.268.3.4 long long **decaf::util::zip::Inflater::getAdler** () const

Returns

the ADLER-32 value of the uncompressed data.

Exceptions

<i>IllegalStateException</i>	if in the end state.
------------------------------	----------------------

6.268.3.5 long long **decaf::util::zip::Inflater::getBytesRead** () const

Returns

the total number of compressed bytes input so far.

Exceptions

<i>IllegalStateException</i>	if in the end state.
------------------------------	----------------------

6.268.3.6 long long decaf::util::zip::Inflater::getBytesWritten () const**Returns**

the total number of decompressed bytes output so far.

Exceptions

<i>IllegalStateException</i>	if in the end state.
------------------------------	----------------------

6.268.3.7 int decaf::util::zip::Inflater::getRemaining () const

Returns the total number of bytes remaining in the input buffer.

This can be used to find out what bytes still remain in the input buffer after decompression has finished.

Returns

the total number of bytes remaining in the input buffer

6.268.3.8 int decaf::util::zip::Inflater::inflate (unsigned char * *buffer*, int *size*, int *offset*, int *length*)

Uncompresses bytes into specified buffer.

Returns actual number of bytes uncompressed. A return value of 0 indicates that **needsInput()** (p. 1453) or **needsDictionary()** (p. 1453) should be called in order to determine if more input data or a preset dictionary is required. In the latter case, **getAdler()** (p. 1450) can be used to get the Adler-32 value of the dictionary required.

Parameters

<i>buffer</i>	The Buffer to write the compressed data to.
<i>size</i>	The size of the buffer passed in.
<i>offset</i>	The position in the Buffer to start writing at.
<i>length</i>	The maximum number of byte of data to write.

Exceptions

<i>NullPointerException</i>	if buffer is NULL.
<i>IllegalStateException</i>	if in the end state.
<i>IndexOutOfBoundsException</i>	if the offset + length > size of the buffer.
DataFormatException (p. 1076)	if the compressed data format is invalid.

6.268.3.9 `int decaf::util::zip::Inflater::inflate (std::vector< unsigned char > & buffer, int offset, int length)`

Uncompresses bytes into specified buffer.

Returns actual number of bytes uncompressed. A return value of 0 indicates that **needsInput()** (p. 1453) or **needsDictionary()** (p. 1453) should be called in order to determine if more input data or a preset dictionary is required. In the latter case, **getAdler()** (p. 1450) can be used to get the Adler-32 value of the dictionary required.

Parameters

<i>buffer</i>	The Buffer to write the compressed data to.
<i>offset</i>	The position in the Buffer to start writing at.
<i>length</i>	The maximum number of byte of data to write.

Exceptions

<i>IllegalStateException</i>	if in the end state.
<i>IndexOutOfBoundsException</i>	if the offset + length > size of the buffer.
DataFormatException (p. 1076)	if the compressed data format is invalid.

6.268.3.10 `int decaf::util::zip::Inflater::inflate (std::vector< unsigned char > & buffer)`

Uncompresses bytes into specified buffer.

Returns actual number of bytes uncompressed. A return value of 0 indicates that **needsInput()** (p. 1453) or **needsDictionary()** (p. 1453) should be called in order to determine if more input data or a preset dictionary is required. In the latter case, **getAdler()** (p. 1450) can be used to get the Adler-32 value of the dictionary required.

Parameters

<i>buffer</i>	The Buffer to write the compressed data to.
---------------	---

Exceptions

<i>IllegalStateException</i>	if in the end state.
DataFormatException (p. 1076)	if the compressed data format is invalid.

6.268.3.11 `bool decaf::util::zip::Inflater::needsDictionary () const`

Returns

true if a preset dictionary is needed for decompression.

6.268.3.12 `bool decaf::util::zip::Inflater::needsInput () const`

Returns

true if the input data buffer is empty and **setInput()** (p. 1455) should be called in order to provide more input

6.268.3.13 `void decaf::util::zip::Inflater::reset ()`

Resets deflater so that a new set of input data can be processed.

Keeps current decompression level and strategy settings.

Exceptions

<i>IllegalStateException</i>	if in the end state.
------------------------------	----------------------

6.268.3.14 `void decaf::util::zip::Inflater::setDictionary (const unsigned char * buffer, int size, int offset, int length)`

Sets the preset dictionary to the given array of bytes.

Should be called when **inflate()** (p. 1451) returns 0 and **needsDictionary()** (p. 1453) returns true indicating that a preset dictionary is required. The method **getAdler()** (p. 1450) can be used to get the Adler-32 value of the dictionary needed.

Parameters

<i>buffer</i>	The Buffer to read in for decompression.
<i>size</i>	The size of the buffer passed in.
<i>offset</i>	The position in the Buffer to start reading from.
<i>length</i>	The number of bytes to read from the input buffer.

Exceptions

<i>NullPointerException</i>	if buffer is NULL.
<i>IndexOutOfBoundsException</i>	if the offset + length > size of the buffer.
<i>IllegalStateException</i>	if in the end state.
<i>IllegalArgument-Exception</i>	if the given dictionary doesn't match thre required dictionaries checksum value.

6.268.3.15 void decaf::util::zip::Inflater::setDictionary (const std::vector< unsigned char > & buffer, int offset, int length)

Sets the preset dictionary to the given array of bytes.

Should be called when **inflate()** (p. 1451) returns 0 and **needsDictionary()** (p. 1453) returns true indicating that a preset dictionary is required. The method **getAdler()** (p. 1450) can be used to get the Adler-32 value of the dictionary needed.

Parameters

<i>buffer</i>	The Buffer to read in for decompression.
<i>offset</i>	The position in the Buffer to start reading from.
<i>length</i>	The number of bytes to read from the input buffer.

Exceptions

<i>IndexOutOfBoundsException</i>	if the offset + length > size of the buffer.
<i>IllegalStateException</i>	if in the end state.
<i>IllegalArgument-Exception</i>	if the given dictionary doesn't match thre required dictionaries checksum value.

6.268.3.16 void decaf::util::zip::Inflater::setDictionary (const std::vector< unsigned char > & buffer)

Sets the preset dictionary to the given array of bytes.

Should be called when **inflate()** (p. 1451) returns 0 and **needsDictionary()** (p. 1453) returns true indicating that a preset dictionary is required. The method **getAdler()** (p. 1450) can be used to get the Adler-32 value of the dictionary needed.

Parameters

<i>buffer</i>	The Buffer to read in for decompression.
---------------	--

Exceptions

<i>IllegalStateException</i>	if in the end state.
------------------------------	----------------------

<i>IllegalArgument-Exception</i>	if the given dictionary doesn't match the required dictionaries check-sum value.
----------------------------------	--

6.268.3.17 **void decaf::util::zip::Inflater::setInput (const unsigned char * *buffer*, int *size*, int *offset*, int *length*)**

Sets input data for decompression.

This should be called whenever **needsInput()** (p. 1453) returns true indicating that more input data is required.

Parameters

<i>buffer</i>	The Buffer to read in for decompression.
<i>size</i>	The size of the buffer passed in.
<i>offset</i>	The position in the Buffer to start reading from.
<i>length</i>	The number of bytes to read from the input buffer.

Exceptions

<i>NullPointerException</i>	if buffer is NULL.
<i>IndexOutOfBoundsException</i>	if the offset + length > size of the buffer.
<i>IllegalStateException</i>	if in the end state.

6.268.3.18 **void decaf::util::zip::Inflater::setInput (const std::vector< unsigned char > & *buffer*, int *offset*, int *length*)**

Sets input data for decompression.

This should be called whenever **needsInput()** (p. 1453) returns true indicating that more input data is required.

Parameters

<i>buffer</i>	The Buffer to read in for decompression.
<i>offset</i>	The position in the Buffer to start reading from.
<i>length</i>	The number of bytes to read from the input buffer.

Exceptions

<i>IndexOutOfBoundsException</i>	if the offset + length > size of the buffer.
<i>IllegalStateException</i>	if in the end state.

6.268.3.19 void **decaf::util::zip::Inflater::setInput** (const std::vector< unsigned char > & *buffer*)

Sets input data for decompression.

This should be called whenever **needsInput()** (p. 1453) returns true indicating that more input data is required.

Parameters

<i>buffer</i>	The Buffer to read in for decompression.
---------------	--

Exceptions

<i>IllegalStateException</i>	if in the end state.
------------------------------	----------------------

The documentation for this class was generated from the following file:

- src/main/decaf/util/zip/**Inflater.h**

6.269 decaf::util::zip::InflaterInputStream Class Reference

A FilterInputStream that decompresses data read from the wrapped InputStream instance.

```
#include <src/main/decaf/util/zip/InflaterInputStream.h>
```

Inheritance diagram for decaf::util::zip::InflaterInputStream:

Public Member Functions

- **InflaterInputStream** (**decaf::io::InputStream** **inputStream*, bool *own*=false)
Create an instance of this class with a default inflater and buffer size.
- **InflaterInputStream** (**decaf::io::InputStream** **inputStream*, **Inflater** **inflater*, bool *own*=false, bool *ownInflater*=false)
Creates a new InflaterInputStream (p. 1456) with a user supplied Inflater (p. 1448) and a default buffer size.
- **InflaterInputStream** (**decaf::io::InputStream** **inputStream*, **Inflater** **inflater*, int *bufferSize*, bool *own*=false, bool *ownInflater*=false)
Creates a new DeflateOutputStream with a user supplied Inflater (p. 1448) and specified buffer size.
- virtual ~**InflaterInputStream** ()
- virtual int **available** () const
*Indicates the number of bytes available.
The default implementation of this methods returns 0. Classes that override this method may return the total number of bytes that are currently available for reading*

and others may simply return a value of one indicating that there is some data available. The caller should view the result of this method as an absolute. The default implementation of this method returns zero.

Returns

the number of bytes available on this input stream.

Exceptions

IOException (p. 1545)	if an I/O error occurs.
------------------------------	-------------------------

- virtual void **close** ()

Closes the **InputStream** (p. 1464) freeing any resources that might have been acquired during the lifetime of this stream.

The default implementation of this method does nothing.

Exceptions

IOException (p. 1545)	if an I/O error occurs while closing the InputStream (p. 1464).
------------------------------	--

- virtual long long **skip** (long long num)

Skips over and discards *n* bytes of data from this input stream.

The skip method may, for a variety of reasons, end up skipping over some smaller number of bytes, possibly 0. This may result from any of a number of conditions; reaching end of file before *n* bytes have been skipped is only one possibility. The actual number of bytes skipped is returned.

The skip method of **InputStream** (p. 1464) creates a byte array and then repeatedly reads into it until *num* bytes have been read or the end of the stream has been reached. Subclasses are encouraged to provide a more efficient implementation of this method.

Parameters

num	The number of bytes to skip.
-----	------------------------------

Returns

total bytes skipped

Exceptions

IOException (p. 1545)	if an I/O error occurs.
------------------------------	-------------------------

UnsupportedOperationException	if the concrete stream class does not support skipping bytes.
-------------------------------	---

- virtual void **mark** (int readLimit)

Marks the current position in the stream. A subsequent call to the reset method repositions this stream at the last marked position so that subsequent reads re-read the same bytes.

If a stream instance reports that marks are supported then the stream will ensure that the same bytes can be read again after the reset method is called so long the readLimit is not reached.

Calling mark on a closed stream instance should have no effect.

The default implementation of this method does nothing.

Parameters

readLimit	The max bytes read before marked position is invalid.
-----------	---

- virtual void **reset** ()

Repositions this stream to the position at the time the mark method was last called on this input stream.

If the method `markSupported` returns `true`, then: * If the method `mark` has not been called since the stream was created, or the number of bytes read from the stream since `mark` was last called is larger than the argument to `mark` at that last call, then an **`IOException`** (p. 1545) might be thrown. * If such an **`IOException`** (p. 1545) is not thrown, then the stream is reset to a state such that all the bytes read since the most recent call to `mark` (or since the start of the file, if `mark` has not been called) will be resupplied to subsequent callers of the `read` method, followed by any bytes that otherwise would have been the next input data as of the time of the call to `reset`.

If the method `markSupported` returns `false`, then: * The call to `reset` may throw an **`IOException`** (p. 1545). * If an **`IOException`** (p. 1545) is not thrown, then the stream is reset to a fixed state that depends on the particular type of the input stream and how it was created. The bytes that will be supplied to subsequent callers of the `read` method depend on the particular type of the input stream.

The default implementation of this method throws an **`IOException`** (p. 1545).

Exceptions

<code>IOException</code> (p. 1545)	if an I/O error occurs.
---	-------------------------

- virtual bool **`markSupported`** () const

Determines if this input stream supports the `mark` and `reset` methods.

Whether or not `mark` and `reset` are supported is an invariant property of a particular input stream instance.

The default implementation of this method returns `false`.

Returns

`true` if this stream instance supports marks

Protected Member Functions

- virtual void **`fill`** ()
Fills the input buffer with the next chunk of data.
- virtual int **`doReadByte`** ()
- virtual int **`doReadArrayBounded`** (unsigned char *buffer, int size, int offset, int length)

Protected Attributes

- **`Inflater`** * **`inflater`**
The **`Inflater`** (p. 1448) instance to use.
- std::vector< unsigned char > **`buff`**
The buffer to hold chunks of data read from the stream before inflation.
- int **`length`**
The amount of data currently stored in the input buffer.
- bool **`ownInflater`**
- bool **`atEOF`**

Static Protected Attributes

- static const int **`DEFAULT_BUFFER_SIZE`**

6.269.1 Detailed Description

A `FilterInputStream` that decompresses data read from the wrapped `InputStream` instance.

Since

1.0

6.269.2 Constructor & Destructor Documentation

6.269.2.1 `decaf::util::zip::InflaterInputStream::InflaterInputStream (decaf::io::InputStream * inputStream, bool own = false)`

Create an instance of this class with a default inflater and buffer size.

Parameters

<i>inputStream</i>	The <code>InputStream</code> instance to wrap.
<i>own</i>	Should this <code>Filter</code> take ownership of the <code>InputStream</code> pointer (defaults to false).

6.269.2.2 `decaf::util::zip::InflaterInputStream::InflaterInputStream (decaf::io::InputStream * inputStream, Inflater * inflater, bool own = false, bool ownInflater = false)`

Creates a new **`InflaterInputStream`** (p. 1456) with a user supplied **`Inflater`** (p. 1448) and a default buffer size.

When the user supplied a **`Inflater`** (p. 1448) instance the **`InflaterInputStream`** (p. 1456) does not take ownership of the **`Inflater`** (p. 1448) pointer unless the `ownInflater` parameter is set to true, otherwise the caller is still responsible for deleting the **`Inflater`** (p. 1448).

Parameters

<i>inputStream</i>	The <code>InputStream</code> instance to wrap.
<i>inflater</i>	The user supplied <code>Inflater</code> (p. 1448) to use for decompression. (
<i>own</i>	Should this filter take ownership of the <code>InputStream</code> pointer (default is false).
<i>ownInflater</i>	Should the filter take ownership of the passed <code>Inflater</code> (p. 1448) object (default is false).

Exceptions

<i><code>NullPointerException</code></i>	if the <code>Inflater</code> (p. 1448) given is NULL.
--	--

6.269.2.3 `decaf::util::zip::InflaterInputStream::InflaterInputStream (decaf::io::InputStream * inputStream, Inflater * inflater, int bufferSize, bool own = false, bool ownInflater = false)`

Creates a new DeflateOutputStream with a user supplied **Inflater** (p. 1448) and specified buffer size.

When the user supplied a **Inflater** (p. 1448) instance the **InflaterInputStream** (p. 1456) does not take ownership of the **Inflater** (p. 1448) pointer unless the `ownInflater` parameter is set to true, otherwise the caller is still responsible for deleting the **Inflater** (p. 1448).

Parameters

<i>inputStream</i>	The InputStream instance to wrap.
<i>inflater</i>	The user supplied Inflater (p. 1448) to use for decompression.
<i>bufferSize</i>	The size of the input buffer.
<i>own</i>	Should this filter take ownership of the InputStream pointer (default is false).
<i>ownInflater</i>	Should the filter take ownership of the passed Inflater (p. 1448) object (default is false).

Exceptions

<i>NullPointerException</i>	if the Inflater (p. 1448) given is NULL.
<i>IllegalArgumentException</i>	if the bufferSize value is zero.

6.269.2.4 `virtual decaf::util::zip::InflaterInputStream::~~InflaterInputStream ()`
[virtual]

6.269.3 Member Function Documentation

6.269.3.1 `virtual int decaf::util::zip::InflaterInputStream::available () const`
[virtual]

Indicates the number of bytes available.

The default implementation of this methods returns 0. Classes that override this method may return the total number of bytes that are currently available for reading and others may simply return a value of one indicating that there is some data available. The caller should view the result of this method as an absolute.

The default implementation of this method returns zero.

Returns

the number of bytes available on this input stream.

Exceptions

<i>IOException</i> (p. 1545)	if an I/O error occurs.
--	-------------------------

Until EOF this method always returns 1, thereafter it always returns 0.

Reimplemented from **decaf::io::FilterInputStream** (p. 1337).

6.269.3.2 **virtual void decaf::util::zip::InflaterInputStream::close ()** [virtual]

Closes the **InputStream** (p. 1464) freeing any resources that might have been acquired during the lifetime of this stream.

The default implementation of this method does nothing.

Exceptions

<i>IOException</i> (p. 1545)	if an I/O error occurs while closing the InputStream (p. 1464).
--	--

Closes any resources associated with this **InflaterInputStream** (p. 1456).

Reimplemented from **decaf::io::FilterInputStream** (p. 1337).

6.269.3.3 **virtual int decaf::util::zip::InflaterInputStream::doReadArrayBounded (unsigned char * *buffer*, int *size*, int *offset*, int *length*)** [protected, virtual]

Reimplemented from **decaf::io::FilterInputStream** (p. 1338).

6.269.3.4 **virtual int decaf::util::zip::InflaterInputStream::doReadByte ()** [protected, virtual]

Reimplemented from **decaf::io::FilterInputStream** (p. 1338).

6.269.3.5 **virtual void decaf::util::zip::InflaterInputStream::fill ()** [protected, virtual]

Fills the input buffer with the next chunk of data.

Exceptions

<i>IOException</i>	if an I/O error occurs.
---------------------------	-------------------------

6.269.3.6 `virtual void decaf::util::zip::InflaterInputStream::mark (int readLimit)`
[virtual]

Marks the current position in the stream. A subsequent call to the reset method repositions this stream at the last marked position so that subsequent reads re-read the same bytes.

If a stream instance reports that marks are supported then the stream will ensure that the same bytes can be read again after the reset method is called so long the readLimit is not reached.

Calling mark on a closed stream instance should have no effect.

The default implementation of this method does nothing.

Parameters

<i>readLimit</i>	The max bytes read before marked position is invalid.
------------------	---

Does nothing.

Reimplemented from **decaf::io::FilterInputStream** (p. 1338).

6.269.3.7 `virtual bool decaf::util::zip::InflaterInputStream::markSupported () const`
[virtual]

Determines if this input stream supports the mark and reset methods.

Whether or not mark and reset are supported is an invariant property of a particular input stream instance.

The default implementation of this method returns false.

Returns

true if this stream instance supports marks

Always returns false.

Reimplemented from **decaf::io::FilterInputStream** (p. 1339).

6.269.3.8 `virtual void decaf::util::zip::InflaterInputStream::reset ()` [virtual]

Repositions this stream to the position at the time the mark method was last called on this input stream.

If the method markSupported returns true, then: * If the method mark has not been called since the stream was created, or the number of bytes read from the stream since mark was last called is larger than the argument to mark at that last call, then an **IOException** (p. 1545) might be thrown. * If such an **IOException** (p. 1545) is not thrown, then the stream is reset to a state such that all the bytes read since the most recent call to mark (or since the start of the file, if mark has not been called) will be resupplied

to subsequent callers of the read method, followed by any bytes that otherwise would have been the next input data as of the time of the call to reset.

If the method `markSupported` returns false, then: * The call to reset may throw an **IOException** (p. 1545). * If an **IOException** (p. 1545) is not thrown, then the stream is reset to a fixed state that depends on the particular type of the input stream and how it was created. The bytes that will be supplied to subsequent callers of the read method depend on the particular type of the input stream.

The default implementation of this method throws an **IOException** (p. 1545).

Exceptions

IOException (p. 1545)	if an I/O error occurs.
---------------------------------	-------------------------

Always throws an **IOException** when called.

Reimplemented from **decaf::io::FilterInputStream** (p. 1339).

6.269.3.9 virtual long long **decaf::util::zip::InflaterInputStream::skip** (long long *num*)
[virtual]

Skips over and discards *n* bytes of data from this input stream.

The skip method may, for a variety of reasons, end up skipping over some smaller number of bytes, possibly 0. This may result from any of a number of conditions; reaching end of file before *n* bytes have been skipped is only one possibility. The actual number of bytes skipped is returned.

The skip method of **InputStream** (p. 1464) creates a byte array and then repeatedly reads into it until *num* bytes have been read or the end of the stream has been reached. Subclasses are encouraged to provide a more efficient implementation of this method.

Parameters

<i>num</i>	The number of bytes to skip.
------------	------------------------------

Returns

total bytes skipped

Exceptions

IOException (p. 1545)	if an I/O error occurs.
UnsupportedOperationException	if the concrete stream class does not support skipping bytes.

Skips the specified amount of uncompressed input data.

Reimplemented from `decaf::io::FilterInputStream` (p. 1340).

6.269.4 Field Documentation

6.269.4.1 `bool decaf::util::zip::InflaterInputStream::atEOF` [protected]

6.269.4.2 `std::vector<unsigned char> decaf::util::zip::InflaterInputStream::buff`
[protected]

The buffer to hold chunks of data read from the stream before inflation.

6.269.4.3 `const int decaf::util::zip::InflaterInputStream::DEFAULT_BUFFER_SIZE`
[static, protected]

6.269.4.4 `Inflater* decaf::util::zip::InflaterInputStream::inflater` [protected]

The `Inflater` (p. 1448) instance to use.

6.269.4.5 `int decaf::util::zip::InflaterInputStream::length` [protected]

The amount of data currently stored in the input buffer.

6.269.4.6 `bool decaf::util::zip::InflaterInputStream::ownInflater` [protected]

The documentation for this class was generated from the following file:

- `src/main/decaf/util/zip/InflaterInputStream.h`

6.270 `decaf::io::InputStream` Class Reference

A base class that must be implemented by all classes wishing to provide a class that reads in a stream of bytes.

```
#include <src/main/decaf/io/InputStream.h>
```

Inheritance diagram for `decaf::io::InputStream`:

Public Member Functions

- `InputStream ()`
- `virtual ~InputStream ()`
- `virtual void close ()`

*Closes the **InputStream** (p. 1464) freeing any resources that might have been acquired during the lifetime of this stream.*

- virtual void **mark** (int readLimit)

Marks the current position in the stream. A subsequent call to the reset method repositions this stream at the last marked position so that subsequent reads re-read the same bytes.

- virtual void **reset** ()

Repositions this stream to the position at the time the mark method was last called on this input stream.

- virtual bool **markSupported** () const

Determines if this input stream supports the mark and reset methods.

- virtual int **available** () const

Indicates the number of bytes available.

- virtual int **read** ()

Reads a single byte from the buffer.

- virtual int **read** (unsigned char *buffer, int size)

Reads up to size bytes of data from the input stream into an array of bytes.

- virtual int **read** (unsigned char *buffer, int size, int offset, int length)

Reads up to length bytes of data from the input stream into an array of bytes.

- virtual long long **skip** (long long num)

Skips over and discards n bytes of data from this input stream.

- virtual std::string **toString** () const

Output a String representation of this object.

- virtual void **lock** ()

Locks the object.

- virtual bool **tryLock** ()

Attempts to Lock the object, if the lock is already held by another thread than this method returns false.

- virtual void **unlock** ()

Unlocks the object.

- virtual void **wait** ()

Waits on a signal from this object, which is generated by a call to Notify.

- virtual void **wait** (long long millisecs)

Waits on a signal from this object, which is generated by a call to Notify.

- virtual void **wait** (long long millisecs, int nanos)

Waits on a signal from this object, which is generated by a call to Notify.

- virtual void **notify** ()

Signals a waiter on this object that it can now wake up and continue.

- virtual void **notifyAll** ()

Signals the waiters on this object that it can now wake up and continue.

Protected Member Functions

- virtual int **doReadByte** ()=0
- virtual int **doReadArray** (unsigned char *buffer, int size)
- virtual int **doReadArrayBounded** (unsigned char *buffer, int size, int offset, int length)

6.270.1 Detailed Description

A base class that must be implemented by all classes wishing to provide a class that reads in a stream of bytes.

Since

1.0

6.270.2 Constructor & Destructor Documentation

6.270.2.1 **decaf::io::InputStream::InputStream** ()

6.270.2.2 **virtual decaf::io::InputStream::~~InputStream** () [virtual]

6.270.3 Member Function Documentation

6.270.3.1 **virtual int decaf::io::InputStream::available** () const [inline, virtual]

Indicates the number of bytes available.

The default implementation of this methods returns 0. Classes that override this method may return the total number of bytes that are currently available for reading and others may simply return a value of one indicating that there is some data available. The caller should view the result of this method as an absolute.

The default implementation of this method returns zero.

Returns

the number of bytes available on this input stream.

Exceptions

<i>IOException</i> (p. 1545)	if an I/O error occurs.
--	-------------------------

Reimplemented in **decaf::io::ByteArrayInputStream** (p. 683), **decaf::io::PushbackInputStream** (p. 2217), **decaf::util::zip::InflaterInputStream** (p. 1460), **decaf::io::BufferedInputStream** (p. 592), **decaf::io::BlockingByteArrayInputStream** (p. 536),

decaf::io::FilterInputStream (p. 1337), **decaf::internal::net::tcp::TcpSocketInputStream** (p. 2689), **decaf::internal::net::ssl::openssl::OpenSSLSocketInputStream** (p. 2041), and **decaf::internal::io::StandardInputStream** (p. 2531).

6.270.3.2 virtual void decaf::io::InputStream::close () [virtual]

Closes the **InputStream** (p. 1464) freeing any resources that might have been acquired during the lifetime of this stream.

The default implementation of this method does nothing.

Exceptions

<i>IOException</i> (p. 1545)	if an I/O error occurs while closing the InputStream (p. 1464).
--	--

Implements **decaf::io::Closeable** (p. 817).

Reimplemented in **decaf::util::zip::InflaterInputStream** (p. 1461), **decaf::io::BufferedInputStream** (p. 592), **decaf::io::BlockingByteArrayInputStream** (p. 537), **decaf::io::FilterInputStream** (p. 1337), **decaf::internal::net::tcp::TcpSocketInputStream** (p. 2690), and **decaf::internal::net::ssl::openssl::OpenSSLSocketInputStream** (p. 2042).

6.270.3.3 virtual int decaf::io::InputStream::doReadArray (unsigned char * buffer, int size) [protected, virtual]

Reimplemented in **decaf::io::FilterInputStream** (p. 1337).

6.270.3.4 virtual int decaf::io::InputStream::doReadArrayBounded (unsigned char * buffer, int size, int offset, int length) [protected, virtual]

Reimplemented in **decaf::io::ByteArrayInputStream** (p. 684), **decaf::util::zip::InflaterInputStream** (p. 1461), **decaf::io::PushbackInputStream** (p. 2217), **decaf::io::BufferedInputStream** (p. 593), **decaf::io::FilterInputStream** (p. 1338), **decaf::io::BlockingByteArrayInputStream** (p. 537), **decaf::internal::net::tcp::TcpSocketInputStream** (p. 2690), **decaf::util::zip::CheckedInputStream** (p. 807), **decaf::internal::net::ssl::openssl::OpenSSLSocketInputStream** (p. 2042), and **activemq::io::LoggingInputStream** (p. 1707).

6.270.3.5 virtual int decaf::io::InputStream::doReadByte () [protected, pure virtual]

Implemented in **decaf::io::ByteArrayInputStream** (p. 684), **decaf::util::zip::InflaterInputStream** (p. 1461), **decaf::io::PushbackInputStream** (p. 2218), **decaf::io::BufferedInputStream** (p. 593), **decaf::io::FilterInputStream** (p. 1338), **decaf::io::BlockingByteArrayInputStream** (p. 537), **decaf::internal::net::tcp::TcpSocketInputStream** (p. 2690), **decaf::util::zip::CheckedInputStream** (p. 807), **decaf::**

decaf::internal::net::ssl::openssl::OpenSSLSocketInputStream (p. 2042), **activemq::io::LoggingInputStream** (p. 1707), and **decaf::internal::io::StandardInputStream** (p. 2531).

6.270.3.6 virtual void **decaf::io::InputStream::lock** () [inline, virtual]

Locks the object.

Exceptions

<i>RuntimeException</i>	if an error occurs while locking the object.
-------------------------	--

Implements **decaf::util::concurrent::Synchronizable** (p. 2640).

6.270.3.7 virtual void **decaf::io::InputStream::mark** (int *readLimit*) [virtual]

Marks the current position in the stream. A subsequent call to the reset method repositions this stream at the last marked position so that subsequent reads re-read the same bytes.

If a stream instance reports that marks are supported then the stream will ensure that the same bytes can be read again after the reset method is called so long the readLimit is not reached.

Calling mark on a closed stream instance should have no effect.

The default implementation of this method does nothing.

Parameters

<i>readLimit</i>	The max bytes read before marked position is invalid.
------------------	---

Reimplemented in **decaf::io::ByteArrayInputStream** (p. 684), **decaf::io::PushbackInputStream** (p. 2218), **decaf::util::zip::InflaterInputStream** (p. 1462), **decaf::io::BufferedInputStream** (p. 593), and **decaf::io::FilterInputStream** (p. 1338).

6.270.3.8 virtual bool **decaf::io::InputStream::markSupported** () const [inline, virtual]

Determines if this input stream supports the mark and reset methods.

Whether or not mark and reset are supported is an invariant property of a particular input stream instance.

The default implementation of this method returns false.

Returns

true if this stream instance supports marks

Reimplemented in **decaf::io::ByteArrayInputStream** (p. 685), **decaf::io::PushbackInputStream** (p. 2218), **decaf::util::zip::InflaterInputStream** (p. 1462), **decaf::io::BufferedInputStream** (p. 593), and **decaf::io::FilterInputStream** (p. 1339).

6.270.3.9 `virtual void decaf::io::InputStream::notify () [inline, virtual]`

Signals a waiter on this object that it can now wake up and continue.

Must have this object locked before calling.

Exceptions

<i>IllegalMonitorStateException</i>	- if the current thread is not the owner of the the Synchronizable Object.
<i>RuntimeException</i>	if an error occurs while notifying one of the waiting threads.

Implements **decaf::util::concurrent::Synchronizable** (p. 2641).

6.270.3.10 `virtual void decaf::io::InputStream::notifyAll () [inline, virtual]`

Signals the waiters on this object that it can now wake up and continue.

Must have this object locked before calling.

Exceptions

<i>IllegalMonitorStateException</i>	- if the current thread is not the owner of the the Synchronizable Object.
<i>RuntimeException</i>	if an error occurs while notifying the waiting threads.

Implements **decaf::util::concurrent::Synchronizable** (p. 2642).

6.270.3.11 `virtual int decaf::io::InputStream::read () [virtual]`

Reads a single byte from the buffer.

The value byte is returned as an int in the range 0 to 255. If no byte is available because the end of the stream has been reached, the value -1 is returned. This method blocks until input data is available, the end of the stream is detected, or an exception is thrown.

The default implementation of this method calls the internal virtual method `doReadByte` which is a pure virtual method and must be overridden by all subclasses.

Returns

The next byte or -1 if the end of stream is reached.

Exceptions

<i>IOException</i> (p. 1545)	if an I/O error occurs.
--	-------------------------

6.270.3.12 `virtual int decaf::io::InputStream::read (unsigned char * buffer, int size)`
[virtual]

Reads up to size bytes of data from the input stream into an array of bytes.

An attempt is made to read as many as size bytes, but a smaller number may be read. The number of bytes actually read is returned as an integer.

This method blocks until input data is available, end of file is detected, or an exception is thrown.

If size is zero, then no bytes are read and 0 is returned; otherwise, there is an attempt to read at least one byte. If no byte is available because the stream is at end of file, the value -1 is returned; otherwise, at least one byte is read and stored into b.

This method called the protected virtual method doReadArray which by default is the same as calling read(buffer, size, 0, size). Subclasses can customize the behavior of this method by overriding the doReadArray method to provide a better performing read operation.

Parameters

<i>buffer</i>	The target buffer to write the read in data to.
<i>size</i>	The size in bytes of the target buffer.

Returns

The number of bytes read or -1 if EOF is detected

Exceptions

<i>IOException</i> (p. 1545)	if an I/O error occurs.
<i>NullPointerException</i>	if buffer passed is NULL.

6.270.3.13 `virtual int decaf::io::InputStream::read (unsigned char * buffer, int size, int offset, int length)` [virtual]

Reads up to length bytes of data from the input stream into an array of bytes.

An attempt is made to read as many as length bytes, but a smaller number may be read. The number of bytes actually read is returned as an integer.

This method blocks until input data is available, end of file is detected, or an exception is thrown.

If length is zero, then no bytes are read and 0 is returned; otherwise, there is an attempt to read at least one byte. If no byte is available because the stream is at end of file, the value -1 is returned; otherwise, at least one byte is read and stored into b.

The first byte read is stored into element b[offset], the next one into b[offset+1], and so on. The number of bytes read is, at most, equal to length. Let k be the number of bytes actually read; these bytes will be stored in elements b[offset] through b[offset+k-1], leaving elements b[offset+k] through b[offset+length-1] unaffected.

In every case, elements b[0] through b[offset] and elements b[offset+length] through b[b.length-1] are unaffected.

This method called the protected virtual method doReadArrayBounded which simply calls the method **doReadByte()** (p. 1467) repeatedly. If the first such call results in an **IOException** (p. 1545), that exception is returned. If any subsequent call to **doReadByte()** (p. 1467) results in a **IOException** (p. 1545), the exception is caught and treated as if it were end of file; the bytes read up to that point are stored into the buffer and the number of bytes read before the exception occurred is returned. The default implementation of this method blocks until the requested amount of input data has been read, end of file is detected, or an exception is thrown. Subclasses are encouraged to provide a more efficient implementation of this method.

Parameters

<i>buffer</i>	The target buffer to write the read in data to.
<i>size</i>	The size in bytes of the target buffer.
<i>offset</i>	The position in the buffer to start inserting the read in data.
<i>length</i>	The maximum number of bytes that should be read into buffer.

Returns

The number of bytes read or -1 if EOF is detected

Exceptions

IOException (p. 1545)	if an I/O error occurs.
<i>NullPointerException</i>	if buffer passed is NULL.
<i>IndexOutOfBoundsException</i>	if length > size - offset.

6.270.3.14 virtual void decaf::io::InputStream::reset () [virtual]

Repositions this stream to the position at the time the mark method was last called on this input stream.

If the method markSupported returns true, then: * If the method mark has not been called since the stream was created, or the number of bytes read from the stream since mark was last called is larger than the argument to mark at that last call, then an **IOException** (p. 1545) might be thrown. * If such an **IOException** (p. 1545) is not thrown, then the stream is reset to a state such that all the bytes read since the most recent call to mark (or since the start of the file, if mark has not been called) will be resupplied to subsequent callers of the read method, followed by any bytes that otherwise would have been the next input data as of the time of the call to reset.

If the method markSupported returns false, then: * The call to reset may throw an **IOException** (p. 1545). * If an **IOException** (p. 1545) is not thrown, then the stream is reset to a fixed state that depends on the particular type of the input stream and how it was created. The bytes that will be supplied to subsequent callers of the read method depend on the particular type of the input stream.

The default implementation of this method throws an **IOException** (p. 1545).

Exceptions

IOException (p. 1545)	if an I/O error occurs.
---------------------------------	-------------------------

Reimplemented in **decaf::io::ByteArrayInputStream** (p. 685), **decaf::io::PushbackInputStream** (p. 2218), **decaf::util::zip::InflaterInputStream** (p. 1462), **decaf::io::BufferedInputStream** (p. 594), and **decaf::io::FilterInputStream** (p. 1339).

6.270.3.15 virtual long long decaf::io::InputStream::skip (long long num) [virtual]

Skips over and discards n bytes of data from this input stream.

The skip method may, for a variety of reasons, end up skipping over some smaller number of bytes, possibly 0. This may result from any of a number of conditions; reaching end of file before n bytes have been skipped is only one possibility. The actual number of bytes skipped is returned.

The skip method of **InputStream** (p. 1464) creates a byte array and then repeatedly reads into it until num bytes have been read or the end of the stream has been reached. Subclasses are encouraged to provide a more efficient implementation of this method.

Parameters

num	The number of bytes to skip.
-----	------------------------------

Returns

total bytes skipped

Exceptions

<i>IOException</i> (p. 1545)	if an I/O error occurs.
<i>Unsupported-Operation</i> <i>Exception</i>	if the concrete stream class does not support skipping bytes.

Reimplemented in **decaf::io::ByteArrayInputStream** (p. 687), **decaf::io::PushbackInputStream** (p. 2219), **decaf::util::zip::InflaterInputStream** (p. 1463), **decaf::io::BufferedInputStream** (p. 594), **decaf::io::BlockingByteArrayInputStream** (p. 537), **decaf::internal::net::tcp::TcpSocketInputStream** (p. 2690), **decaf::io::FilterInputStream** (p. 1340), **decaf::util::zip::CheckedInputStream** (p. 807), and **decaf::internal::net::ssl::openssl::OpenSSLSocketInputStream** (p. 2042).

6.270.3.16 `virtual std::string decaf::io::InputStream::toString () const [virtual]`

Output a String representation of this object.

The default version of this method just prints the Class Name.

Returns

a string representation of the object.

6.270.3.17 `virtual bool decaf::io::InputStream::tryLock () [inline, virtual]`

Attempts to Lock the object, if the lock is already held by another thread than this method returns false.

Returns

true if the lock was acquired, false if it is already held by another thread.

Exceptions

<i>RuntimeException</i>	if an error occurs while locking the object.
--------------------------------	--

Implements **decaf::util::concurrent::Synchronizable** (p. 2643).

6.270.3.18 `virtual void decaf::io::InputStream::unlock () [inline, virtual]`

Unlocks the object.

Exceptions

<i>RuntimeException</i>	if an error occurs while unlocking the object.
-------------------------	--

Implements **decaf::util::concurrent::Synchronizable** (p. 2645).

6.270.3.19 `virtual void decaf::io::InputStream::wait () [inline, virtual]`

Waits on a signal from this object, which is generated by a call to Notify.

Must have this object locked before calling.

Exceptions

<i>RuntimeException</i>	if an error occurs while waiting on the object.
<i>InterruptedException</i>	if the wait is interrupted before it completes.
<i>IllegalMonitorState-Exception</i>	- if the current thread is not the owner of the the Synchronizable Object.

Implements **decaf::util::concurrent::Synchronizable** (p. 2646).

6.270.3.20 `virtual void decaf::io::InputStream::wait (long long millisecs) [inline, virtual]`

Waits on a signal from this object, which is generated by a call to Notify.

Must have this object locked before calling. This wait will timeout after the specified time interval.

Parameters

<i>millisecs</i>	the time in milliseconds to wait, or WAIT_INFINITE
------------------	--

Exceptions

<i>RuntimeException</i>	if an error occurs while waiting on the object.
<i>InterruptedException</i>	if the wait is interrupted before it completes.
<i>IllegalMonitorState-Exception</i>	- if the current thread is not the owner of the the Synchronizable Object.

Implements **decaf::util::concurrent::Synchronizable** (p. 2647).

6.270.3.21 `virtual void decaf::io::InputStream::wait (long long millisecs, int nanos) [inline, virtual]`

Waits on a signal from this object, which is generated by a call to Notify.

Must have this object locked before calling. This wait will timeout after the specified time interval. This method is similar to the one argument wait function except that it add

a finer grained control over the amount of time that it waits by adding in the additional nanosecond argument.

NOTE: The ability to wait accurately at a nanosecond scale depends on the platform and OS that the Decaf API is running on, some systems do not provide an accurate enough clock to provide this level of granularity.

Parameters

<i>milliseconds</i>	the time in milliseconds to wait, or WAIT_INFINITE
<i>nanos</i>	additional time in nanoseconds with a range of 0-999999

Exceptions

<i>IllegalArgumentException</i>	if an error occurs or the nanos argument is not in the range of [0-999999]
<i>RuntimeException</i>	if an error occurs while waiting on the object.
<i>InterruptedException</i>	if the wait is interrupted before it completes.
<i>IllegalMonitorStateException</i>	- if the current thread is not the owner of the the Synchronizable Object.

Implements **decaf::util::concurrent::Synchronizable** (p. 2648).

The documentation for this class was generated from the following file:

- src/main/decaf/io/InputStream.h

6.271 decaf::io::InputStreamReader Class Reference

An **InputStreamReader** (p. 1475) is a bridge from byte streams to character streams.

```
#include <src/main/decaf/io/InputStreamReader.h>
```

Inheritance diagram for decaf::io::InputStreamReader:

Public Member Functions

- **InputStreamReader** (**InputStream** *stream, bool own=false)
Create a new **InputStreamReader** (p. 1475) that wraps the given **InputStream** (p. 1464).
- virtual ~**InputStreamReader** ()
- virtual void **close** ()
Closes this object and deallocates the appropriate resources.
- virtual bool **ready** () const
Tells whether this stream is ready to be read.

Protected Member Functions

- virtual int **doReadArrayBounded** (char *buffer, int size, int offset, int length)
Override this method to customize the functionality of the method read(unsigned char buffer, int size, int offset, int length).*
- virtual void **checkClosed** () const

6.271.1 Detailed Description

An **InputStreamReader** (p. 1475) is a bridge from byte streams to character streams.

For top efficiency, consider wrapping an **InputStreamReader** (p. 1475) within a `BufferedReader`. For example:

```
BufferedReader* in = new BufferedReader( new InputStreamReader (p. 1475)( -
System.in, false ), true );
```

See also

OutputStreamWriter (p. 2074)

Since

1.0

6.271.2 Constructor & Destructor Documentation

6.271.2.1 **decaf::io::InputStreamReader::InputStreamReader** (**InputStream** * *stream*, bool *own* = false)

Create a new **InputStreamReader** (p. 1475) that wraps the given **InputStream** (p. 1464).

Parameters

<i>stream</i>	The InputStream (p. 1464) to read from. (cannot be null).
<i>own</i>	Does this object own the passed InputStream (p. 1464) (defaults to false).

Exceptions

<i>NullPointerException</i>	if the passed InputStream (p. 1464) is NULL.
-----------------------------	---

6.271.2.2 **virtual decaf::io::InputStreamReader::~~InputStreamReader** ()
[virtual]

6.271.3 Member Function Documentation

6.271.3.1 **virtual void decaf::io::InputStreamReader::checkClosed () const**
[protected, virtual]

6.271.3.2 **virtual void decaf::io::InputStreamReader::close ()** [virtual]

Closes this object and deallocates the appropriate resources.

The object is generally no longer usable after calling close.

Exceptions

<i>IOException</i> (p. 1545)	if an error occurs while closing.
--	-----------------------------------

Implements **decaf::io::Closeable** (p. 817).

6.271.3.3 **virtual int decaf::io::InputStreamReader::doReadArrayBounded (char *
buffer, int size, int offset, int length)** [protected, virtual]

Override this method to customize the functionality of the method read(unsigned char*
buffer, int size, int offset, int length).

All subclasses must override this method to provide the basic **Reader** (p. 2237) functionality.

Implements **decaf::io::Reader** (p. 2239).

6.271.3.4 **virtual bool decaf::io::InputStreamReader::ready () const** [virtual]

Tells whether this stream is ready to be read.

Returns

True if the next **read()** (p. 2242) is guaranteed not to block for input, false otherwise.
Note that returning false does not guarantee that the next read will block.

Exceptions

<i>IOException</i> (p. 1545)	if an I/O error occurs.
--	-------------------------

Reimplemented from **decaf::io::Reader** (p. 2242).

The documentation for this class was generated from the following file:

- src/main/decaf/io/InputStreamReader.h

6.272 decaf::internal::nio::IntArrayBuffer Class Reference

```
#include <src/main/decaf/internal/nio/IntArrayBuffer.h>
```

Inheritance diagram for decaf::internal::nio::IntArrayBuffer:

Public Member Functions

- **IntArrayBuffer** (int size, bool readOnly=false)
*Creates a **IntArrayBuffer** (p. 1477) object that has its backing array allocated internally and is then owned and deleted when this object is deleted.*
- **IntArrayBuffer** (int *array, int size, int offset, int length, bool readOnly=false)
*Creates a **IntArrayBuffer** (p. 1477) object that wraps the given array.*
- **IntArrayBuffer** (const decaf::lang::Pointer< **ByteArrayAdapter** > &array, int offset, int length, bool readOnly=false)
Creates a byte buffer that wraps the passed ByteArrayAdapter and start at the given offset.
- **IntArrayBuffer** (const **IntArrayBuffer** &other)
*Create a **IntArrayBuffer** (p. 1477) that mirrors this one, meaning it shares a reference to this buffers ByteArrayAdapter and when changes are made to that data it is reflected in both.*
- virtual ~**IntArrayBuffer** ()
- virtual int * **array** ()
*Returns the int array that backs this buffer (optional operation).
Modifications to this buffer's content will cause the returned array's content to be modified, and vice versa.
Invoke the hasArray method before invoking this method in order to ensure that this buffer has an accessible backing array.*
Returns
*the array that backs this **Buffer** (p. 582).*

Exceptions

ReadOnlyBuffer-Exception (p. 2244)	<i>if this Buffer (p. 582) is read only.</i>
UnsupportedOperation-Exception	<i>if the underlying store has no array.</i>

- virtual int **arrayOffset** ()
*Returns the offset within this buffer's backing array of the first element of the buffer (optional operation).
Invoke the hasArray method before invoking this method in order to ensure that this buffer has an accessible backing array.*
Returns
The offset into the backing array where index zero starts.

Exceptions

ReadOnlyBuffer-Exception (p. 2244)	if this Buffer (p. 582) is read only.
UnsupportedOperation-Exception	if the underlying store has no array.

- virtual **IntBuffer** * **asReadOnlyBuffer** () const

Creates a new, read-only int buffer that shares this buffer's content.

The content of the new buffer will be that of this buffer. Changes to this buffer's content will be visible in the new buffer; the new buffer itself, however, will be read-only and will not allow the shared content to be modified. The two buffers' position, limit, and mark values will be independent.

If this buffer is itself read-only then this method behaves in exactly the same way as the duplicate method.

The new buffer's capacity, limit, position, and mark values will be identical to those of this buffer.

Returns

The new, read-only int buffer which the caller then owns.

- virtual **IntBuffer** & **compact** ()

Compacts this buffer.

The bytes between the buffer's current position and its limit, if any, are copied to the beginning of the buffer. That is, the byte at index $p = \text{position}()$ (p. 587) is copied to index zero, the byte at index $p + 1$ is copied to index one, and so forth until the byte at index $\text{limit}()$ (p. 586) - 1 is copied to index $n = \text{limit}()$ (p. 586) - 1 - p . The buffer's position is then set to $n+1$ and its limit is set to its capacity. The mark, if defined, is discarded.

The buffer's position is set to the number of bytes copied, rather than to zero, so that an invocation of this method can be followed immediately by an invocation of another relative put method.

Returns

*a reference to this **IntBuffer** (p. 1488)*

Exceptions

ReadOnlyBuffer-Exception (p. 2244)	if this buffer is read-only.
--	------------------------------

- virtual **IntBuffer** * **duplicate** ()

Creates a new int buffer that shares this buffer's content.

The content of the new buffer will be that of this buffer. Changes to this buffer's content will be visible in the new buffer, and vice versa; the two buffers' position, limit, and mark values will be independent.

The new buffer's capacity, limit, position, and mark values will be identical to those of this buffer. The new buffer will be read-only if, and only if, this buffer is read-only.

Returns

*a new int **Buffer** (p. 582) which the caller owns.*

- virtual int **get** ()

Relative get method.

Reads the value at this buffer's current position, and then increments the position.

Returns

the int at the current position.

Exceptions

BufferUnderflow-Exception (p. 611)	<i>if there no more data to return.</i>
--	---

- virtual int **get** (int index) const

Absolute get method.

Reads the value at the given index.

Parameters

index	<i>The index in the Buffer (p. 582) where the int is to be read.</i>
-------	---

Returns

the int that is located at the given index.

Exceptions

IndexOutOfBounds-Exception	<i>if index is not smaller than the buffer's limit, or index is negative.</i>
----------------------------	---

- virtual bool **hasArray** () const

Tells whether or not this buffer is backed by an accessible int array.

If this method returns true then the array and arrayOffset methods may safely be invoked. Subclasses should override this method if they do not have a backing array as this class always returns true.

Returns

true if, and only if, this buffer is backed by an array and is not read-only.

- virtual bool **isReadOnly** () const

Tells whether or not this buffer is read-only.

Returns

true if, and only if, this buffer is read-only.

- virtual **IntArrayBuffer & put** (int value)

Writes the given integer into this buffer at the current position, and then increments the position.

Parameters

value	<i>The integer value to be written.</i>
-------	---

Returns

a reference to this buffer.

Exceptions

BufferOverflow-Exception (p. 609)	<i>if this buffer's current position is not smaller than its limit.</i>
ReadOnlyBuffer-Exception (p. 2244)	<i>if this buffer is read-only.</i>

- virtual **IntArrayBuffer & put** (int index, int value)

Writes the given ints into this buffer at the given index.

Parameters

index	<i>The position in the Buffer (p. 582) to write the data.</i>
value	<i>The ints to write.</i>

Returns

a reference to this buffer.

Exceptions

IndexOutOfBoundsException	- If index greater than the buffer's limit minus the size of the type being written, or the index is negative.
ReadOnlyBuffer-Exception (p. 2244)	- If this buffer is read-only.

- virtual **IntBuffer * slice ()** const

*Creates a new **IntBuffer** (p. 1488) whose content is a shared subsequence of this buffer's content.*

The content of the new buffer will start at this buffer's current position. Changes to this buffer's content will be visible in the new buffer, and vice versa; the two buffers' position, limit, and mark values will be independent.

The new buffer's position will be zero, its capacity and its limit will be the number of bytes remaining in this buffer, and its mark will be undefined. The new buffer will be read-only if, and only if, this buffer is read-only.

Returns

*the newly create **IntBuffer** (p. 1488) which the caller owns.*

Protected Member Functions

- virtual void **setReadOnly** (bool value)

*Sets this **IntArrayBuffer** (p. 1477) as Read-Only.*

6.272.1 Constructor & Destructor Documentation

6.272.1.1 **decaf::internal::nio::IntArrayBuffer::IntArrayBuffer** (int size, bool readOnly = false)

Creates a **IntArrayBuffer** (p. 1477) object that has its backing array allocated internally and is then owned and deleted when this object is deleted.

The array is initially created with all elements initialized to zero.

Parameters

size	The size of the array, this is the limit we read and write to.
readOnly	Boolean indicating if this buffer should be read-only, default as false.

Exceptions

<i>IllegalArgument-Exception</i>	if the capacity value is negative.
----------------------------------	------------------------------------

6.272.1.2 decaf::internal::nio::IntArrayBuffer::IntArrayBuffer (int * array, int size, int offset, int length, bool readOnly = false)

Creates a **IntArrayBuffer** (p. 1477) object that wraps the given array.

If the own flag is set then it will delete this array when this object is deleted.

Parameters

<i>array</i>	The actual array to wrap.
<i>size</i>	The size of the given array.
<i>offset</i>	The position that is this buffers start position.
<i>length</i>	The limit of how many bytes into the array this Buffer can write.
<i>readOnly</i>	Boolean indicating if this buffer should be read-only, default as false.

Exceptions

<i>NullPointerException</i>	if buffer is NULL
<i>IndexOutOfBoundsException</i>	if offset is greater than array capacity.

6.272.1.3 decaf::internal::nio::IntArrayBuffer::IntArrayBuffer (const decaf::lang::Pointer< ByteArrayAdapter > & array, int offset, int length, bool readOnly = false)

Creates a byte buffer that wraps the passed ByteArrayAdapter and start at the given offset.

The capacity and limit of the new **IntArrayBuffer** (p. 1477) will be that of the remaining capacity of the passed buffer.

Parameters

<i>array</i>	The ByteArrayAdapter to wrap.
<i>offset</i>	The position that is this buffers start position.
<i>length</i>	The limit of how many bytes into the array this Buffer can write.
<i>readOnly</i>	Boolean indicating if this buffer should be read-only, default as false.

Exceptions

<i>NullPointerException</i>	if array is NULL
<i>IndexOutOfBoundsException</i>	if offset + length is greater than array size.

6.272.1.4 `decaf::internal::nio::IntArrayBuffer::IntArrayBuffer (const IntArrayBuffer & other)`

Create a **IntArrayBuffer** (p. 1477) that mirrors this one, meaning it shares a reference to this buffers `ByteArrayAdapter` and when changes are made to that data it is reflected in both.

Parameters

<i>other</i>	The IntArrayBuffer (p. 1477) this one is to mirror.
--------------	--

6.272.1.5 `virtual decaf::internal::nio::IntArrayBuffer::~~IntArrayBuffer ()` [virtual]

6.272.2 Member Function Documentation

6.272.2.1 `virtual int* decaf::internal::nio::IntArrayBuffer::array ()` [virtual]

Returns the int array that backs this buffer (optional operation).

Modifications to this buffer's content will cause the returned array's content to be modified, and vice versa.

Invoke the `hasArray` method before invoking this method in order to ensure that this buffer has an accessible backing array.

Returns

the array that backs this **Buffer** (p. 582).

Exceptions

<i>ReadOnlyBuffer-Exception</i> (p. 2244)	if this Buffer (p. 582) is read only.
<i>Unsupported-OperationException</i>	if the underlying store has no array.

Implements **decaf::nio::IntBuffer** (p. 1491).

6.272.2.2 `virtual int decaf::internal::nio::IntArrayBuffer::arrayOffset ()` [virtual]

Returns the offset within this buffer's backing array of the first element of the buffer (optional operation).

Invoke the `hasArray` method before invoking this method in order to ensure that this buffer has an accessible backing array.

Returns

The offset into the backing array where index zero starts.

Exceptions

<i>ReadOnlyBufferException</i> (p. 2244)	if this Buffer (p. 582) is read only.
<i>UnsupportedOperationException</i>	if the underlying store has no array.

Implements **decaf::nio::IntBuffer** (p. 1491).

6.272.2.3 `virtual IntBuffer* decaf::internal::nio::IntArrayBuffer::asReadOnlyBuffer () const [virtual]`

Creates a new, read-only int buffer that shares this buffer's content.

The content of the new buffer will be that of this buffer. Changes to this buffer's content will be visible in the new buffer; the new buffer itself, however, will be read-only and will not allow the shared content to be modified. The two buffers' position, limit, and mark values will be independent.

If this buffer is itself read-only then this method behaves in exactly the same way as the duplicate method.

The new buffer's capacity, limit, position, and mark values will be identical to those of this buffer.

Returns

The new, read-only int buffer which the caller then owns.

Implements **decaf::nio::IntBuffer** (p. 1492).

6.272.2.4 `virtual IntBuffer& decaf::internal::nio::IntArrayBuffer::compact () [virtual]`

Compacts this buffer.

The bytes between the buffer's current position and its limit, if any, are copied to the beginning of the buffer. That is, the byte at index $p = \text{position}()$ (p. 587) is copied to index zero, the byte at index $p + 1$ is copied to index one, and so forth until the byte at index $\text{limit}()$ (p. 586) - 1 is copied to index $n = \text{limit}()$ (p. 586) - 1 - p . The buffer's position is then set to $n+1$ and its limit is set to its capacity. The mark, if defined, is discarded.

The buffer's position is set to the number of bytes copied, rather than to zero, so that an invocation of this method can be followed immediately by an invocation of another relative put method.

Returns

a reference to this **IntBuffer** (p. 1488)

Exceptions

<i>ReadOnlyBuffer-Exception</i> (p. 2244)	if this buffer is read-only.
---	------------------------------

Implements **decaf::nio::IntBuffer** (p. 1492).

6.272.2.5 `virtual IntBuffer* decaf::internal::nio::IntArrayBuffer::duplicate ()`
[virtual]

Creates a new int buffer that shares this buffer's content.

The content of the new buffer will be that of this buffer. Changes to this buffer's content will be visible in the new buffer, and vice versa; the two buffers' position, limit, and mark values will be independent.

The new buffer's capacity, limit, position, and mark values will be identical to those of this buffer. The new buffer will be read-only if, and only if, this buffer is read-only.

Returns

a new int **Buffer** (p. 582) which the caller owns.

Implements **decaf::nio::IntBuffer** (p. 1493).

6.272.2.6 `virtual int decaf::internal::nio::IntArrayBuffer::get ()` [virtual]

Relative get method.

Reads the value at this buffer's current position, and then increments the position.

Returns

the int at the current position.

Exceptions

<i>BufferUnderflow-Exception</i> (p. 611)	if there no more data to return.
---	----------------------------------

Implements **decaf::nio::IntBuffer** (p. 1493).

6.272.2.7 `virtual int decaf::internal::nio::IntArrayBuffer::get (int index) const`
`[virtual]`

Absolute get method.

Reads the value at the given index.

Parameters

<i>index</i>	The index in the Buffer (p. 582) where the int is to be read.
--------------	--

Returns

the int that is located at the given index.

Exceptions

<i>IndexOutOfBoundsException</i>	if index is not smaller than the buffer's limit, or index is negative.
----------------------------------	--

Implements **decaf::nio::IntBuffer** (p. 1493).

6.272.2.8 `virtual bool decaf::internal::nio::IntArrayBuffer::hasArray () const`
`[inline, virtual]`

Tells whether or not this buffer is backed by an accessible int array.

If this method returns true then the `array` and `arrayOffset` methods may safely be invoked. Subclasses should override this method if they do not have a backing array as this class always returns true.

Returns

true if, and only if, this buffer is backed by an array and is not read-only.

Implements **decaf::nio::IntBuffer** (p. 1495).

6.272.2.9 `virtual bool decaf::internal::nio::IntArrayBuffer::isReadOnly () const`
`[inline, virtual]`

Tells whether or not this buffer is read-only.

Returns

true if, and only if, this buffer is read-only.

Implements **decaf::nio::Buffer** (p. 586).

6.272.2.10 `virtual IntBuffer& decaf::internal::nio::IntArrayBuffer::put (int value)`
`[virtual]`

Writes the given integer into this buffer at the current position, and then increments the position.

Parameters

<i>value</i>	The integer value to be written.
--------------	----------------------------------

Returns

a reference to this buffer.

Exceptions

<i>BufferOverflow-Exception</i> (p. 609)	if this buffer's current position is not smaller than its limit.
<i>ReadOnlyBuffer-Exception</i> (p. 2244)	if this buffer is read-only.

Implements **decaf::nio::IntBuffer** (p. 1497).

6.272.2.11 `virtual IntBuffer& decaf::internal::nio::IntArrayBuffer::put (int index, int value)`
`[virtual]`

Writes the given ints into this buffer at the given index.

Parameters

<i>index</i>	The position in the Buffer (p. 582) to write the data.
<i>value</i>	The ints to write.

Returns

a reference to this buffer.

Exceptions

<i>IndexOutOfBoundsException</i>	- If index greater than the buffer's limit minus the size of the type being written, or the index is negative.
<i>ReadOnlyBuffer-Exception</i> (p. 2244)	- If this buffer is read-only.

Implements **decaf::nio::IntBuffer** (p. 1498).

6.272.2.12 `virtual void decaf::internal::nio::IntArrayBuffer::setReadOnly (bool value) [inline, protected, virtual]`

Sets this **IntArrayBuffer** (p. 1477) as Read-Only.

Parameters

<i>value</i>	Boolean value, true if this buffer is to be read-only, false otherwise.
--------------	---

6.272.2.13 `virtual IntBuffer* decaf::internal::nio::IntArrayBuffer::slice () const [virtual]`

Creates a new **IntBuffer** (p. 1488) whose content is a shared subsequence of this buffer's content.

The content of the new buffer will start at this buffer's current position. Changes to this buffer's content will be visible in the new buffer, and vice versa; the two buffers' position, limit, and mark values will be independent.

The new buffer's position will be zero, its capacity and its limit will be the number of bytes remaining in this buffer, and its mark will be undefined. The new buffer will be read-only if, and only if, this buffer is read-only.

Returns

the newly create **IntBuffer** (p. 1488) which the caller owns.

Implements **decaf::nio::IntBuffer** (p. 1498).

The documentation for this class was generated from the following file:

- `src/main/decaf/internal/nio/IntArrayBuffer.h`

6.273 decaf::nio::IntBuffer Class Reference

This class defines four categories of operations upon int buffers:

```
#include <src/main/decaf/nio/IntBuffer.h>
```

Inheritance diagram for `decaf::nio::IntBuffer`:

Public Member Functions

- `virtual ~IntBuffer ()`
- `virtual std::string toString () const`
- `virtual int * array ()=0`

Returns the int array that backs this buffer (optional operation).

- virtual int **arrayOffset** ()=0

Returns the offset within this buffer's backing array of the first element of the buffer (optional operation).

- virtual **IntBuffer** * **asReadOnlyBuffer** () const =0

Creates a new, read-only int buffer that shares this buffer's content.

- virtual **IntBuffer** & **compact** ()=0

Compacts this buffer.

- virtual **IntBuffer** * **duplicate** ()=0

Creates a new int buffer that shares this buffer's content.

- virtual int **get** ()=0

Relative get method.

- virtual int **get** (int index) const =0

Absolute get method.

- **IntBuffer** & **get** (std::vector< int > buffer)

Relative bulk get method.

- **IntBuffer** & **get** (int *buffer, int size, int offset, int length)

Relative bulk get method.

- virtual bool **hasArray** () const =0

Tells whether or not this buffer is backed by an accessible int array.

- **IntBuffer** & **put** (**IntBuffer** &src)

This method transfers the ints remaining in the given source buffer into this buffer.

- **IntBuffer** & **put** (const int *buffer, int size, int offset, int length)

This method transfers ints into this buffer from the given source array.

- **IntBuffer** & **put** (std::vector< int > &buffer)

This method transfers the entire content of the given source ints array into this buffer.

- virtual **IntBuffer** & **put** (int value)=0

Writes the given integer into this buffer at the current position, and then increments the position.

- virtual **IntBuffer** & **put** (int index, int value)=0

Writes the given ints into this buffer at the given index.

- virtual **IntBuffer** * **slice** () const =0

*Creates a new **IntBuffer** (p. 1488) whose content is a shared subsequence of this buffer's content.*

- virtual int **compareTo** (const **IntBuffer** &value) const

- virtual bool **equals** (const **IntBuffer** &value) const

- virtual bool **operator==** (const **IntBuffer** &value) const

- virtual bool **operator<** (const **IntBuffer** &value) const

Static Public Member Functions

- static **IntBuffer** * **allocate** (int **capacity**)
Allocates a new Double buffer.
- static **IntBuffer** * **wrap** (int *array, int size, int offset, int length)
*Wraps the passed buffer with a new **IntBuffer** (p. 1488).*
- static **IntBuffer** * **wrap** (std::vector< int > &buffer)
*Wraps the passed STL int Vector in a **IntBuffer** (p. 1488).*

Protected Member Functions

- **IntBuffer** (int **capacity**)
*Creates a **IntBuffer** (p. 1488) object that has its backing array allocated internally and is then owned and deleted when this object is deleted.*

6.273.1 Detailed Description

This class defines four categories of operations upon int buffers:

o Absolute and relative get and put methods that read and write single ints; o Relative bulk get methods that transfer contiguous sequences of ints from this buffer into an array; and o Relative bulk put methods that transfer contiguous sequences of ints from a int array or some other int buffer into this buffer o Methods for compacting, duplicating, and slicing a int buffer.

Double buffers can be created either by allocation, which allocates space for the buffer's content, by wrapping an existing int array into a buffer, or by creating a view of an existing byte buffer

Methods in this class that do not otherwise have a value to return are specified to return the buffer upon which they are invoked. This allows method invocations to be chained.

6.273.2 Constructor & Destructor Documentation

6.273.2.1 decaf::nio::IntBuffer::IntBuffer (int *capacity*) [protected]

Creates a **IntBuffer** (p. 1488) object that has its backing array allocated internally and is then owned and deleted when this object is deleted.

The array is initially created with all elements initialized to zero.

Parameters

<i>capacity</i>	The size and limit of the Buffer (p. 582) in integers.
-----------------	---

Exceptions

<i>IllegalArgument-Exception</i>	if capacity is negative.
----------------------------------	--------------------------

6.273.2.2 `virtual decaf::nio::IntBuffer::~~IntBuffer() [inline, virtual]`

6.273.3 Member Function Documentation

6.273.3.1 `static IntBuffer* decaf::nio::IntBuffer::allocate(int capacity) [static]`

Allocates a new Double buffer.

The new buffer's position will be zero, its limit will be its capacity, and its mark will be undefined. It will have a backing array, and its array offset will be zero.

Parameters

<i>capacity</i>	The size of the Double buffer in integers.
-----------------	--

Returns

the **IntBuffer** (p. 1488) that was allocated, caller owns.

6.273.3.2 `virtual int* decaf::nio::IntBuffer::array() [pure virtual]`

Returns the int array that backs this buffer (optional operation).

Modifications to this buffer's content will cause the returned array's content to be modified, and vice versa.

Invoke the `hasArray` method before invoking this method in order to ensure that this buffer has an accessible backing array.

Returns

the array that backs this **Buffer** (p. 582).

Exceptions

<i>ReadOnlyBufferException</i> (p. 2244)	if this Buffer (p. 582) is read only.
<i>UnsupportedOperationException</i>	if the underlying store has no array.

Implemented in **decaf::internal::nio::IntArrayBuffer** (p. 1482).

6.273.3.3 `virtual int decaf::nio::IntBuffer::arrayOffset() [pure virtual]`

Returns the offset within this buffer's backing array of the first element of the buffer (optional operation).

Invoke the `hasArray` method before invoking this method in order to ensure that this

buffer has an accessible backing array.

Returns

The offset into the backing array where index zero starts.

Exceptions

<i>ReadOnlyBufferException</i> (p. 2244)	if this Buffer (p. 582) is read only.
<i>UnsupportedOperationException</i>	if the underlying store has no array.

Implemented in **decaf::internal::nio::IntArrayBuffer** (p. 1483).

6.273.3.4 `virtual IntBuffer* decaf::nio::IntBuffer::asReadOnlyBuffer () const`
[pure virtual]

Creates a new, read-only int buffer that shares this buffer's content.

The content of the new buffer will be that of this buffer. Changes to this buffer's content will be visible in the new buffer; the new buffer itself, however, will be read-only and will not allow the shared content to be modified. The two buffers' position, limit, and mark values will be independent.

If this buffer is itself read-only then this method behaves in exactly the same way as the duplicate method.

The new buffer's capacity, limit, position, and mark values will be identical to those of this buffer.

Returns

The new, read-only int buffer which the caller then owns.

Implemented in **decaf::internal::nio::IntArrayBuffer** (p. 1483).

6.273.3.5 `virtual IntBuffer& decaf::nio::IntBuffer::compact ()` [pure virtual]

Compacts this buffer.

The bytes between the buffer's current position and its limit, if any, are copied to the beginning of the buffer. That is, the byte at index $p = \text{position}()$ (p. 587) is copied to index zero, the byte at index $p + 1$ is copied to index one, and so forth until the byte at index $\text{limit}()$ (p. 586) - 1 is copied to index $n = \text{limit}()$ (p. 586) - 1 - p . The buffer's position is then set to $n+1$ and its limit is set to its capacity. The mark, if defined, is discarded.

The buffer's position is set to the number of bytes copied, rather than to zero, so that an invocation of this method can be followed immediately by an invocation of another relative put method.

Returns

a reference to this **IntBuffer** (p. 1488)

Exceptions

<i>ReadOnlyBuffer-Exception</i> (p. 2244)	if this buffer is read-only.
---	------------------------------

Implemented in **decaf::internal::nio::IntArrayBuffer** (p. 1484).

6.273.3.6 `virtual int decaf::nio::IntBuffer::compareTo (const IntBuffer & value) const`
[virtual]

6.273.3.7 `virtual IntBuffer* decaf::nio::IntBuffer::duplicate ()` [pure virtual]

Creates a new int buffer that shares this buffer's content.

The content of the new buffer will be that of this buffer. Changes to this buffer's content will be visible in the new buffer, and vice versa; the two buffers' position, limit, and mark values will be independent.

The new buffer's capacity, limit, position, and mark values will be identical to those of this buffer. The new buffer will be read-only if, and only if, this buffer is read-only.

Returns

a new int **Buffer** (p. 582) which the caller owns.

Implemented in **decaf::internal::nio::IntArrayBuffer** (p. 1484).

6.273.3.8 `virtual bool decaf::nio::IntBuffer::equals (const IntBuffer & value) const`
[virtual]

6.273.3.9 `virtual int decaf::nio::IntBuffer::get ()` [pure virtual]

Relative get method.

Reads the value at this buffer's current position, and then increments the position.

Returns

the int at the current position.

Exceptions

<i>BufferUnderflow-Exception</i> (p. 611)	if there no more data to return.
---	----------------------------------

Implemented in **decaf::internal::nio::IntArrayBuffer** (p. 1485).

6.273.3.10 `virtual int decaf::nio::IntBuffer::get (int index) const` `[pure virtual]`

Absolute get method.

Reads the value at the given index.

Parameters

<i>index</i>	The index in the Buffer (p. 582) where the int is to be read.
--------------	--

Returns

the int that is located at the given index.

Exceptions

<i>IndexOutOfBoundsException</i>	if index is not smaller than the buffer's limit, or index is negative.
----------------------------------	--

Implemented in **decaf::internal::nio::IntArrayBuffer** (p. 1485).

6.273.3.11 `IntBuffer& decaf::nio::IntBuffer::get (std::vector< int > buffer)`

Relative bulk get method.

This method transfers values from this buffer into the given destination vector. An invocation of this method of the form `src.get(a)` behaves in exactly the same way as the invocation. The vector must be sized to the amount of data that is to be read, that is to say, the caller should call `buffer.resize(N)` before calling this get method.

Returns

a reference to this **Buffer** (p. 582).

Exceptions

<i>BufferUnderflow-Exception</i> (p. 611)	if there are fewer than length ints remaining in this buffer.
---	---

6.273.3.12 `IntBuffer& decaf::nio::IntBuffer::get (int * buffer, int size, int offset, int length)`

Relative bulk get method.

This method transfers ints from this buffer into the given destination array. If there are fewer ints remaining in the buffer than are required to satisfy the request, that is, if `length > remaining()` (p. 588), then no bytes are transferred and a **BufferUnderflowException** (p. 611) is thrown.

Otherwise, this method copies `length` ints from this buffer into the given array, starting at the current position of this buffer and at the given offset in the array. The position of this buffer is then incremented by `length`.

Parameters

<i>buffer</i>	The pointer to an allocated buffer to fill.
<i>size</i>	The size of the buffer that was passed in.
<i>offset</i>	The position in the buffer to start filling.
<i>length</i>	The amount of data to put in the passed buffer.

Returns

a reference to this **Buffer** (p. 582).

Exceptions

BufferUnderflowException (p. 611)	if there are fewer than <code>length</code> ints remaining in this buffer.
<i>NullPointerException</i>	if the passed buffer is null.
<i>IndexOutOfBoundsException</i>	if the preconditions of <code>size</code> , <code>offset</code> , or <code>length</code> are not met.

6.273.3.13 `virtual bool decaf::nio::IntBuffer::hasArray () const` [pure virtual]

Tells whether or not this buffer is backed by an accessible int array.

If this method returns true then the `array` and `arrayOffset` methods may safely be invoked. Subclasses should override this method if they do not have a backing array as this class always returns true.

Returns

true if, and only if, this buffer is backed by an array and is not read-only.

Implemented in **decaf::internal::nio::IntArrayBuffer** (p. 1486).

6.273.3.14 `virtual bool decaf::nio::IntBuffer::operator< (const IntBuffer & value) const`
[virtual]

6.273.3.15 `virtual bool decaf::nio::IntBuffer::operator== (const IntBuffer & value) const`
[virtual]

6.273.3.16 IntBuffer& decaf::nio::IntBuffer::put (IntBuffer & src)

This method transfers the ints remaining in the given source buffer into this buffer.

If there are more ints remaining in the source buffer than in this buffer, that is, if `src.remaining() > remaining()` (p. 588), then no ints are transferred and a **BufferOverflowException** (p. 609) is thrown.

Otherwise, this method copies `n = src.remaining()` ints from the given buffer into this buffer, starting at each buffer's current position. The positions of both buffers are then incremented by `n`.

Parameters

<i>src</i>	The buffer to take ints from an place in this one.
------------	--

Returns

a reference to this buffer.

Exceptions

BufferOverflowException (p. 609)	if there is insufficient space in this buffer for the remaining ints in the source buffer.
<i>IllegalArgumentException</i>	if the source buffer is this buffer.
ReadOnlyBufferException (p. 2244)	if this buffer is read-only.

6.273.3.17 IntBuffer& decaf::nio::IntBuffer::put (const int * buffer, int size, int offset, int length)

This method transfers ints into this buffer from the given source array.

If there are more ints to be copied from the array than remain in this buffer, that is, if `length > remaining()` (p. 588), then no ints are transferred and a **BufferOverflowException** (p. 609) is thrown.

Otherwise, this method copies `length` bytes from the given array into this buffer, starting at the given offset in the array and at the current position of this buffer. The position of this buffer is then incremented by `length`.

Parameters

<i>buffer</i>	The array from which integers are to be read.
<i>size</i>	The size of the buffer passed.
<i>offset</i>	The offset within the array of the first char to be read.
<i>length</i>	The number of integers to be read from the given array.

Returns

a reference to this buffer.

Exceptions

<i>BufferOverflow-Exception</i> (p. 609)	if there is insufficient space in this buffer.
<i>ReadOnlyBuffer-Exception</i> (p. 2244)	if this buffer is read-only.
<i>NullPointerException</i>	if the passed buffer is null.
<i>IndexOutOfBoundsException</i>	if the preconditions of size, offset, or length are not met.

6.273.3.18 `IntBuffer& decaf::nio::IntBuffer::put (std::vector< int > & buffer)`

This method transfers the entire content of the given source ints array into this buffer.

This is the same as calling `put(&buffer[0], 0, buffer.size())`.

Parameters

<i>buffer</i>	The buffer whose contents are copied to this IntBuffer (p. 1488).
---------------	--

Returns

a reference to this buffer.

Exceptions

<i>BufferOverflow-Exception</i> (p. 609)	if there is insufficient space in this buffer.
<i>ReadOnlyBuffer-Exception</i> (p. 2244)	if this buffer is read-only.

6.273.3.19 `virtual IntBuffer& decaf::nio::IntBuffer::put (int value) [pure virtual]`

Writes the given integer into this buffer at the current position, and then increments the position.

Parameters

<i>value</i>	The integer value to be written.
--------------	----------------------------------

Returns

a reference to this buffer.

Exceptions

<i>BufferOverflow-Exception</i> (p. 609)	if this buffer's current position is not smaller than its limit.
<i>ReadOnlyBuffer-Exception</i> (p. 2244)	if this buffer is read-only.

Implemented in **decaf::internal::nio::IntArrayBuffer** (p. 1486).

6.273.3.20 `virtual IntBuffer& decaf::nio::IntBuffer::put (int index, int value)` [pure virtual]

Writes the given ints into this buffer at the given index.

Parameters

<i>index</i>	The position in the Buffer (p. 582) to write the data.
<i>value</i>	The ints to write.

Returns

a reference to this buffer.

Exceptions

<i>IndexOutOfBoundsException</i>	- If index greater than the buffer's limit minus the size of the type being written, or the index is negative.
<i>ReadOnlyBuffer-Exception</i> (p. 2244)	- If this buffer is read-only.

Implemented in **decaf::internal::nio::IntArrayBuffer** (p. 1487).

6.273.3.21 `virtual IntBuffer* decaf::nio::IntBuffer::slice () const` [pure virtual]

Creates a new **IntBuffer** (p. 1488) whose content is a shared subsequence of this buffer's content.

The content of the new buffer will start at this buffer's current position. Changes to this buffer's content will be visible in the new buffer, and vice versa; the two buffers' position, limit, and mark values will be independent.

The new buffer's position will be zero, its capacity and its limit will be the number of bytes remaining in this buffer, and its mark will be undefined. The new buffer will be read-only if, and only if, this buffer is read-only.

Returns

the newly create **IntBuffer** (p. 1488) which the caller owns.

Implemented in **decaf::internal::nio::IntArrayBuffer** (p. 1487).

6.273.3.22 `virtual std::string decaf::nio::IntBuffer::toString () const` [virtual]

Returns

a std::string describing this object

6.273.3.23 `static IntBuffer* decaf::nio::IntBuffer::wrap (int * array, int size, int offset, int length)` [static]

Wraps the passed buffer with a new **IntBuffer** (p. 1488).

The new buffer will be backed by the given int array; that is, modifications to the buffer will cause the array to be modified and vice versa. The new buffer's capacity will be array.length, its position will be offset, its limit will be offset + length, and its mark will be undefined. Its backing array will be the given array, and its array offset will be zero.

Parameters

<i>array</i>	The array that will back the new buffer.
<i>size</i>	The size of the passed in array.
<i>offset</i>	The offset of the subarray to be used.
<i>length</i>	The length of the subarray to be used.

Returns

a new **IntBuffer** (p. 1488) that is backed by buffer, caller owns.

Exceptions

<i>NullPointerException</i>	if the array pointer is NULL.
<i>IndexOutOfBoundsException</i>	if the preconditions of size, offset, or length are not met.

6.273.3.24 `static IntBuffer* decaf::nio::IntBuffer::wrap (std::vector< int > & buffer)`
`[static]`

Wraps the passed STL int Vector in a **IntBuffer** (p. 1488).

The new buffer will be backed by the given int array; modifications to the buffer will cause the array to be modified and vice versa. The new buffer's capacity and limit will be `buffer.size()`, its position will be zero, and its mark will be undefined. Its backing array will be the given array, and its array offset will be zero.

Parameters

<i>buffer</i>	The vector that will back the new buffer, the vector must have been sized to the desired size already by calling <code>vector.resize(N)</code> .
---------------	--

Returns

a new **IntBuffer** (p. 1488) that is backed by `buffer`, caller owns.

The documentation for this class was generated from the following file:

- `src/main/decaf/nio/IntBuffer.h`

6.274 decaf::lang::Integer Class Reference

```
#include <src/main/decaf/lang/Integer.h>
```

Inheritance diagram for `decaf::lang::Integer`:

Public Member Functions

- **Integer** (int value)
- **Integer** (const std::string &value)
*Constructs a new **Integer** (p. 1500) and attempts to convert the given string to an int value, assigning it to the new object is successful or throwing a `NumberFormatException` if the string is not a properly formatted int.*
- virtual **~Integer** ()
- virtual int **compareTo** (const **Integer** &i) const
*Compares this **Integer** (p. 1500) instance with another.*

- bool **equals** (const **Integer** &i) const
- virtual bool **operator==** (const **Integer** &i) const
Compares equality between this object and the one passed.
- virtual bool **operator<** (const **Integer** &i) const
Compares this object to another and returns true if this object is considered to be less than the one passed.
- virtual int **compareTo** (const int &i) const
*Compares this **Integer** (p. 1500) instance with another.*
- bool **equals** (const int &i) const
- virtual bool **operator==** (const int &i) const
Compares equality between this object and the one passed.
- virtual bool **operator<** (const int &i) const
Compares this object to another and returns true if this object is considered to be less than the one passed.
- std::string **toString** () const
- virtual double **doubleValue** () const
Answers the double value which the receiver represents.
- virtual float **floatValue** () const
Answers the float value which the receiver represents.
- virtual unsigned char **byteValue** () const
Answers the byte value which the receiver represents.
- virtual short **shortValue** () const
Answers the short value which the receiver represents.
- virtual int **intValue** () const
Answers the int value which the receiver represents.
- virtual long long **longValue** () const
Answers the long value which the receiver represents.

Static Public Member Functions

- static **Integer decode** (const std::string &value)
*Decodes a **String** (p. 2620) into a **Integer** (p. 1500).*
- static int **reverseBytes** (int value)
Returns the value obtained by reversing the order of the bytes in the two's complement representation of the specified int value.
- static int **reverse** (int value)
Returns the value obtained by reversing the order of the bits in the two's complement binary representation of the specified int value.
- static int **parseInt** (const std::string &s, int radix)
Parses the string argument as a signed int in the radix specified by the second argument.
- static int **parseInt** (const std::string &s)
Parses the string argument as a signed decimal int.
- static **Integer valueOf** (int value)

Returns a **Integer** (p. 1500) instance representing the specified *int* value.

- static **Integer valueOf** (const std::string &value)

Returns a **Integer** (p. 1500) object holding the value given by the specified *std::string*.

- static **Integer valueOf** (const std::string &value, int radix)

Returns a **Integer** (p. 1500) object holding the value extracted from the specified *std::string* when parsed with the radix given by the second argument.

- static int **bitCount** (int value)

Returns the number of one-bits in the two's complement binary representation of the specified *int* value.

- static std::string **toString** (int value)

Converts the *int* to a **String** (p. 2620) representation.

- static std::string **toString** (int value, int radix)

Returns a string representation of the first argument in the radix specified by the second argument.

- static std::string **toHexString** (int value)

Returns a string representation of the integer argument as an unsigned integer in base 16.

- static std::string **toOctalString** (int value)

Returns a string representation of the integer argument as an unsigned integer in base 8.

- static std::string **toBinaryString** (int value)

Returns a string representation of the integer argument as an unsigned integer in base 2.

- static int **highestOneBit** (int value)

Returns an *int* value with at most a single one-bit, in the position of the highest-order ("leftmost") one-bit in the specified *int* value.

- static int **lowestOneBit** (int value)

Returns an *int* value with at most a single one-bit, in the position of the lowest-order ("rightmost") one-bit in the specified *int* value.

- static int **numberOfLeadingZeros** (int value)

Returns the number of zero bits preceding the highest-order ("leftmost") one-bit in the two's complement binary representation of the specified *int* value.

- static int **numberOfTrailingZeros** (int value)

Returns the number of zero bits following the lowest-order ("rightmost") one-bit in the two's complement binary representation of the specified *int* value.

- static int **rotateLeft** (int value, int distance)

Returns the value obtained by rotating the two's complement binary representation of the specified *int* value left by the specified number of bits.

- static int **rotateRight** (int value, int distance)

Returns the value obtained by rotating the two's complement binary representation of the specified *int* value right by the specified number of bits.

- static int **signum** (int value)

Returns the signum function of the specified *int* value.

Static Public Attributes

- static const int **SIZE** = 32
The size in bits of the primitive int type.
- static const int **MAX_VALUE** = (int)0x7FFFFFFF
The maximum value that the primitive type can hold.
- static const int **MIN_VALUE** = (int)0x80000000
The minimum value that the primitive type can hold.

6.274.1 Constructor & Destructor Documentation

6.274.1.1 `decaf::lang::Integer::Integer (int value)`

Parameters

<i>value</i>	The primitive value to wrap in an Integer (p.1500) instance.
--------------	---

6.274.1.2 `decaf::lang::Integer::Integer (const std::string & value)`

Constructs a new **Integer** (p.1500) and attempts to convert the given string to an int value, assigning it to the new object is successful or throwing a `NumberFormatException` if the string is not a properly formatted int.

Parameters

<i>value</i>	The string to convert to a primitive type to wrap.
--------------	--

Exceptions

<i>NumberFormatException</i>	if the string is not a a valid integer.
------------------------------	---

6.274.1.3 `virtual decaf::lang::Integer::~~Integer ()` `[inline, virtual]`

6.274.2 Member Function Documentation

6.274.2.1 `static int decaf::lang::Integer::bitCount (int value)` `[static]`

Returns the number of one-bits in the two's complement binary representation of the specified int value.

This function is sometimes referred to as the population count.

Parameters

<i>value</i>	- the int to count
--------------	--------------------

Returns

the number of one-bits in the two's complement binary representation of the specified int value.

6.274.2.2 `virtual unsigned char decaf::lang::Integer::byteValue () const` `[inline, virtual]`

Answers the byte value which the receiver represents.

Returns

int the value of the receiver.

Reimplemented from **decaf::lang::Number** (p. 1993).

6.274.2.3 `virtual int decaf::lang::Integer::compareTo (const Integer & i) const` `[virtual]`

Compares this **Integer** (p. 1500) instance with another.

Parameters

<i>i</i>	- the Integer (p. 1500) instance to be compared
----------	--

Returns

zero if this object represents the same integer value as the argument; a positive value if this object represents a value greater than the passed in value, and -1 if this object represents a value less than the passed in value.

Implements **decaf::lang::Comparable**< **Integer** > (p. 886).

6.274.2.4 `virtual int decaf::lang::Integer::compareTo (const int & i) const` `[virtual]`

Compares this **Integer** (p. 1500) instance with another.

Parameters

<i>i</i>	- the Integer (p. 1500) instance to be compared
----------	--

Returns

zero if this object represents the same integer value as the argument; a positive value if this object represents a value greater than the passed in value, and -1 if this object represents a value less than the passed in value.

Implements **decaf::lang::Comparable**< **int** > (p. 886).

6.274.2.5 `static Integer decaf::lang::Integer::decode (const std::string & value)`
`[static]`

Decodes a **String** (p. 2620) into a **Integer** (p. 1500).

Accepts decimal, hexadecimal, and octal numbers given by the following grammar:

The sequence of characters following an (optional) negative sign and/or radix specifier ("0x", "0X", "#", or leading zero) is parsed as by the Integer.parseInt method with the indicated radix (10, 16, or 8). This sequence of characters must represent a positive value or a NumberFormatException will be thrown. The result is negated if first character of the specified **String** (p. 2620) is the minus sign. No whitespace characters are permitted in the string.

Parameters

<i>value</i>	- The string to decode
--------------	------------------------

Returns

a **Integer** (p. 1500) object containing the decoded value

Exceptions

<i>NumberFormatException</i>	if the string is not formatted correctly.
------------------------------	---

6.274.2.6 `virtual double decaf::lang::Integer::doubleValue () const` `[inline, virtual]`

Answers the double value which the receiver represents.

Returns

double the value of the receiver.

Implements **decaf::lang::Number** (p. 1993).

6.274.2.7 `bool decaf::lang::Integer::equals (const Integer & i) const` `[inline, virtual]`

Parameters

<i>i</i>	- the Integer (p. 1500) object to compare against.
----------	---

Returns

true if the two **Integer** (p. 1500) Objects have the same value.

Implements **decaf::lang::Comparable**< **Integer** > (p. 887).

6.274.2.8 `bool decaf::lang::Integer::equals (const int & i) const` `[inline, virtual]`

Parameters

<i>i</i>	- the Integer (p. 1500) object to compare against.
----------	---

Returns

true if the two **Integer** (p. 1500) Objects have the same value.

Implements **decaf::lang::Comparable**< **int** > (p. 887).

6.274.2.9 `virtual float decaf::lang::Integer::floatValue () const` `[inline, virtual]`

Answers the float value which the receiver represents.

Returns

float the value of the receiver.

Implements **decaf::lang::Number** (p. 1993).

6.274.2.10 `static int decaf::lang::Integer::highestOneBit (int value)` `[static]`

Returns an int value with at most a single one-bit, in the position of the highest-order ("leftmost") one-bit in the specified int value.

Returns zero if the specified value has no one-bits in its two's complement binary representation, that is, if it is equal to zero.

Parameters

<i>value</i>	- the int to be inspected
--------------	---------------------------

Returns

an int value with a single one-bit, in the position of the highest-order one-bit in the specified value, or zero if the specified value is itself equal to zero.

6.274.2.11 `virtual int decaf::lang::Integer::intValue () const [inline, virtual]`

Answers the int value which the receiver represents.

Returns

int the value of the receiver.

Implements **decaf::lang::Number** (p. 1993).

6.274.2.12 `virtual long long decaf::lang::Integer::longValue () const [inline, virtual]`

Answers the long value which the receiver represents.

Returns

long the value of the receiver.

Implements **decaf::lang::Number** (p. 1994).

6.274.2.13 `static int decaf::lang::Integer::lowestOneBit (int value) [static]`

Returns an int value with at most a single one-bit, in the position of the lowest-order ("rightmost") one-bit in the specified int value.

Returns zero if the specified value has no one-bits in its two's complement binary representation, that is, if it is equal to zero.

Parameters

<i>value</i>	- the int to be inspected
--------------	---------------------------

Returns

an int value with a single one-bit, in the position of the lowest-order one-bit in the specified value, or zero if the specified value is itself equal to zero.

6.274.2.14 static int decaf::lang::Integer::numberOfLeadingZeros (int *value*)
[static]

Returns the number of zero bits preceding the highest-order ("leftmost") one-bit in the two's complement binary representation of the specified int value.

Returns 32 if the specified value has no one-bits in its two's complement representation, in other words if it is equal to zero.

Note that this method is closely related to the logarithm base 2. For all positive int values *x*:

$\text{* floor(log2(x)) = 31 - numberOfLeadingZeros(x) * ceil(log2(x)) = 32 - numberOfLeadingZeros(x - 1)}$

Parameters

<i>value</i>	- the int to be inspected
--------------	---------------------------

Returns

the number of zero bits preceding the highest-order ("leftmost") one-bit in the two's complement binary representation of the specified int value, or 32 if the value is equal to zero.

6.274.2.15 static int decaf::lang::Integer::numberOfTrailingZeros (int *value*)
[static]

Returns the number of zero bits following the lowest-order ("rightmost") one-bit in the two's complement binary representation of the specified int value.

Returns 32 if the specified value has no one-bits in its two's complement representation, in other words if it is equal to zero.

Parameters

<i>value</i>	- the int to be inspected
--------------	---------------------------

Returns

the number of zero bits following the lowest-order ("rightmost") one-bit in the two's complement binary representation of the specified int value, or 32 if the value is equal to zero.

6.274.2.16 `virtual bool decaf::lang::Integer::operator< (const Integer & i) const`
`[inline, virtual]`

Compares this object to another and returns true if this object is considered to be less than the one passed.

This

Parameters

<code>i</code>	- the value to be compared to this one.
----------------	---

Returns

true if this object is equal to the one passed.

Implements **decaf::lang::Comparable< Integer >** (p. 887).

6.274.2.17 `virtual bool decaf::lang::Integer::operator< (const int & i) const` `[inline, virtual]`

Compares this object to another and returns true if this object is considered to be less than the one passed.

This

Parameters

<code>i</code>	- the value to be compared to this one.
----------------	---

Returns

true if this object is equal to the one passed.

Implements **decaf::lang::Comparable< int >** (p. 887).

6.274.2.18 `virtual bool decaf::lang::Integer::operator== (const Integer & i) const`
`[inline, virtual]`

Compares equality between this object and the one passed.

Parameters

<code>i</code>	- the value to be compared to this one.
----------------	---

Returns

true if this object is equal to the one passed.

Implements **decaf::lang::Comparable< Integer >** (p. 888).

6.274.2.19 `virtual bool decaf::lang::Integer::operator==(const int & i) const` `[inline, virtual]`

Compares equality between this object and the one passed.

Parameters

<i>i</i>	- the value to be compared to this one.
----------	---

Returns

true if this object is equal to the one passed.

Implements **decaf::lang::Comparable**< int > (p. 888).

6.274.2.20 `static int decaf::lang::Integer::parseInt (const std::string & s, int radix)` `[static]`

Parses the string argument as a signed int in the radix specified by the second argument.

The characters in the string must all be digits, of the specified radix (as determined by whether **Character.digit(char, int)** (p. 768) returns a nonnegative value) except that the first character may be an ASCII minus sign '-' to indicate a negative value. The resulting byte value is returned.

An exception of type `NumberFormatException` is thrown if any of the following situations occurs: * The first argument is null or is a string of length zero. * The radix is either smaller than **Character.MIN_RADIX** (p. 773) or larger than **Character.MAX_RADIX** (p. 772). * Any character of the string is not a digit of the specified radix, except that the first character may be a minus sign '-' provided that the string is longer than length 1. * The value represented by the string is not a value of type int.

Parameters

<i>s</i>	- the String (p. 2620) containing the int representation to be parsed
<i>radix</i>	- the radix to be used while parsing s

Returns

the int represented by the string argument in the specified radix.

Exceptions

<i>NumberFormatException</i>	- If String (p. 2620) does not contain a parsable int.
------------------------------	---

6.274.2.21 static int decaf::lang::Integer::parseInt (const std::string & s) [static]

Parses the string argument as a signed decimal int.

The characters in the string must all be decimal digits, except that the first character may be an ASCII minus sign '-' to indicate a negative value. The resulting int value is returned, exactly as if the argument and the radix 10 were given as arguments to the parseInt(const std::string, int) method.

Parameters

<i>s</i>	- String (p. 2620) to convert to a int
----------	---

Returns

the converted int value

Exceptions

<i>NumberFormatException</i>	if the string is not a int.
------------------------------	-----------------------------

6.274.2.22 static int decaf::lang::Integer::reverse (int value) [static]

Returns the value obtained by reversing the order of the bits in the two's complement binary representation of the specified int value.

Parameters

<i>value</i>	- the value whose bits are to be reversed
--------------	---

Returns

the reversed bits int.

6.274.2.23 static int decaf::lang::Integer::reverseBytes (int value) [static]

Returns the value obtained by reversing the order of the bytes in the two's complement representation of the specified int value.

Parameters

<i>value</i>	- the int whose bytes we are to reverse
--------------	---

Returns

the reversed int.

6.274.2.24 `static int decaf::lang::Integer::rotateLeft (int value, int distance)`
[static]

Returns the value obtained by rotating the two's complement binary representation of the specified int value left by the specified number of bits.

(Bits shifted out of the left hand, or high-order, side reenter on the right, or low-order.)

Note that left rotation with a negative distance is equivalent to right rotation: `rotateLeft(val, -distance) == rotateRight(val, distance)`. Note also that rotation by any multiple of 32 is a no-op, so all but the last five bits of the rotation distance can be ignored, even if the distance is negative: `rotateLeft(val, distance) == rotateLeft(val, distance & 0x1F)`.

Parameters

<i>value</i>	- the int to be inspected
<i>distance</i>	- the number of bits to rotate

Returns

the value obtained by rotating the two's complement binary representation of the specified int value left by the specified number of bits.

6.274.2.25 `static int decaf::lang::Integer::rotateRight (int value, int distance)`
[static]

Returns the value obtained by rotating the two's complement binary representation of the specified int value right by the specified number of bits.

(Bits shifted out of the right hand, or low-order, side reenter on the left, or high-order.)

Note that right rotation with a negative distance is equivalent to left rotation: `rotateRight(val, -distance) == rotateLeft(val, distance)`. Note also that rotation by any multiple of 32 is a no-op, so all but the last five bits of the rotation distance can be ignored, even if the distance is negative: `rotateRight(val, distance) == rotateRight(val, distance & 0x1F)`.

Parameters

<i>value</i>	- the int to be inspected
<i>distance</i>	- the number of bits to rotate

Returns

the value obtained by rotating the two's complement binary representation of the specified int value right by the specified number of bits.

6.274.2.26 `virtual short decaf::lang::Integer::shortValue () const [inline, virtual]`

Answers the short value which the receiver represents.

Returns

int the value of the receiver.

Reimplemented from **decaf::lang::Number** (p. 1994).

6.274.2.27 `static int decaf::lang::Integer::signum (int value) [static]`

Returns the signum function of the specified int value.

(The return value is -1 if the specified value is negative; 0 if the specified value is zero; and 1 if the specified value is positive.)

Parameters

<i>value</i>	- the int to be inspected
--------------	---------------------------

Returns

the signum function of the specified int value.

6.274.2.28 `static std::string decaf::lang::Integer::toBinaryString (int value) [static]`

Returns a string representation of the integer argument as an unsigned integer in base 2.

The unsigned integer value is the argument plus 2^{32} if the argument is negative; otherwise it is equal to the argument. This value is converted to a string of ASCII digits in binary (base 2) with no extra leading 0s. If the unsigned magnitude is zero, it is represented by a single zero character '0'; otherwise, the first character of the representation of the unsigned magnitude will not be the zero character.

The characters '0' and '1' are used as binary digits.

Parameters

<i>value</i>	- the int to be translated to a binary string
--------------	---

Returns

the unsigned int value as a binary string

6.274.2.29 static std::string decaf::lang::Integer::toHexString (int *value*) [static]

Returns a string representation of the integer argument as an unsigned integer in base 16.

The unsigned integer value is the argument plus 2^{32} if the argument is negative; otherwise, it is equal to the argument. This value is converted to a string of ASCII digits in hexadecimal (base 16) with no extra leading 0s. If the unsigned magnitude is zero, it is represented by a single zero character '0'; otherwise, the first character of the representation of the unsigned magnitude will not be the zero character. The following characters are used as hexadecimal digits:

0123456789abcdef

If uppercase letters are desired, the toUpperCase() method may be called on the result:

Parameters

<i>value</i>	- the int to be translated to an Octal string
--------------	---

Returns

the unsigned int value as a Octal string

6.274.2.30 static std::string decaf::lang::Integer::toOctalString (int *value*) [static]

Returns a string representation of the integer argument as an unsigned integer in base 8.

The unsigned integer value is the argument plus 2^{32} if the argument is negative; otherwise, it is equal to the argument. This value is converted to a string of ASCII digits in octal (base 8) with no extra leading 0s.

If the unsigned magnitude is zero, it is represented by a single zero character '0'; otherwise, the first character of the representation of the unsigned magnitude will not be the zero character. The following characters are used as octal digits:

01234567

Parameters

<i>value</i>	- the int to be translated to an Octal string
--------------	---

Returns

the unsigned int value as a Octal string

6.274.2.31 `std::string decaf::lang::Integer::toString () const`**Returns**

this **Integer** (p. 1500) Object as a **String** (p. 2620) Representation

Referenced by `decaf::util::ArrayList< E >::toString()`.

6.274.2.32 `static std::string decaf::lang::Integer::toString (int value) [static]`

Converts the int to a **String** (p. 2620) representation.

Parameters

<i>value</i>	The int to convert to a <code>std::string</code> instance.
--------------	--

Returns

string representation

6.274.2.33 `static std::string decaf::lang::Integer::toString (int value, int radix) [static]`

Returns a string representation of the first argument in the radix specified by the second argument.

If the radix is smaller than **Character.MIN_RADIX** (p. 773) or larger than **Character.MAX_RADIX** (p. 772), then the radix 10 is used instead.

If the first argument is negative, the first element of the result is the ASCII minus character '-'. If the first argument is not negative, no sign character appears in the result.

The remaining characters of the result represent the magnitude of the first argument. If the magnitude is zero, it is represented by a single zero character '0'; otherwise, the first character of the representation of the magnitude will not be the zero character. The following ASCII characters are used as digits:

0123456789abcdefghijklmnopqrstuvwxyz

Parameters

<i>value</i>	- the int to convert to a string
<i>radix</i>	- the radix to format the string in

Returns

an int formatted to the string value of the radix given.

6.274.2.34 `static Integer decaf::lang::Integer::valueOf (int value) [inline, static]`

Returns a **Integer** (p. 1500) instance representing the specified int value.

Parameters

<i>value</i>	- the int to wrap
--------------	-------------------

Returns

the new **Integer** (p. 1500) object wrapping value.

6.274.2.35 `static Integer decaf::lang::Integer::valueOf (const std::string & value) [static]`

Returns a **Integer** (p. 1500) object holding the value given by the specified std::string.

The argument is interpreted as representing a signed decimal int, exactly as if the argument were given to the `parseInt(std::string)` method. The result is a **Integer** (p. 1500) object that represents the int value specified by the string.

Parameters

<i>value</i>	- std::string to parse as base 10
--------------	-----------------------------------

Returns

new **Integer** (p. 1500) Object wrapping the primitive

Exceptions

<i>NumberFormatException</i>	if the string is not a decimal int.
------------------------------	-------------------------------------

6.274.2.36 `static Integer decaf::lang::Integer::valueOf (const std::string & value, int radix) [static]`

Returns a **Integer** (p. 1500) object holding the value extracted from the specified std::string when parsed with the radix given by the second argument.

The first argument is interpreted as representing a signed int in the radix specified by the second argument, exactly as if the argument were given to the `parseInt(std::string, int`

) method. The result is a **Integer** (p. 1500) object that represents the int value specified by the string.

Parameters

<i>value</i>	- std::string to parse as base (radix)
<i>radix</i>	- base of the string to parse.

Returns

new **Integer** (p. 1500) Object wrapping the primitive

Exceptions

<i>NumberFormatException</i>	if the string is not a valid int.
------------------------------	-----------------------------------

6.274.3 Field Documentation

6.274.3.1 `const int decaf::lang::Integer::MAX_VALUE = (int)0x7FFFFFFF` [static]

The maximum value that the primitive type can hold.

Referenced by decaf::util::concurrent::LinkedBlockingQueue< E >::drainTo().

6.274.3.2 `const int decaf::lang::Integer::MIN_VALUE = (int)0x80000000` [static]

The minimum value that the primitive type can hold.

6.274.3.3 `const int decaf::lang::Integer::SIZE = 32` [static]

The size in bits of the primitive int type.

The documentation for this class was generated from the following file:

- src/main/decaf/lang/**Integer.h**

6.275 activemq::commands::IntegerResponse Class Reference

```
#include <src/main/activemq/commands/IntegerResponse.h>
```

Inheritance diagram for activemq::commands::IntegerResponse:

Public Member Functions

- **IntegerResponse** ()
- virtual **~IntegerResponse** ()
- virtual unsigned char **getDataStructureType** () const
*Get the **DataSetructure** (p. 1133) Type as defined in CommandTypes.h.*
- virtual **IntegerResponse** * **cloneDataSetructure** () const
Clone this obbject and return a new instance that the caller now owns, this will be an exact copy of this one.
- virtual void **copyDataSetructure** (const **DataSetructure** *src)
Copy the contents of the passed object into this objects members, overwriting any existing data.
- virtual std::string **toString** () const
*Returns a string containing the information for this **DataSetructure** (p. 1133) such as its type and value of its elements.*
- virtual bool **equals** (const **DataSetructure** *value) const
*Compares the **DataSetructure** (p. 1133) passed in to this one, and returns if they are equivalent.*
- virtual int **getResult** () const
- virtual void **setResult** (int result)

Static Public Attributes

- static const unsigned char **ID_INTEGERRESPONSE** = 34

Protected Attributes

- int **result**

6.275.1 Constructor & Destructor Documentation

6.275.1.1 **activemq::commands::IntegerResponse::IntegerResponse** ()

6.275.1.2 **virtual activemq::commands::IntegerResponse::~~IntegerResponse** ()
 [virtual]

6.275.2 Member Function Documentation

6.275.2.1 **virtual IntegerResponse*** **activemq::commands::IntegerResponse::cloneDataSetructure** () const
 [virtual]

Clone this obbject and return a new instance that the caller now owns, this will be an exact copy of this one.

Returns

new copy of this object.

Reimplemented from **activemq::commands::Response** (p. 2299).

6.275.2.2 `virtual void activemq::commands::IntegerResponse::copyDataStructure (const DataStructure * src) [virtual]`

Copy the contents of the passed object into this objects members, overwriting any existing data.

Returns

src - Source Object

Reimplemented from **activemq::commands::Response** (p. 2299).

6.275.2.3 `virtual bool activemq::commands::IntegerResponse::equals (const DataStructure * value) const [virtual]`

Compares the **DataStructure** (p. 1133) passed in to this one, and returns if they are equivalent.

Equivalent here means that they are of the same type, and that each element of the objects are the same.

Returns

true if **DataStructure** (p. 1133)'s are Equal.

Reimplemented from **activemq::commands::Response** (p. 2299).

6.275.2.4 `virtual unsigned char activemq::commands::IntegerResponse::getDataStructureType () const [virtual]`

Get the **DataStructure** (p. 1133) Type as defined in CommandTypes.h.

Returns

The type of the data structure

Reimplemented from **activemq::commands::Response** (p. 2300).

6.275.2.5 `virtual int activemq::commands::IntegerResponse::getResult () const [virtual]`

6.276 activemq::wireformat::openwire::marshal::generated::IntegerResponseMarshaller Class

Reference

1523

6.275.2.6 `virtual void activemq::commands::IntegerResponse::setResult (int result)`
[virtual]

6.275.2.7 `virtual std::string activemq::commands::IntegerResponse::toString ()`
`const` [virtual]

Returns a string containing the information for this **DataSet** (p. 1133) such as its type and value of its elements.

Returns

formatted string useful for debugging.

Reimplemented from **activemq::commands::Response** (p. 2300).

6.275.3 Field Documentation

6.275.3.1 `const unsigned char activemq::commands::IntegerResponse::ID_INTEGERRESPONSE = 34` [static]

6.275.3.2 `int activemq::commands::IntegerResponse::result` [protected]

The documentation for this class was generated from the following file:

- `src/main/activemq/commands/IntegerResponse.h`

6.276 activemq::wireformat::openwire::marshal::generated::IntegerResponseMarshaller Class Reference

Marshaling code for Open Wire Format for **IntegerResponseMarshaller** (p. 1519).

```
#include <src/main/activemq/wireformat/openwire/marshal/generated/-  
IntegerResponseMarshaller.h>
```

Inheritance diagram for `activemq::wireformat::openwire::marshal::generated::IntegerResponseMarshaller`:

Public Member Functions

- **IntegerResponseMarshaller** ()
- `virtual ~IntegerResponseMarshaller` ()
- `virtual commands::DataSet * createObject` () const
Creates a new instance of the class that this class is a marshaling director for.
- `virtual unsigned char getDataStructureType` () const

Gets the DataStructureType that this class marshals/unmarshals.

- virtual void **tightUnmarshal** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataInputStream** *dataIn, **utils::BooleanStream** *bs)

Tight Un-marhsal to the given stream.

- virtual int **tightMarshal1** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **utils::BooleanStream** *bs)

Tight Marhsal to the given stream.

- virtual void **tightMarshal2** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataOutputStream** *dataOut, **utils::BooleanStream** *bs)

Tight Marhsal to the given stream.

- virtual void **looseUnmarshal** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataInputStream** *dataIn)

Loose Un-marhsal to the given stream.

- virtual void **looseMarshal** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataOutputStream** *dataOut)

Tight Marhsal to the given stream.

6.276.1 Detailed Description

Marshaling code for Open Wire Format for **IntegerResponseMarshaller** (p. 1519).

NOTE!: This file is auto generated - do not modify! if you need to make a change, please see the Java Classes in the activemq-openwire-generator module

6.276.2 Constructor & Destructor Documentation

6.276.2.1 **activemq::wireformat::openwire::marshal::generated::IntegerResponseMarshaller::IntegerResponseMarshaller ()**
[inline]

6.276.2.2 **virtual activemq::wireformat::openwire::marshal::generated::IntegerResponseMarshaller::~~IntegerResponseMarshaller ()** [inline, virtual]

6.276.3 Member Function Documentation

6.276.3.1 **virtual commands::DataStructure* activemq::wireformat::openwire::marshal::generated::IntegerResponseMarshaller::createObject ()**
const [virtual]

Creates a new instance of the class that this class is a marshaling director for.

6.276 `activemq::wireformat::openwire::marshal::generated::IntegerResponseMarshaller` Class

Reference

1525

Returns

newly allocated Command

Reimplemented from `activemq::wireformat::openwire::marshal::generated::ResponseMarshaller` (p. 2308).

6.276.3.2 `virtual unsigned char activemq::wireformat::openwire::marshal::generated::IntegerResponseMarshaller::getDataStructureType () const`
[virtual]

Gets the `DataStreamType` that this class marshals/unmarshals.

Returns

byte Id of this classes `DataStreamType`

Reimplemented from `activemq::wireformat::openwire::marshal::generated::ResponseMarshaller` (p. 2309).

6.276.3.3 `virtual void activemq::wireformat::openwire::marshal::generated::IntegerResponseMarshaller::looseMarshal (OpenWireFormat * format, commands::DataStream * command, decaf::io::DataOutputStream * ds)` [virtual]

Tight Marshal to the given stream.

Parameters

<i>format</i>	- The <code>OpenWireFormat</code> properties
<i>command</i>	- the object to Marshal
<i>ds</i>	- <code>DataOutputStream</code> to marshal to

Exceptions

<i>IOException</i>	if an error occurs.
--------------------	---------------------

Reimplemented from `activemq::wireformat::openwire::marshal::generated::ResponseMarshaller` (p. 2309).

6.276.3.4 `virtual void activemq::wireformat::openwire::marshal::generated::IntegerResponseMarshaller::looseUnmarshal (OpenWireFormat * format, commands::DataStream * command, decaf::io::DataInputStream * dis)` [virtual]

Loose Un-marshal to the given stream.

Parameters

<i>format</i>	- The OpenwireFormat properties
<i>command</i>	- the object to Un-Marshal
<i>dis</i>	- the DataInputStream to Un-Marshal from

Exceptions

<i>IOException</i>	if an error occurs.
--------------------	---------------------

Reimplemented from **activemq::wireformat::openwire::marshal::generated::ResponseMarshaller** (p. 2310).

6.276.3.5 `virtual int activemq::wireformat::openwire::marshal::generated::IntegerResponseMarshaller::tightMarshal1 (OpenWireFormat * format, commands::DataStructure * command, utils::BooleanStream * bs)`
`[virtual]`

Tight Marshal to the given stream.

Parameters

<i>format</i>	The OpenwireFormat properties
<i>command</i>	The object to Marshal
<i>bs</i>	The boolean stream to marshal to.

Exceptions

<i>IOException</i>	if an error occurs.
--------------------	---------------------

Reimplemented from **activemq::wireformat::openwire::marshal::generated::ResponseMarshaller** (p. 2310).

6.276.3.6 `virtual void activemq::wireformat::openwire::marshal::generated::IntegerResponseMarshaller::tightMarshal2 (OpenWireFormat * format, commands::DataStructure * command, decaf::io::DataOutputStream * ds, utils::BooleanStream * bs)`
`[virtual]`

Tight Marshal to the given stream.

Parameters

<i>format</i>	- The OpenwireFormat properties
<i>command</i>	- the object to Marshal
<i>ds</i>	- the DataOutputStream to Marshal to
<i>bs</i>	- boolean stream to marshal to.

Exceptions

<i>IOException</i>	if an error occurs.
--------------------	---------------------

Reimplemented from **activemq::wireformat::openwire::marshal::generated::ResponseMarshaller** (p. 2311).

6.276.3.7 virtual void **activemq::wireformat::openwire::marshal::generated::IntegerResponseMarshaller::tightUnmarshal** (**OpenWireFormat** * *format*, **commands::DataStructure** * *command*, **decaf::io::DataInputStream** * *dis*, **utils::BooleanStream** * *bs*) [virtual]

Tight Un-marhsal to the given stream.

Parameters

<i>format</i>	- The OpenWireFormat properties
<i>command</i>	- the object to Un-Marshal
<i>dis</i>	- the DataInputStream to Un-Marshal from
<i>bs</i>	- boolean stream to unmarshal from.

Exceptions

<i>IOException</i>	if an error occurs.
--------------------	---------------------

Reimplemented from **activemq::wireformat::openwire::marshal::generated::ResponseMarshaller** (p. 2311).

The documentation for this class was generated from the following file:

- **src/main/activemq/wireformat/openwire/marshal/generated/IntegerResponseMarshaller.h**

6.277 internal_state Struct Reference

```
#include <src/main/decaf/internal/util/zip/deflate.h>
```

Data Fields

- **z_stream** strm
- int **status**
- **Bytef** * **pending_buf**
- **ulg** **pending_buf_size**
- **Bytef** * **pending_out**
- **uInt** **pending**
- int **wrap**

- **gz_headerp** gzhead
- **ulnt** gzindex
- **Byte** method
- **int** last_flush
- **ulnt** w_size
- **ulnt** w_bits
- **ulnt** w_mask
- **Bytef** * window
- **ulg** window_size
- **Posf** * prev
- **Posf** * head
- **ulnt** ins_h
- **ulnt** hash_size
- **ulnt** hash_bits
- **ulnt** hash_mask
- **ulnt** hash_shift
- **long** block_start
- **ulnt** match_length
- **lPos** prev_match
- **int** match_available
- **ulnt** strstart
- **ulnt** match_start
- **ulnt** lookahead
- **ulnt** prev_length
- **ulnt** max_chain_length
- **ulnt** max_lazy_match
- **int** level
- **int** strategy
- **ulnt** good_match
- **int** nice_match
- **struct** **ct_data_s** dyn_ltree [HEAP_SIZE]
- **struct** **ct_data_s** dyn_dtree [2 *D_CODES+1]
- **struct** **ct_data_s** bl_tree [2 *BL_CODES+1]
- **struct** **tree_desc_s** l_desc
- **struct** **tree_desc_s** d_desc
- **struct** **tree_desc_s** bl_desc
- **ush** bl_count [MAX_BITS+1]
- **int** heap [2 *L_CODES+1]
- **int** heap_len
- **int** heap_max
- **uch** depth [2 *L_CODES+1]
- **uchf** * l_buf
- **ulnt** lit_bufsize
- **ulnt** last_lit
- **ushf** * d_buf
- **ulg** opt_len

- **ulg static_len**
- **ulnt matches**
- **int last_eob_len**
- **ush bi_buf**
- **int bi_valid**
- **ulg high_water**
- **int dummy**

6.277.1 Field Documentation

- 6.277.1.1 **ush** internal_state::bi_buf
- 6.277.1.2 **int** internal_state::bi_valid
- 6.277.1.3 **ush** internal_state::bl_count[MAX_BITS+1]
- 6.277.1.4 **struct tree_desc_s** internal_state::bl_desc
- 6.277.1.5 **struct ct_data_s** internal_state::bl_tree[2 * BL_CODES+1]
- 6.277.1.6 **long** internal_state::block_start
- 6.277.1.7 **ushf*** internal_state::d_buf
- 6.277.1.8 **struct tree_desc_s** internal_state::d_desc
- 6.277.1.9 **uch** internal_state::depth[2 * L_CODES+1]
- 6.277.1.10 **int** internal_state::dummy
- 6.277.1.11 **struct ct_data_s** internal_state::dyn_dtree[2 * D_CODES+1]
- 6.277.1.12 **struct ct_data_s** internal_state::dyn_ltree[HEAP_SIZE]
- 6.277.1.13 **ulnt** internal_state::good_match
- 6.277.1.14 **gz_headerp** internal_state::gzhead
- 6.277.1.15 **ulnt** internal_state::gzindex
- 6.277.1.16 **ulnt** internal_state::hash_bits
- 6.277.1.17 **ulnt** internal_state::hash_mask
- 6.277.1.18 **ulnt** internal_state::hash_shift
- 6.277.1.19 **ulnt** internal_state::hash_size

- 6.277.1.20 **Posf*** internal_state::head
- 6.277.1.21 **int** internal_state::heap[2 *L_CODES+1]
- 6.277.1.22 **int** internal_state::heap_len
- 6.277.1.23 **int** internal_state::heap_max
- 6.277.1.24 **ulg** internal_state::high_water
- 6.277.1.25 **ulnt** internal_state::ins_h
- 6.277.1.26 **uchf*** internal_state::l_buf
- 6.277.1.27 **struct tree_desc_s** internal_state::l_desc
- 6.277.1.28 **int** internal_state::last_eob_len
- 6.277.1.29 **int** internal_state::last_flush
- 6.277.1.30 **ulnt** internal_state::last_lit
- 6.277.1.31 **int** internal_state::level
- 6.277.1.32 **ulnt** internal_state::lit_bufsize
- 6.277.1.33 **ulnt** internal_state::lookahead
- 6.277.1.34 **int** internal_state::match_available
- 6.277.1.35 **ulnt** internal_state::match_length
- 6.277.1.36 **ulnt** internal_state::match_start
- 6.277.1.37 **ulnt** internal_state::matches
- 6.277.1.38 **ulnt** internal_state::max_chain_length
- 6.277.1.39 **ulnt** internal_state::max_lazy_match
- 6.277.1.40 **Byte** internal_state::method
- 6.277.1.41 **int** internal_state::nice_match
- 6.277.1.42 **ulg** internal_state::opt_len
- 6.277.1.43 **ulnt** internal_state::pending

6.278 activemq::transport::mock::InternalCommandListener Class Reference 531

- 6.277.1.44 `Bytef* internal_state::pending_buf`
- 6.277.1.45 `ulg internal_state::pending_buf_size`
- 6.277.1.46 `Bytef* internal_state::pending_out`
- 6.277.1.47 `Posf* internal_state::prev`
- 6.277.1.48 `ulnt internal_state::prev_length`
- 6.277.1.49 `IPos internal_state::prev_match`
- 6.277.1.50 `ulg internal_state::static_len`
- 6.277.1.51 `int internal_state::status`
- 6.277.1.52 `int internal_state::strategy`
- 6.277.1.53 `z_stream* internal_state::strm`
- 6.277.1.54 `ulnt internal_state::strstart`
- 6.277.1.55 `ulnt internal_state::w_bits`
- 6.277.1.56 `ulnt internal_state::w_mask`
- 6.277.1.57 `ulnt internal_state::w_size`
- 6.277.1.58 `Bytef* internal_state::window`
- 6.277.1.59 `ulg internal_state::window_size`
- 6.277.1.60 `int internal_state::wrap`

The documentation for this struct was generated from the following files:

- `src/main/decaf/internal/util/zip/deflate.h`
- `src/main/decaf/internal/util/zip/zlib.h`

6.278 activemq::transport::mock::InternalCommandListener Class Reference

Listens for Commands sent from the **MockTransport** (p. 1948).

```
#include <src/main/activemq/transport/mock/InternalCommand-  
Listener.h>
```

Inheritance diagram for `activemq::transport::mock::InternalCommandListener`:

Public Member Functions

- **InternalCommandListener** ()
- virtual **~InternalCommandListener** ()
- void **setTransport** (**MockTransport** *transport)
- void **setResponseBuilder** (const **Pointer**< **ResponseBuilder** > &responseBuilder)
- virtual void **onCommand** (const **Pointer**< **Command** > &command)
Event handler for the receipt of a command.
- void **run** ()
Default implementation of the run method - does nothing.

6.278.1 Detailed Description

Listens for Commands sent from the **MockTransport** (p. 1948).

This class processes all outbound commands and sends responses that are constructed by calling the Protocol provided **ResponseBuilder** (p. 2301) and getting a set of Commands to send back into the **MockTransport** (p. 1948) as incoming Commands and Responses.

6.278.2 Constructor & Destructor Documentation

6.278.2.1 `activemq::transport::mock::InternalCommandListener::InternalCommandListener ()`

6.278.2.2 `virtual activemq::transport::mock::InternalCommandListener::~~InternalCommandListener () [virtual]`

6.278.3 Member Function Documentation

6.278.3.1 `virtual void activemq::transport::mock::InternalCommandListener::onCommand (const Pointer< Command > &command) [virtual]`

Event handler for the receipt of a command.

The transport passes off all received commands to its listeners, the listener then owns the Object. If there is no registered listener the **Transport** (p. 2790) deletes the command upon receipt.

Parameters

<i>command</i>	The received command object.
----------------	------------------------------

Implements **activemq::transport::TransportListener** (p. 2811).

6.278.3.2 void **activemq::transport::mock::InternalCommandListener::run** ()
[virtual]

Default implementation of the run method - does nothing.

Reimplemented from **decaf::lang::Thread** (p. 2710).

6.278.3.3 void **activemq::transport::mock::InternalCommandListener::set-ResponseBuilder** (const Pointer< ResponseBuilder > & responseBuilder)
[inline]

6.278.3.4 void **activemq::transport::mock::InternalCommand-Listener::setTransport** (MockTransport * transport)
[inline]

The documentation for this class was generated from the following file:

- src/main/activemq/transport/mock/**InternalCommandListener.h**

6.279 decaf::lang::exceptions::InterruptedException Class - Reference

```
#include <src/main/decaf/lang/exceptions/InterruptedException.h>
```

Inheritance diagram for decaf::lang::exceptions::InterruptedException:

Public Member Functions

- **InterruptedException** () throw ()
Default Constructor.
- **InterruptedException** (const **Exception** &ex) throw ()
*Conversion Constructor from some other **Exception** (p. 1279).*
- **InterruptedException** (const **InterruptedException** &ex) throw ()
Copy Constructor.
- **InterruptedException** (const char *file, const int lineNumber, const std::exception *cause, const char *msg,...) throw ()
Constructor - Initializes the file name and line number where this message occurred.
- **InterruptedException** (const std::exception *cause) throw ()
Constructor.

- **InterruptedException** (const char *file, const int lineNumber, const char *msg,...) throw ()
Constructor - Initializes the file name and line number where this message occurred.
- virtual **InterruptedException** * **clone** () const
Clones this exception.
- virtual ~**InterruptedException** () throw ()

6.279.1 Constructor & Destructor Documentation

6.279.1.1 **decaf::lang::exceptions::InterruptedException::InterruptedException () throw ()** `[inline]`

Default Constructor.

6.279.1.2 **decaf::lang::exceptions::InterruptedException::InterruptedException (const Exception & ex) throw ()** `[inline]`

Conversion Constructor from some other **Exception** (p. 1279).

Parameters

ex	The Exception (p. 1279) whose data is to be copied into this one.
-----------	--

6.279.1.3 **decaf::lang::exceptions::InterruptedException::InterruptedException (const InterruptedException & ex) throw ()** `[inline]`

Copy Constructor.

Parameters

ex	The Exception (p. 1279) whose data is to be copied into this one.
-----------	--

6.279.1.4 **decaf::lang::exceptions::InterruptedException::InterruptedException (const char * file, const int lineNumber, const std::exception * cause, const char * msg, ...) throw ()** `[inline]`

Constructor - Initializes the file name and line number where this message occurred.

Sets the message to report, using an optional list of arguments to parse into the message

Parameters

<i>file</i>	The file name where exception occurs
<i>lineNumber</i>	The line number where the exception occurred.
<i>cause</i>	The exception that was the cause for this one to be thrown.

<i>msg</i>	The message to report
...	list of primitives that are formatted into the message

6.279.1.5 `decaf::lang::exceptions::InterruptedException::InterruptedException (const std::exception * cause) throw () [inline]`

Constructor.

Parameters

<i>cause</i>	Pointer (p. 2083) to the exception that caused this one to be thrown, the object is cloned caller retains ownership.
--------------	---

6.279.1.6 `decaf::lang::exceptions::InterruptedException::InterruptedException (const char * file, const int lineNumber, const char * msg, ...) throw () [inline]`

Constructor - Initializes the file name and line number where this message occurred.

Sets the message to report, using an optional list of arguments to parse into the message

Parameters

<i>file</i>	The file name where exception occurs
<i>lineNumber</i>	The line number where the exception occurred.
<i>msg</i>	The message to report
...	list of primitives that are formatted into the message

6.279.1.7 `virtual decaf::lang::exceptions::InterruptedException::~~InterruptedException () throw () [inline, virtual]`

6.279.2 Member Function Documentation

6.279.2.1 `virtual InterruptedException* decaf::lang::exceptions::InterruptedException::clone () const [inline, virtual]`

Clones this exception.

This is useful for cases where you need to preserve the type of the original exception as well as the message. All subclasses should override.

Returns

an new **Exception** (p. 1279) instance that is a copy of this one.

Reimplemented from **decaf::lang::Exception** (p. 1282).

The documentation for this class was generated from the following file:

- `src/main/decaf/lang/exceptions/InterruptedException.h`

6.280 decaf::io::InterruptedException Class Reference

```
#include <src/main/decaf/io/InterruptedException.h>
```

Inheritance diagram for `decaf::io::InterruptedException`:

Public Member Functions

- **InterruptedException** () throw ()
Default Constructor.
- **InterruptedException** (const **lang::Exception** &ex) throw ()
Copy Constructor.
- **InterruptedException** (const **InterruptedException** &ex) throw ()
Copy Constructor.
- **InterruptedException** (const char *file, const int lineNumber, const std::exception *cause, const char *msg,...) throw ()
Constructor - Initializes the file name and line number where this message occurred.
- **InterruptedException** (const std::exception *cause) throw ()
Constructor.
- **InterruptedException** (const char *file, const int lineNumber, const char *msg,...) throw ()
Constructor.
- virtual **InterruptedException** * **clone** () const
Clones this exception.
- virtual ~**InterruptedException** () throw ()

6.280.1 Constructor & Destructor Documentation

6.280.1.1 **decaf::io::InterruptedException::InterruptedException** () throw ()
[inline]

Default Constructor.

6.280.1.2 **decaf::io::InterruptedIOException::InterruptedIOException (const lang::Exception & *ex*) throw ()** `[inline]`

Copy Constructor.

Parameters

<i>ex</i>	the exception to copy
-----------	-----------------------

6.280.1.3 **decaf::io::InterruptedIOException::InterruptedIOException (const InterruptedIOException & *ex*) throw ()** `[inline]`

Copy Constructor.

Parameters

<i>ex</i>	the exception to copy, which is an instance of this type
-----------	--

6.280.1.4 **decaf::io::InterruptedIOException::InterruptedIOException (const char * *file*, const int *lineNumber*, const std::exception * *cause*, const char * *msg*, ...) throw ()** `[inline]`

Constructor - Initializes the file name and line number where this message occurred.

Sets the message to report, using an optional list of arguments to parse into the message

Parameters

<i>file</i>	The file name where exception occurs
<i>lineNumber</i>	The line number where the exception occurred.
<i>cause</i>	The exception that was the cause for this one to be thrown.
<i>msg</i>	The message to report
...	list of primitives that are formatted into the message

6.280.1.5 **decaf::io::InterruptedIOException::InterruptedIOException (const std::exception * *cause*) throw ()** `[inline]`

Constructor.

Parameters

<i>cause</i>	Pointer to the exception that caused this one to be thrown, the object is cloned caller retains ownership.
--------------	--

6.280.1.6 **decaf::io::InterruptedIOException::InterruptedIOException** (const char * *file*, const int *lineNumber*, const char * *msg*, ...) throw () [inline]

Constructor.

Parameters

<i>file</i>	The file name where exception occurs
<i>lineNumber</i>	The line number where the exception occurred.
<i>msg</i>	The message to report
...	list of primitives that are formatted into the message

6.280.1.7 **virtual decaf::io::InterruptedIOException::~~InterruptedIOException** () throw () [inline, virtual]

6.280.2 Member Function Documentation

6.280.2.1 **virtual InterruptedIOException* decaf::io::InterruptedIOException::clone** () const [inline, virtual]

Clones this exception.

This is useful for cases where you need to preserve the type of the original exception as well as the message. All subclasses should override.

Returns

a new exception that is a copy of this one.

Reimplemented from **decaf::io::IOException** (p. 1547).

Reimplemented in **decaf::net::SocketTimeoutException** (p. 2495).

The documentation for this class was generated from the following file:

- src/main/decaf/io/**InterruptedIOException.h**

6.281 cms::InvalidClientIdException Class Reference

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

```
#include <src/main/cms/InvalidClientIdException.h>
```

Inheritance diagram for cms::InvalidClientIdException:

Public Member Functions

- **InvalidClientIdException** ()
- **InvalidClientIdException** (const **InvalidClientIdException** &ex)
- **InvalidClientIdException** (const std::string &message)
- **InvalidClientIdException** (const std::string &message, const std::exception *cause)
- **InvalidClientIdException** (const std::string &message, const std::exception *cause, const std::vector< std::pair< std::string, int > > &stackTrace)
- virtual ~**InvalidClientIdException** () throw ()

6.281.1 Detailed Description

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

Since

1.3

6.281.2 Constructor & Destructor Documentation

- 6.281.2.1 **cms::InvalidClientIdException::InvalidClientIdException** ()
- 6.281.2.2 **cms::InvalidClientIdException::InvalidClientIdException** (const **InvalidClientIdException** & ex)
- 6.281.2.3 **cms::InvalidClientIdException::InvalidClientIdException** (const std::string & message)
- 6.281.2.4 **cms::InvalidClientIdException::InvalidClientIdException** (const std::string & message, const std::exception * cause)
- 6.281.2.5 **cms::InvalidClientIdException::InvalidClientIdException** (const std::string & message, const std::exception * cause, const std::vector< std::pair< std::string, int > > & stackTrace)
- 6.281.2.6 virtual **cms::InvalidClientIdException::~~InvalidClientIdException** () throw () [virtual]

The documentation for this class was generated from the following file:

- src/main/cms/**InvalidClientIdException.h**

6.282 cms::InvalidDestinationException Class Reference

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

```
#include <src/main/cms/InvalidDestinationException.h>
```

Inheritance diagram for cms::InvalidDestinationException:

Public Member Functions

- **InvalidDestinationException** ()
- **InvalidDestinationException** (const **InvalidDestinationException** &ex)
- **InvalidDestinationException** (const std::string &message)
- **InvalidDestinationException** (const std::string &message, const std::exception *cause)
- **InvalidDestinationException** (const std::string &message, const std::exception *cause, const std::vector< std::pair< std::string, int > > &stackTrace)
- virtual ~**InvalidDestinationException** () throw ()

6.282.1 Detailed Description

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

Since

1.3

6.282.2 Constructor & Destructor Documentation

- 6.282.2.1 **cms::InvalidDestinationException::InvalidDestinationException** ()
- 6.282.2.2 **cms::InvalidDestinationException::InvalidDestinationException** (const **InvalidDestinationException** & ex)
- 6.282.2.3 **cms::InvalidDestinationException::InvalidDestinationException** (const std::string & message)
- 6.282.2.4 **cms::InvalidDestinationException::InvalidDestinationException** (const std::string & message, const std::exception * cause)
- 6.282.2.5 **cms::InvalidDestinationException::InvalidDestinationException** (const std::string & message, const std::exception * cause, const std::vector< std::pair< std::string, int > > & stackTrace)

6.282.2.6 virtual cms::InvalidDestinationException::~InvalidDestinationException
() throw () [virtual]

The documentation for this class was generated from the following file:

- src/main/cms/InvalidDestinationException.h

6.283 decaf::security::InvalidKeyException Class Reference

```
#include <src/main/decaf/security/InvalidKeyException.h>
```

Inheritance diagram for decaf::security::InvalidKeyException:

Public Member Functions

- **InvalidKeyException** () throw ()
Default Constructor.
- **InvalidKeyException** (const **Exception** &ex) throw ()
Conversion Constructor from some other Exception.
- **InvalidKeyException** (const **InvalidKeyException** &ex) throw ()
Copy Constructor.
- **InvalidKeyException** (const char *file, const int lineNumber, const std::exception *cause, const char *msg,...) throw ()
Constructor - Initializes the file name and line number where this message occurred.
- **InvalidKeyException** (const std::exception *cause) throw ()
Constructor.
- **InvalidKeyException** (const char *file, const int lineNumber, const char *msg,...) throw ()
Constructor - Initializes the file name and line number where this message occurred.
- virtual **InvalidKeyException** * **clone** () const
Clones this exception.
- virtual ~**InvalidKeyException** () throw ()

6.283.1 Constructor & Destructor Documentation

6.283.1.1 decaf::security::InvalidKeyException::InvalidKeyException () throw ()
[inline]

Default Constructor.

6.283.1.2 **decaf::security::InvalidKeyException::InvalidKeyException (const Exception & ex) throw ()** `[inline]`

Conversion Constructor from some other Exception.

Parameters

<i>ex</i>	An exception that should become this type of Exception
-----------	--

6.283.1.3 **decaf::security::InvalidKeyException::InvalidKeyException (const InvalidKeyException & ex) throw ()** `[inline]`

Copy Constructor.

Parameters

<i>ex</i>	An exception that should become this type of Exception
-----------	--

6.283.1.4 **decaf::security::InvalidKeyException::InvalidKeyException (const char * file, const int lineNumber, const std::exception * cause, const char * msg, ...) throw ()** `[inline]`

Constructor - Initializes the file name and line number where this message occurred.

Sets the message to report, using an optional list of arguments to parse into the message

Parameters

<i>file</i>	The file name where exception occurs
<i>lineNumber</i>	The line number where the exception occurred.
<i>cause</i>	The exception that was the cause for this one to be thrown.
<i>msg</i>	The message to report
<i>...</i>	list of primitives that are formatted into the message

6.283.1.5 **decaf::security::InvalidKeyException::InvalidKeyException (const std::exception * cause) throw ()** `[inline]`

Constructor.

Parameters

<i>cause</i>	Pointer to the exception that caused this one to be thrown, the object is cloned caller retains ownership.
--------------	--

6.283.1.6 **decaf::security::InvalidKeyException::InvalidKeyException** (*const char * file*, *const int lineNumber*, *const char * msg*, ...) *throw ()* [inline]

Constructor - Initializes the file name and line number where this message occurred.

Sets the message to report, using an optional list of arguments to parse into the message

Parameters

<i>file</i>	name where exception occurs
<i>lineNumber</i>	line number where the exception occurred.
<i>msg</i>	message to report
...	list of primitives that are formatted into the message

6.283.1.7 **virtual decaf::security::InvalidKeyException::~~InvalidKeyException** () *throw ()* [inline, virtual]

6.283.2 Member Function Documentation

6.283.2.1 **virtual InvalidKeyException* decaf::security::InvalidKeyException::clone** () *const* [inline, virtual]

Clones this exception.

This is useful for cases where you need to preserve the type of the original exception as well as the message. All subclasses should override.

Returns

A deep copy of this exception.

Reimplemented from **decaf::security::KeyException** (p. 1603).

The documentation for this class was generated from the following file:

- `src/main/decaf/security/InvalidKeyException.h`

6.284 decaf::nio::InvalidMarkException Class Reference

```
#include <src/main/decaf/nio/InvalidMarkException.h>
```

Inheritance diagram for decaf::nio::InvalidMarkException:

Public Member Functions

- **InvalidMarkException** () *throw ()*

Default Constructor.

- **InvalidMarkException** (const lang::Exception &ex) throw ()

Conversion Constructor from some other Exception.

- **InvalidMarkException** (const InvalidMarkException &ex) throw ()

Copy Constructor.

- **InvalidMarkException** (const char *file, const int lineNumber, const std::exception *cause, const char *msg,...) throw ()

Constructor - Initializes the file name and line number where this message occurred.

- **InvalidMarkException** (const std::exception *cause) throw ()

Constructor.

- **InvalidMarkException** (const char *file, const int lineNumber, const char *msg,...) throw ()

Constructor - Initializes the file name and line number where this message occurred.

- virtual **InvalidMarkException** * clone () const

Clones this exception.

- virtual ~**InvalidMarkException** () throw ()

6.284.1 Constructor & Destructor Documentation

6.284.1.1 **decaf::nio::InvalidMarkException::InvalidMarkException () throw ()**
[inline]

Default Constructor.

6.284.1.2 **decaf::nio::InvalidMarkException::InvalidMarkException (const lang::Exception & ex) throw ()** [inline]

Conversion Constructor from some other Exception.

Parameters

ex	The Exception whose state data is to be copied into this Exception.
----	---

6.284.1.3 **decaf::nio::InvalidMarkException::InvalidMarkException (const InvalidMarkException & ex) throw ()** [inline]

Copy Constructor.

Parameters

ex	The Exception whose state data is to be copied into this Exception.
----	---

6.284.1.4 **decaf::nio::InvalidMarkException::InvalidMarkException** (const char * *file*, const int *lineNumber*, const std::exception * *cause*, const char * *msg*, ...)
throw () [inline]

Constructor - Initializes the file name and line number where this message occurred.

Sets the message to report, using an optional list of arguments to parse into the message

Parameters

<i>file</i>	The file name where exception occurs
<i>lineNumber</i>	The line number where the exception occurred.
<i>cause</i>	The exception that was the cause for this one to be thrown.
<i>msg</i>	The message to report
...	list of primitives that are formatted into the message

6.284.1.5 **decaf::nio::InvalidMarkException::InvalidMarkException** (const std::exception * *cause*) throw () [inline]

Constructor.

Parameters

<i>cause</i>	Pointer to the exception that caused this one to be thrown, the object is cloned caller retains ownership.
--------------	--

6.284.1.6 **decaf::nio::InvalidMarkException::InvalidMarkException** (const char * *file*, const int *lineNumber*, const char * *msg*, ...) throw () [inline]

Constructor - Initializes the file name and line number where this message occurred.

Sets the message to report, using an optional list of arguments to parse into the message

Parameters

<i>file</i>	The file name where exception occurs
<i>lineNumber</i>	The line number where the exception occurred.
<i>msg</i>	The message to report
...	list of primitives that are formatted into the message

6.284.1.7 **virtual decaf::nio::InvalidMarkException::~InvalidMarkException** ()
throw () [inline, virtual]

6.284.2 Member Function Documentation

6.284.2.1 `virtual InvalidMarkException* decaf::nio::InvalidMarkException::clone () const [inline, virtual]`

Clones this exception.

This is useful for cases where you need to preserve the type of the original exception as well as the message. All subclasses should override.

Returns

A new Exception instance that is a copy of this Exception.

Reimplemented from `decaf::lang::exceptions::IllegalStateException` (p. 1422).

The documentation for this class was generated from the following file:

- `src/main/decaf/nio/InvalidMarkException.h`

6.285 cms::InvalidSelectorException Class Reference

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

```
#include <src/main/cms/InvalidSelectorException.h>
```

Inheritance diagram for cms::InvalidSelectorException:

Public Member Functions

- `InvalidSelectorException ()`
- `InvalidSelectorException (const InvalidSelectorException &ex)`
- `InvalidSelectorException (const std::string &message)`
- `InvalidSelectorException (const std::string &message, const std::exception *cause)`
- `InvalidSelectorException (const std::string &message, const std::exception *cause, const std::vector< std::pair< std::string, int > > &stackTrace)`
- `virtual ~InvalidSelectorException () throw ()`

6.285.1 Detailed Description

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

Since

1.3

6.285.2 Constructor & Destructor Documentation

- 6.285.2.1 `cms::InvalidSelectorException::InvalidSelectorException ()`
- 6.285.2.2 `cms::InvalidSelectorException::InvalidSelectorException (const InvalidSelectorException & ex)`
- 6.285.2.3 `cms::InvalidSelectorException::InvalidSelectorException (const std::string & message)`
- 6.285.2.4 `cms::InvalidSelectorException::InvalidSelectorException (const std::string & message, const std::exception * cause)`
- 6.285.2.5 `cms::InvalidSelectorException::InvalidSelectorException (const std::string & message, const std::exception * cause, const std::vector< std::pair< std::string, int > > & stackTrace)`
- 6.285.2.6 `virtual cms::InvalidSelectorException::~~InvalidSelectorException () throw () [virtual]`

The documentation for this class was generated from the following file:

- `src/main/cms/InvalidSelectorException.h`

6.286 decaf::lang::exceptions::InvalidStateException Class Reference

```
#include <src/main/decaf/lang/exceptions/InvalidStateException.h>
```

Inheritance diagram for decaf::lang::exceptions::InvalidStateException:

Public Member Functions

- **InvalidStateException** () throw ()
Default Constructor.
- **InvalidStateException** (const **Exception** &ex) throw ()
*Conversion Constructor from some other **Exception** (p. 1279).*
- **InvalidStateException** (const **InvalidStateException** &ex) throw ()
Copy Constructor.
- **InvalidStateException** (const char *file, const int lineNumber, const std::exception *cause, const char *msg,...) throw ()
Constructor - Initializes the file name and line number where this message occurred.

- **InvalidStateException** (const std::exception ***cause**) throw ()
Constructor.
- **InvalidStateException** (const char *file, const int lineNumber, const char *msg,...) throw ()
Constructor - Initializes the file name and line number where this message occurred.
- virtual **InvalidStateException** * **clone** () const
Clones this exception.
- virtual ~**InvalidStateException** () throw ()

6.286.1 Constructor & Destructor Documentation

6.286.1.1 **decaf::lang::exceptions::InvalidStateException::InvalidStateException () throw ()** `[inline]`

Default Constructor.

6.286.1.2 **decaf::lang::exceptions::InvalidStateException::InvalidStateException (const Exception & ex) throw ()** `[inline]`

Conversion Constructor from some other **Exception** (p. 1279).

Parameters

ex	The Exception (p. 1279) whose data is to be copied into this one.
-----------	--

6.286.1.3 **decaf::lang::exceptions::InvalidStateException::InvalidStateException (const InvalidStateException & ex) throw ()** `[inline]`

Copy Constructor.

Parameters

ex	The Exception (p. 1279) whose data is to be copied into this one.
-----------	--

6.286.1.4 **decaf::lang::exceptions::InvalidStateException::InvalidStateException (const char * file, const int lineNumber, const std::exception * cause, const char * msg, ...) throw ()** `[inline]`

Constructor - Initializes the file name and line number where this message occurred.

Sets the message to report, using an optional list of arguments to parse into the message

Parameters

<i>file</i>	The file name where exception occurs
<i>lineNumber</i>	The line number where the exception occurred.
<i>cause</i>	The exception that was the cause for this one to be thrown.
<i>msg</i>	The message to report
...	list of primitives that are formatted into the message

6.286.1.5 `decaf::lang::exceptions::InvalidStateException::InvalidStateException (const std::exception * cause) throw () [inline]`

Constructor.

Parameters

<i>cause</i>	Pointer (p. 2083) to the exception that caused this one to be thrown, the object is cloned caller retains ownership.
--------------	---

6.286.1.6 `decaf::lang::exceptions::InvalidStateException::InvalidStateException (const char * file, const int lineNumber, const char * msg, ...) throw () [inline]`

Constructor - Initializes the file name and line number where this message occurred.

Sets the message to report, using an optional list of arguments to parse into the message

Parameters

<i>file</i>	The file name where exception occurs
<i>lineNumber</i>	The line number where the exception occurred.
<i>msg</i>	The message to report
...	list of primitives that are formatted into the message

6.286.1.7 `virtual decaf::lang::exceptions::InvalidStateException::~~InvalidStateException () throw () [inline, virtual]`

6.286.2 Member Function Documentation

6.286.2.1 `virtual InvalidStateException* decaf::lang::exceptions::InvalidStateException::clone () const [inline, virtual]`

Clones this exception.

This is useful for cases where you need to preserve the type of the original exception

as well as the message. All subclasses should override.

Returns

an new **Exception** (p. 1279) instance that is a copy of this one.

Reimplemented from **decaf::lang::Exception** (p. 1282).

The documentation for this class was generated from the following file:

- src/main/decaf/lang/exceptions/**InvalidStateException.h**

6.287 decaf::io::IOException Class Reference

```
#include <src/main/decaf/io/IOException.h>
```

Inheritance diagram for decaf::io::IOException:

Public Member Functions

- **IOException** () throw ()
Default Constructor.
- **IOException** (const **lang::Exception** &ex) throw ()
Copy Constructor.
- **IOException** (const **IOException** &ex) throw ()
Copy Constructor.
- **IOException** (const char *file, const int lineNumber, const std::exception ***cause**, const char *msg,...) throw ()
Constructor - Initializes the file name and line number where this message occurred.
- **IOException** (const std::exception ***cause**) throw ()
Constructor.
- **IOException** (const char *file, const int lineNumber, const char *msg,...) throw ()
Constructor.
- virtual **IOException** * **clone** () const
Clones this exception.
- virtual ~**IOException** () throw ()

6.287.1 Constructor & Destructor Documentation

6.287.1.1 decaf::io::IOException::IOException () throw () [inline]

Default Constructor.

6.287.1.2 **decaf::io::IOException::IOException (const lang::Exception & ex) throw ()**
`[inline]`

Copy Constructor.

Parameters

<i>ex</i>	the exception to copy
-----------	-----------------------

6.287.1.3 **decaf::io::IOException::IOException (const IOException & ex) throw ()**
`[inline]`

Copy Constructor.

Parameters

<i>ex</i>	the exception to copy, which is an instance of this type
-----------	--

6.287.1.4 **decaf::io::IOException::IOException (const char * file, const int lineNumber, const std::exception * cause, const char * msg, ...) throw ()** `[inline]`

Constructor - Initializes the file name and line number where this message occurred.

Sets the message to report, using an optional list of arguments to parse into the message

Parameters

<i>file</i>	The file name where exception occurs
<i>lineNumber</i>	The line number where the exception occurred.
<i>cause</i>	The exception that was the cause for this one to be thrown.
<i>msg</i>	The message to report
<i>...</i>	list of primitives that are formatted into the message

6.287.1.5 **decaf::io::IOException::IOException (const std::exception * cause) throw ()**
`[inline]`

Constructor.

Parameters

<i>cause</i>	Pointer to the exception that caused this one to be thrown, the object is cloned caller retains ownership.
--------------	--

6.287.1.6 `decaf::io::IOException::IOException (const char * file, const int lineNumber, const char * msg, ...) throw ()` `[inline]`

Constructor.

Parameters

<i>file</i>	The file name where exception occurs
<i>lineNumber</i>	The line number where the exception occurred.
<i>msg</i>	The message to report
...	list of primitives that are formatted into the message

6.287.1.7 `virtual decaf::io::IOException::~IOException () throw ()` `[inline, virtual]`

6.287.2 Member Function Documentation

6.287.2.1 `virtual IOException* decaf::io::IOException::clone () const` `[inline, virtual]`

Clones this exception.

This is useful for cases where you need to preserve the type of the original exception as well as the message. All subclasses should override.

Returns

a new instance of an Exception that is a copy of this instance.

Reimplemented from **decaf::lang::Exception** (p. 1282).

Reimplemented in **decaf::internal::net::ssl::openssl::OpenSSLSocketException** (p. 2032), **decaf::net::BindException** (p. 534), **decaf::net::ConnectException** (p. 933), **decaf::io::UTFDataFormatException** (p. 2877), **decaf::io::UnsupportedEncodingException** (p. 2824), **decaf::net::HttpRetryException** (p. 1411), **decaf::net::MalformedURLException** (p. 1768), **decaf::net::NoRouteToHostException** (p. 1981), **decaf::net::PortUnreachableException** (p. 2109), **decaf::net::ProtocolException** (p. 2213), **decaf::net::SocketTimeoutException** (p. 2495), **decaf::net::UnknownHostException** (p. 2818), **decaf::net::UnknownServiceException** (p. 2821), **decaf::util::zip::ZipException** (p. 2955), **decaf::io::EOFException** (p. 1277), **decaf::io::InterruptedIOException** (p. 1533), and **decaf::net::SocketException** (p. 2473).

The documentation for this class was generated from the following file:

- `src/main/decaf/io/IOException.h`

6.288 activemq::transport::IOTransport Class Reference

Implementation of the **Transport** (p. 2790) interface that performs marshaling of commands to IO streams.

```
#include <src/main/activemq/transport/IOTransport.h>
```

Inheritance diagram for activemq::transport::IOTransport:

Public Member Functions

- **IOTransport** ()
Default Constructor.
- **IOTransport** (const **Pointer**< **wireformat::WireFormat** > &wireFormat)
*Create an instance of this **Transport** (p. 2790) and assign its **WireFormat** instance at creation time.*
- virtual ~**IOTransport** ()
- virtual void **setInputStream** (**decaf::io::DataInputStream** *is)
*Sets the stream from which this **Transport** (p. 2790) implementation will read its data.*
- virtual void **setOutputStream** (**decaf::io::DataOutputStream** *os)
*Sets the stream to which this **Transport** (p. 2790) implementation will write its data.*
- virtual void **oneway** (const **Pointer**< **Command** > &command)
Sends a one-way command.
- virtual **Pointer**< **Response** > **request** (const **Pointer**< **Command** > &command)
Sends the given command to the broker and then waits for the response.

Parameters

command	<i>the command to be sent.</i>
---------	--------------------------------

Returns

the response from the broker.

Exceptions

IOException	<i>if an exception occurs during the read of the command.</i>
UnsupportedOperationException	<i>if this method is not implemented by this transport.</i>

- virtual **Pointer**< **Response** > **request** (const **Pointer**< **Command** > &command, unsigned int timeout)
Sends the given command to the broker and then waits for the response.

Parameters

command	<i>The command to be sent.</i>
timeout	<i>The time to wait for this response.</i>

Returns

the response from the broker.

Exceptions

IOException	<i>if an exception occurs during the read of the command.</i>
UnsupportedOperationException	<i>if this method is not implemented by this transport.</i>

- virtual **Pointer** < **wireformat::WireFormat** > **getWireFormat** () const
Gets the WireFormat instance that is in use by this transport.
- virtual void **setWireFormat** (const **Pointer**< **wireformat::WireFormat** > &wireFormat)
Sets the WireFormat instance to use.
- virtual void **setTransportListener** (**TransportListener** *listener)
Sets the observer of asynchronous events from this transport.
- virtual **TransportListener** * **getTransportListener** () const
Gets the observer of asynchronous events from this transport.
- virtual void **start** ()
*Starts the **Transport** (p. 2790), the send methods of a **Transport** (p. 2790) will throw an exception if used before the **Transport** (p. 2790) is started.*
- virtual void **stop** ()
*Stops the **Transport** (p. 2790).*
- virtual void **close** ()
Closes this object and deallocates the appropriate resources.
- virtual **Transport** * **narrow** (const std::type_info &typeid)
*Narrows down a Chain of Transports to a specific **Transport** (p. 2790) to allow a higher level transport to skip intermediate Transports in certain circumstances.*
- virtual bool **isFaultTolerant** () const
*Is this **Transport** (p. 2790) fault tolerant, meaning that it will reconnect to a broker on disconnect.*
- virtual bool **isConnected** () const
*Is the **Transport** (p. 2790) Connected to its Broker.*
- virtual bool **isClosed** () const
*Has the **Transport** (p. 2790) been shutdown and no longer usable.*
- virtual std::string **getRemoteAddress** () const
- virtual bool **isReconnectSupported** () const
- virtual bool **isUpdateURIsSupported** () const
- virtual void **updateURIs** (bool rebalance AMQCPP_UNUSED, const **decaf::util::List**< **decaf::net::URI** > &uris AMQCPP_UNUSED)
- virtual void **reconnect** (const **decaf::net::URI** &uri AMQCPP_UNUSED)

- virtual void **run** ()
Runs the polling thread.

6.288.1 Detailed Description

Implementation of the **Transport** (p. 2790) interface that performs marshaling of commands to IO streams.

This class does not implement the request method, it only handles oneway messages. A thread polls on the input stream for in-coming commands. When a command is received, the command listener is notified. The polling thread is not started until the start method is called. The close method will close the associated streams. Close can be called explicitly by the user, but is also called in the destructor. Once this object has been closed, it cannot be restarted.

6.288.2 Constructor & Destructor Documentation

6.288.2.1 activemq::transport::IOTransport::IOTransport ()

Default Constructor.

6.288.2.2 activemq::transport::IOTransport::IOTransport (const Pointer< wireformat::WireFormat > & wireFormat)

Create an instance of this **Transport** (p. 2790) and assign its WireFormat instance at creation time.

Parameters

<i>wireFormat</i>	Data encoder / decoder to use when reading and writing.
-------------------	---

6.288.2.3 virtual activemq::transport::IOTransport::~~IOTransport () [virtual]

6.288.3 Member Function Documentation

6.288.3.1 virtual void activemq::transport::IOTransport::close () [virtual]

Closes this object and deallocates the appropriate resources.

The object is generally no longer usable after calling close.

Exceptions

<i>IOException</i>	if an error occurs while closing.
--------------------	-----------------------------------

Implements **decaf::io::Closeable** (p. 817).

6.288.3.2 `virtual std::string activemq::transport::IOTransport::getRemoteAddress () const [inline, virtual]`

Returns

the remote address for this connection

Implements **activemq::transport::Transport** (p. 2792).

6.288.3.3 `virtual TransportListener* activemq::transport::IO-Transport::getTransportListener () const [inline, virtual]`

Gets the observer of asynchronous events from this transport.

Returns

the listener of transport events.

Implements **activemq::transport::Transport** (p. 2792).

6.288.3.4 `virtual Pointer<wireformat::WireFormat> activemq::transport-::IOTransport::getWireFormat () const [inline, virtual]`

Gets the WireFormat instance that is in use by this transport.

In the case of nested transport this method delegates down to the lowest level transport that actually maintains a WireFormat info instance.

Returns

The WireFormat the object used to encode / decode commands.

Implements **activemq::transport::Transport** (p. 2792).

6.288.3.5 `virtual bool activemq::transport::IOTransport::isClosed () const [inline, virtual]`

Has the **Transport** (p. 2790) been shutdown and no longer usable.

Returns

true if the **Transport** (p. 2790)

Implements **activemq::transport::Transport** (p. 2793).

6.288.3.6 `virtual bool activemq::transport::IOTransport::isConnected () const`
`[inline, virtual]`

Is the **Transport** (p. 2790) Connected to its Broker.

Returns

true if a connection has been made.

Implements **activemq::transport::Transport** (p. 2793).

6.288.3.7 `virtual bool activemq::transport::IOTransport::isFaultTolerant () const`
`[inline, virtual]`

Is this **Transport** (p. 2790) fault tolerant, meaning that it will reconnect to a broker on disconnect.

Returns

true if the **Transport** (p. 2790) is fault tolerant.

Implements **activemq::transport::Transport** (p. 2793).

6.288.3.8 `virtual bool activemq::transport::IOTransport::isReconnectSupported ()`
`const [inline, virtual]`

Returns

true if reconnect is supported.

Implements **activemq::transport::Transport** (p. 2794).

6.288.3.9 `virtual bool activemq::transport::IOTransport::isUpdateURIsSupported () const`
`[inline, virtual]`

Returns

true if updating uris is supported.

Implements **activemq::transport::Transport** (p. 2794).

6.288.3.10 `virtual Transport* activemq::transport::IOTransport::narrow (const`
`std::type_info & typeId) [inline, virtual]`

Narrows down a Chain of Transports to a specific **Transport** (p. 2790) to allow a higher level transport to skip intermediate Transports in certain circumstances.

Parameters

<i>typeid</i>	- The type_info of the Object we are searching for.
---------------	---

Returns

the requested Object. or NULL if its not in this chain.

Implements **activemq::transport::Transport** (p. 2794).

6.288.3.11 `virtual void activemq::transport::IOTransport::oneway (const Pointer< Command > & command) [virtual]`

Sends a one-way command.

Does not wait for any response from the broker.

Parameters

<i>command</i>	The command to be sent.
----------------	-------------------------

Exceptions

<i>IOException</i>	if an exception occurs during writing of the command.
<i>Unsupported-OperationException</i>	if this method is not implemented by this transport.

Implements **activemq::transport::Transport** (p. 2795).

6.288.3.12 `virtual void activemq::transport::IOTransport::reconnect (const decaf::net::URI &uri AMQCPP_UNUSED) [inline, virtual]`

This method does nothing in this subclass.

6.288.3.13 `virtual Pointer<Response> activemq::transport::IOTransport::request (const Pointer< Command > & command) [virtual]`

Sends the given command to the broker and then waits for the response.

Parameters

<i>command</i>	the command to be sent.
----------------	-------------------------

Returns

the response from the broker.

Exceptions

<i>IOException</i>	if an exception occurs during the read of the command.
<i>Unsupported-Operation-Exception</i>	if this method is not implemented by this transport.

This method always thrown an `UnsupportedOperationException`.

Implements **activemq::transport::Transport** (p. 2795).

6.288.3.14 `virtual Pointer<Response> activemq::transport::IOTransport::request
(const Pointer<Command> & command, unsigned int timeout)
[virtual]`

Sends the given command to the broker and then waits for the response.

Parameters

<i>command</i>	The command to be sent.
<i>timeout</i>	The time to wait for this response.

Returns

the response from the broker.

Exceptions

<i>IOException</i>	if an exception occurs during the read of the command.
<i>Unsupported-Operation-Exception</i>	if this method is not implemented by this transport.

This method always thrown an `UnsupportedOperationException`.

Implements **activemq::transport::Transport** (p. 2796).

6.288.3.15 `virtual void activemq::transport::IOTransport::run () [virtual]`

Runs the polling thread.

Implements **decaf::lang::Runnable** (p. 2312).

6.288.3.16 `virtual void activemq::transport::IOTransport::setInputStream (
decaf::io::DataInputStream * is) [inline, virtual]`

Sets the stream from which this **Transport** (p. 2790) implementation will read its data.

Parameters

<i>is</i>	The InputStream that will be read from by this object.
-----------	--

6.288.3.17 `virtual void activemq::transport::IOTransport::setOutputStream (decaf::io::DataOutputStream * os) [inline, virtual]`

Sets the stream to which this **Transport** (p. 2790) implementation will write its data.

Parameters

<i>os</i>	The OuputStream that will be written to by this object.
-----------	---

6.288.3.18 `virtual void activemq::transport::IOTransport::setTransportListener (TransportListener * listener) [inline, virtual]`

Sets the observer of asynchronous events from this transport.

Parameters

<i>listener</i>	the listener of transport events.
-----------------	-----------------------------------

Implements **activemq::transport::Transport** (p. 2797).

6.288.3.19 `virtual void activemq::transport::IOTransport::setWireFormat (const Pointer< wireformat::WireFormat > & wireFormat) [inline, virtual]`

Sets the WireFormat instance to use.

Parameters

<i>wireFormat</i>	The WireFormat the object used to encode / decode commands.
-------------------	---

Implements **activemq::transport::Transport** (p. 2797).

6.288.3.20 `virtual void activemq::transport::IOTransport::start () [virtual]`

Starts the **Transport** (p. 2790), the send methods of a **Transport** (p. 2790) will throw an exception if used before the **Transport** (p. 2790) is started.

Exceptions

<i>IOException</i>	if and error occurs while starting the Transport (p. 2790).
--------------------	--

Implements **activemq::transport::Transport** (p. 2797).

6.288.3.21 `virtual void activemq::transport::IOTransport::stop () [virtual]`

Stops the **Transport** (p. 2790).

Exceptions

<i>IOException</i>	if an error occurs while stopping the transport.
--------------------	--

Implements `activemq::transport::Transport` (p. 2797).

6.288.3.22 `virtual void activemq::transport::IOTransport::updateURIs (bool rebalance
AMQCPP_UNUSED, const decaf::util::List< decaf::net::URI > &uris
AMQCPP_UNUSED) [inline, virtual]`

The documentation for this class was generated from the following file:

- `src/main/activemq/transport/IOTransport.h`

6.289 `decaf::lang::Iterable< E >` Class Template Reference

Implementing this interface allows an object to be cast to an **Iterable** (p. 1556) type for generic collections API calls.

```
#include <src/main/decaf/lang/Iterable.h>
```

Inheritance diagram for `decaf::lang::Iterable< E >`:

Public Member Functions

- `virtual ~Iterable ()`
- `virtual decaf::util::Iterator < E > * iterator ()=0`
- `virtual decaf::util::Iterator < E > * iterator () const =0`

6.289.1 Detailed Description

```
template<typename E>class decaf::lang::Iterable< E >
```

Implementing this interface allows an object to be cast to an **Iterable** (p. 1556) type for generic collections API calls.

6.289.2 Constructor & Destructor Documentation

6.289.2.1 `template<typename E> virtual decaf::lang::Iterable< E >::~~Iterable ()`
`[inline, virtual]`

6.289.3 Member Function Documentation

6.289.3.1 `template<typename E> virtual decaf::util::Iterator<E>*`
`decaf::lang::Iterable< E >::iterator () [pure virtual]`

Returns

an iterator over a set of elements of type T.

Implemented in `decaf::util::concurrent::LinkedBlockingQueue< E >` (p. 1626), `decaf::util::concurrent::CopyOnWriteArrayList< E >` (p. 1040), `decaf::util::concurrent::CopyOnWriteArrayList< ServiceListener * >` (p. 1040), `decaf::util::AbstractList< E >` (p. 128), `decaf::util::AbstractList< Pointer< Transport > >` (p. 128), `decaf::util::AbstractList< cms::MessageConsumer * >` (p. 128), `decaf::util::AbstractList< CompositeTask * >` (p. 128), `decaf::util::AbstractList< URI >` (p. 128), `decaf::util::AbstractList< Pointer< MessageDispatch > >` (p. 128), `decaf::util::AbstractList< Pointer< DestinationInfo > >` (p. 128), `decaf::util::AbstractList< PrimitiveValueNode >` (p. 128), `decaf::util::AbstractList< Pointer< Command > >` (p. 128), `decaf::util::AbstractList< Pointer< BackupTransport > >` (p. 128), `decaf::util::AbstractList< cms::MessageProducer * >` (p. 128), `decaf::util::AbstractList< cms::Destination * >` (p. 128), `decaf::util::AbstractList< cms::Session * >` (p. 128), `decaf::util::AbstractList< cms::Connection * >` (p. 128), `decaf::util::StlList< E >` (p. 2548), `decaf::util::PriorityQueue< E >` (p. 2167), `decaf::util::concurrent::SynchronousQueue< E >` (p. 2661), `decaf::util::StlSet< E >` (p. 2580), `decaf::util::StlSet< Pointer< Synchronization > >` (p. 2580), `decaf::util::StlSet< Resource * >` (p. 2580), `decaf::util::concurrent::CopyOnWriteArraySet< E >` (p. 1058), `decaf::util::AbstractSequentialList< E >` (p. 148), `decaf::util::AbstractSequentialList< Pointer< Transport > >` (p. 148), `decaf::util::AbstractSequentialList< cms::MessageConsumer * >` (p. 148), `decaf::util::AbstractSequentialList< CompositeTask * >` (p. 148), `decaf::util::AbstractSequentialList< URI >` (p. 148), `decaf::util::AbstractSequentialList< Pointer< MessageDispatch > >` (p. 148), `decaf::util::AbstractSequentialList< Pointer< DestinationInfo > >` (p. 148), `decaf::util::AbstractSequentialList< PrimitiveValueNode >` (p. 148), `decaf::util::AbstractSequentialList< Pointer< Command > >` (p. 148), `decaf::util::AbstractSequentialList< Pointer< BackupTransport > >` (p. 148), `decaf::util::AbstractSequentialList< cms::MessageProducer * >` (p. 148), `decaf::util::AbstractSequentialList< cms::Destination * >` (p. 148), `decaf::util::AbstractSequentialList< cms::Session * >` (p. 148), and `decaf::util::AbstractSequentialList< cms::Connection * >` (p. 148).

Referenced by `decaf::util::AbstractSequentialList< cms::Connection * >::addAll()`, `decaf::util::concurrent::CopyOnWriteArrayList< ServiceListener * >::addAll()`, `decaf::util::AbstractCollection< cms::Connection * >::addAll()`, `decaf::util::AbstractList< cms::Connection * >::addAll()`, `decaf::util::concurrent::CopyOnWriteArrayList< ServiceListener * >::addAllAbsent()`, `decaf::util::ArrayList< E >::ArrayList()`, `decaf::util::AbstractCollection< cms::Connection * >::clear()`, `decaf::util::AbstractCollection< cms::Connection * >::contains()`, `decaf::util::AbstractCollection< cms::Connection`

* >::containsAll(), decaf::util::concurrent::CopyOnWriteArrayList< ServiceListener * >::containsAll(), decaf::util::AbstractCollection< cms::Connection * >::copy(), decaf::util::concurrent::CopyOnWriteArraySet< E >::equals(), decaf::util::concurrent::CopyOnWriteArrayList< ServiceListener * >::equals(), decaf::util::concurrent::LinkedBlockingQueue< E >::LinkedBlockingQueue(), decaf::util::AbstractCollection< cms::Connection * >::operator=(), decaf::util::AbstractCollection< cms::Connection * >::remove(), decaf::util::AbstractSet< Resource * >::removeAll(), decaf::util::AbstractCollection< cms::Connection * >::removeAll(), decaf::util::AbstractCollection< cms::Connection * >::retainAll(), and decaf::util::AbstractCollection< cms::Connection * >::toArray().

6.289.3.2 template<typename E> virtual decaf::util::Iterator<E>*
decaf::lang::Iterable< E >::iterator () const [pure virtual]

Implemented in decaf::util::concurrent::LinkedBlockingQueue< E > (p. 1627), decaf::util::concurrent::CopyOnWriteArrayList< E > (p. 1040), decaf::util::concurrent::CopyOnWriteArrayList< ServiceListener * > (p. 1040), decaf::util::AbstractList< E > (p. 129), decaf::util::AbstractList< Pointer< Transport > > (p. 129), decaf::util::AbstractList< cms::MessageConsumer * > (p. 129), decaf::util::AbstractList< CompositeTask * > (p. 129), decaf::util::AbstractList< URI > (p. 129), decaf::util::AbstractList< Pointer< MessageDispatch > > (p. 129), decaf::util::AbstractList< Pointer< DestinationInfo > > (p. 129), decaf::util::AbstractList< PrimitiveValueNode > (p. 129), decaf::util::AbstractList< Pointer< Command > > (p. 129), decaf::util::AbstractList< Pointer< BackupTransport > > (p. 129), decaf::util::AbstractList< cms::MessageProducer * > (p. 129), decaf::util::AbstractList< cms::Destination * > (p. 129), decaf::util::AbstractList< cms::Session * > (p. 129), decaf::util::AbstractList< cms::Connection * > (p. 129), decaf::util::StlList< E > (p. 2548), decaf::util::PriorityQueue< E > (p. 2167), decaf::util::concurrent::SynchronousQueue< E > (p. 2661), decaf::util::StlSet< E > (p. 2580), decaf::util::StlSet< Pointer< Synchronization > > (p. 2580), decaf::util::StlSet< Resource * > (p. 2580), decaf::util::concurrent::CopyOnWriteArraySet< E > (p. 1058), decaf::util::AbstractSequentialList< E > (p. 149), decaf::util::AbstractSequentialList< Pointer< Transport > > (p. 149), decaf::util::AbstractSequentialList< cms::MessageConsumer * > (p. 149), decaf::util::AbstractSequentialList< CompositeTask * > (p. 149), decaf::util::AbstractSequentialList< URI > (p. 149), decaf::util::AbstractSequentialList< Pointer< MessageDispatch > > (p. 149), decaf::util::AbstractSequentialList< Pointer< DestinationInfo > > (p. 149), decaf::util::AbstractSequentialList< PrimitiveValueNode > (p. 149), decaf::util::AbstractSequentialList< Pointer< Command > > (p. 149), decaf::util::AbstractSequentialList< Pointer< BackupTransport > > (p. 149), decaf::util::AbstractSequentialList< cms::MessageProducer * > (p. 149), decaf::util::AbstractSequentialList< cms::Destination * > (p. 149), decaf::util::AbstractSequentialList< cms::Session * > (p. 149), and decaf::util::AbstractSequentialList< cms::Connection * > (p. 149).

The documentation for this class was generated from the following file:

- src/main/decaf/lang/Iterable.h

6.290 decaf::util::Iterator< E > Class Template Reference

Defines an object that can be used to iterate over the elements of a collection.

```
#include <src/main/decaf/util/Iterator.h>
```

Inheritance diagram for decaf::util::Iterator< E >:

Public Member Functions

- virtual **~Iterator** ()
- virtual E **next** ()=0
Returns the next element in the iteration.
- virtual bool **hasNext** () const =0
Returns true if the iteration has more elements.
- virtual void **remove** ()=0
Removes from the underlying collection the last element returned by the iterator (optional operation).

6.290.1 Detailed Description

```
template<typename E>class decaf::util::Iterator< E >
```

Defines an object that can be used to iterate over the elements of a collection.

The iterator provides a way to access and remove elements with well defined semantics.

6.290.2 Constructor & Destructor Documentation

6.290.2.1 `template<typename E> virtual decaf::util::Iterator< E >::~~Iterator ()`
`[inline, virtual]`

6.290.3 Member Function Documentation

6.290.3.1 `template<typename E> virtual bool decaf::util::Iterator< E >::hasNext ()`
`const [pure virtual]`

Returns true if the iteration has more elements.

Returns false if the next call to next would result in an **NoSuchElementException** (p. 1984) to be thrown.

Returns

true if there are more elements available for iteration.

Implemented in **decaf::util::concurrent::CopyOnWriteArrayList< E >::ArrayList-Iterator** (p. 459).

6.290.3.2 `template<typename E> virtual E decaf::util::Iterator< E >::next ()` [pure virtual]

Returns the next element in the iteration.

Calling this method repeatedly until the **hasNext()** (p. 1559) method returns false will return each element in the underlying collection exactly once.

Returns

the next element in the iteration of elements.

Exceptions

<i>NoSuchElementException</i> (p. 1984)	if the iteration has no more elements.
---	--

Implemented in **decaf::util::concurrent::CopyOnWriteArrayList< E >::ArrayList-Iterator** (p. 460).

6.290.3.3 `template<typename E> virtual void decaf::util::Iterator< E >::remove ()` [pure virtual]

Removes from the underlying collection the last element returned by the iterator (optional operation).

This method can be called only once per call to next. The behavior of an iterator is unspecified if the underlying collection is modified while the iteration is in progress in any way other than by calling this method.

Exceptions

<i>UnsupportedOperationException</i>	if the remove operation is not supported by this Iterator (p. 1559).
<i>IllegalStateException</i>	if the next method has not yet been called, or the remove method has already been called after the last call to the next method.

Implemented in **decaf::util::concurrent::CopyOnWriteArrayList< E >::ArrayList-Iterator** (p. 461).

The documentation for this class was generated from the following file:

- `src/main/decaf/util/Iterator.h`

6.291 `activemq::commands::JournalQueueAck` Class Reference

```
#include <src/main/activemq/commands/JournalQueueAck.h>
```

Inheritance diagram for `activemq::commands::JournalQueueAck`:

Public Member Functions

- **`JournalQueueAck ()`**
- virtual **`~JournalQueueAck ()`**
- virtual unsigned char **`getDataStructureType ()`** const
*Get the **DataStructure** (p. 1133) Type as defined in CommandTypes.h.*
- virtual **`JournalQueueAck * cloneDataStructure ()`** const
Clone this object and return a new instance that the caller now owns, this will be an exact copy of this one.
- virtual void **`copyDataStructure (const DataStructure *src)`**
Copy the contents of the passed object into this objects members, overwriting any existing data.
- virtual std::string **`toString ()`** const
*Returns a string containing the information for this **DataStructure** (p. 1133) such as its type and value of its elements.*
- virtual bool **`equals (const DataStructure *value)`** const
*Compares the **DataStructure** (p. 1133) passed in to this one, and returns if they are equivalent.*
- virtual const **`Pointer < ActiveMQDestination > & getDestination ()`** const
- virtual **`Pointer < ActiveMQDestination > & getDestination ()`**
- virtual void **`setDestination (const Pointer< ActiveMQDestination > &destination)`**
- virtual const **`Pointer < MessageAck > & getMessageAck ()`** const
- virtual **`Pointer< MessageAck > & getMessageAck ()`**
- virtual void **`setMessageAck (const Pointer< MessageAck > &messageAck)`**

Static Public Attributes

- static const unsigned char **`ID_JOURNALQUEUEACK = 52`**

Protected Attributes

- **`Pointer< ActiveMQDestination > destination`**
- **`Pointer< MessageAck > messageAck`**

6.291.1 Constructor & Destructor Documentation

6.291.1.1 `activemq::commands::JournalQueueAck::JournalQueueAck ()`

6.291.1.2 `virtual activemq::commands::JournalQueueAck::~~JournalQueueAck ()`
[virtual]

6.291.2 Member Function Documentation

6.291.2.1 `virtual JournalQueueAck* activemq::commands::JournalQueueAck::cloneDataStructure () const`
[virtual]

Clone this obbject and return a new instance that the caller now owns, this will be an exact copy of this one.

Returns

new copy of this object.

Implements **activemq::commands::DataStructure** (p. 1133).

6.291.2.2 `virtual void activemq::commands::JournalQueueAck::copyDataStructure (const DataStructure * src)` [virtual]

Copy the contents of the passed object into this objects members, overwriting any existing data.

Returns

src - Source Object

Implements **activemq::commands::DataStructure** (p. 1134).

6.291.2.3 `virtual bool activemq::commands::JournalQueueAck::equals (const DataStructure * value) const` [virtual]

Compares the **DataStructure** (p. 1133) passed in to this one, and returns if they are equivalent.

Equivalent here means that they are of the same type, and that each element of the objects are the same.

Returns

true if **DataStructure** (p. 1133)'s are Equal.

Implements **activemq::commands::DataStructure** (p. 1135).

6.291.2.4 `virtual unsigned char activemq::commands::JournalQueueAck::getDataStructureType () const` [virtual]

Get the **DataStructure** (p. 1133) Type as defined in CommandTypes.h.

Returns

The type of the data structure

Implements **activemq::commands::DataStructure** (p. 1137).

6.291.2.5 `virtual const Pointer<ActiveMQDestination>& activemq::commands::JournalQueueAck::getDestination () const` [virtual]

6.291.2.6 `virtual Pointer<ActiveMQDestination>& activemq::commands::JournalQueueAck::getDestination ()` [virtual]

6.291.2.7 `virtual const Pointer<MessageAck>& activemq::commands::JournalQueueAck::getMessageAck () const` [virtual]

6.291.2.8 `virtual Pointer<MessageAck>& activemq::commands::JournalQueueAck::getMessageAck ()` [virtual]

6.291.2.9 `virtual void activemq::commands::JournalQueueAck::setDestination (const Pointer< ActiveMQDestination > & destination)` [virtual]

6.291.2.10 `virtual void activemq::commands::JournalQueueAck::setMessageAck (const Pointer< MessageAck > & messageAck)` [virtual]

6.291.2.11 `virtual std::string activemq::commands::JournalQueueAck::toString () const` [virtual]

Returns a string containing the information for this **DataStructure** (p. 1133) such as its type and value of its elements.

Returns

formatted string useful for debugging.

Reimplemented from **activemq::commands::BaseDataStructure** (p. 531).

6.291.3 Field Documentation

6.291.3.1 `Pointer<ActiveMQDestination> activemq::commands::JournalQueueAck::destination` [protected]

6.292 activemq::wireformat::openwire::marshal::generated::JournalQueueAckMarshaller Class

Reference

1569

6.291.3.2 `const unsigned char activemq::commands::JournalQueueAck::ID_JOURNALQUEUEACK = 52` [static]

6.291.3.3 `Pointer<MessageAck> activemq::commands::JournalQueueAck::messageAck` [protected]

The documentation for this class was generated from the following file:

- `src/main/activemq/commands/JournalQueueAck.h`

6.292 activemq::wireformat::openwire::marshal::generated::JournalQueueAckMarshaller Class Reference

Marshaling code for Open Wire Format for **JournalQueueAckMarshaller** (p. 1564).

```
#include <src/main/activemq/wireformat/openwire/marshal/generated/-  
JournalQueueAckMarshaller.h>
```

Inheritance diagram for `activemq::wireformat::openwire::marshal::generated::JournalQueueAckMarshaller`:

Public Member Functions

- **JournalQueueAckMarshaller** ()
- virtual `~JournalQueueAckMarshaller` ()
- virtual `commands::DataStructure * createObject` () const
Creates a new instance of the class that this class is a marshaling director for.
- virtual unsigned char **getDataStructureType** () const
Gets the DataStructureType that this class marshals/unmarshals.
- virtual void **tightUnmarshal** (`OpenWireFormat *wireFormat`, `commands::DataStructure *dataStructure`, `decaf::io::DataInputStream *dataIn`, `utils::BooleanStream *bs`)
Tight Un-marhsal to the given stream.
- virtual int **tightMarshal1** (`OpenWireFormat *wireFormat`, `commands::DataStructure *dataStructure`, `utils::BooleanStream *bs`)
Tight Marshal to the given stream.
- virtual void **tightMarshal2** (`OpenWireFormat *wireFormat`, `commands::DataStructure *dataStructure`, `decaf::io::DataOutputStream *dataOut`, `utils::BooleanStream *bs`)
Tight Marshal to the given stream.
- virtual void **looseUnmarshal** (`OpenWireFormat *wireFormat`, `commands::DataStructure *dataStructure`, `decaf::io::DataInputStream *dataIn`)
Loose Un-marhsal to the given stream.

- virtual void **looseMarshal** (**OpenWireFormat** *wireFormat, **commands::Data-Structure** *dataStructure, **decaf::io::DataOutputStream** *dataOut)

Tight Marhsal to the given stream.

6.292.1 Detailed Description

Marshaling code for Open Wire Format for **JournalQueueAckMarshaller** (p. 1564).

NOTE!: This file is auto generated - do not modify! if you need to make a change, please see the Java Classes in the activemq-openwire-generator module

6.292.2 Constructor & Destructor Documentation

6.292.2.1 **activemq::wireformat::openwire::marshal::generated::JournalQueueAckMarshaller::JournalQueueAckMarshaller** ()
[inline]

6.292.2.2 **virtual activemq::wireformat::openwire::marshal::generated::JournalQueueAckMarshaller::~~JournalQueueAckMarshaller** () [inline, virtual]

6.292.3 Member Function Documentation

6.292.3.1 **virtual commands::DataStructure* activemq::wireformat::openwire::marshal::generated::JournalQueueAckMarshaller::createObject** ()
const [virtual]

Creates a new instance of the class that this class is a marshaling director for.

Returns

newly allocated Command

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1120).

6.292.3.2 **virtual unsigned char activemq::wireformat::openwire::marshal::generated::JournalQueueAckMarshaller::getDataStructureType** () **const**
[virtual]

Gets the DataStructureType that this class marshals/unmarshals.

Returns

byte Id of this classes DataStructureType

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1122).

6.292 activemq::wireformat::openwire::marshal::generated::JournalQueueAck-Marshaller Class

Reference

1571

6.292.3.3 virtual void **activemq::wireformat::openwire::marshal::generated::JournalQueueAckMarshaller::looseMarshal** (**OpenWireFormat** * *format*, **commands::DataStructure** * *command*, **decaf::io::DataOutputStream** * *ds*) [virtual]

Tight Marhsal to the given stream.

Parameters

<i>format</i>	- The OpenWireFormat properties
<i>command</i>	- the object to Marshal
<i>ds</i>	- DataOutputStream to marshal to

Exceptions

<i>IOException</i>	if an error occurs.
--------------------	---------------------

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1123).

6.292.3.4 virtual void **activemq::wireformat::openwire::marshal::generated::JournalQueueAckMarshaller::looseUnmarshal** (**OpenWireFormat** * *format*, **commands::DataStructure** * *command*, **decaf::io::DataInputStream** * *dis*) [virtual]

Loose Un-marhsal to the given stream.

Parameters

<i>format</i>	- The OpenWireFormat properties
<i>command</i>	- the object to Un-Marshal
<i>dis</i>	- the DataInputStream to Un-Marshal from

Exceptions

<i>IOException</i>	if an error occurs.
--------------------	---------------------

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1125).

6.292.3.5 virtual int **activemq::wireformat::openwire::marshal::generated::JournalQueueAckMarshaller::tightMarshal1** (**OpenWireFormat** * *format*, **commands::DataStructure** * *command*, **utils::BooleanStream** * *bs*) [virtual]

Tight Marhsal to the given stream.

Parameters

<i>format</i>	The OpenwireFormat properties
<i>command</i>	The object to Marshal
<i>bs</i>	The boolean stream to marshal to.

Exceptions

<i>IOException</i>	if an error occurs.
--------------------	---------------------

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1127).

6.292.3.6 `virtual void activemq::wireformat::openwire::marshal::generated::JournalQueueAckMarshaller::tightMarshal2 (OpenWireFormat * format, commands::DataStructure * command, decaf::io::DataOutputStream * ds, utils::BooleanStream * bs) [virtual]`

Tight Marshal to the given stream.

Parameters

<i>format</i>	- The OpenwireFormat properties
<i>command</i>	- the object to Marshal
<i>ds</i>	- the DataOutputStream to Marshal to
<i>bs</i>	- boolean stream to marshal to.

Exceptions

<i>IOException</i>	if an error occurs.
--------------------	---------------------

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1129).

6.292.3.7 `virtual void activemq::wireformat::openwire::marshal::generated::JournalQueueAckMarshaller::tightUnmarshal (OpenWireFormat * format, commands::DataStructure * command, decaf::io::DataInputStream * dis, utils::BooleanStream * bs) [virtual]`

Tight Un-marshal to the given stream.

Parameters

<i>format</i>	- The OpenwireFormat properties
<i>command</i>	- the object to Un-Marshal
<i>dis</i>	- the DataInputStream to Un-Marshal from
<i>bs</i>	- boolean stream to unmarshal from.

Exceptions

<i>IOException</i>	if an error occurs.
--------------------	---------------------

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1131).

The documentation for this class was generated from the following file:

- src/main/activemq/wireformat/openwire/marshal/generated/**JournalQueueAck-Marshaller.h**

6.293 activemq::commands::JournalTopicAck Class Reference

```
#include <src/main/activemq/commands/JournalTopicAck.h>
```

Inheritance diagram for activemq::commands::JournalTopicAck:

Public Member Functions

- **JournalTopicAck** ()
- virtual **~JournalTopicAck** ()
- virtual unsigned char **getDataStructureType** () const
*Get the **DataStructure** (p. 1133) Type as defined in CommandTypes.h.*
- virtual **JournalTopicAck * cloneDataStructure** () const
Clone this obbect and return a new instance that the caller now owns, this will be an exact copy of this one.
- virtual void **copyDataStructure** (const **DataStructure** *src)
Copy the contents of the passed object into this objects members, overwriting any existing data.
- virtual std::string **toString** () const
*Returns a string containing the information for this **DataStructure** (p. 1133) such as its type and value of its elements.*
- virtual bool **equals** (const **DataStructure** *value) const
*Compares the **DataStructure** (p. 1133) passed in to this one, and returns if they are equivalent.*
- virtual const **Pointer** < **ActiveMQDestination** > & **getDestination** () const
- virtual **Pointer** < **ActiveMQDestination** > & **getDestination** ()
- virtual void **setDestination** (const **Pointer**< **ActiveMQDestination** > &destination)
- virtual const **Pointer** < **MessageId** > & **getMessageId** () const
- virtual **Pointer**< **MessageId** > & **getMessageId** ()
- virtual void **setMessageId** (const **Pointer**< **MessageId** > &messageId)
- virtual long long **getMessageSequenceId** () const

- virtual void **setMessageSequenceld** (long long **messageSequenceld**)
- virtual const std::string & **getSubscriptionName** () const
- virtual std::string & **getSubscriptionName** ()
- virtual void **setSubscriptionName** (const std::string &**subscriptionName**)
- virtual const std::string & **getClientId** () const
- virtual std::string & **getClientId** ()
- virtual void **setClientId** (const std::string &**clientId**)
- virtual const **Pointer** < **TransactionId** > & **getTransactionId** () const
- virtual **Pointer**< **TransactionId** > & **getTransactionId** ()
- virtual void **setTransactionId** (const **Pointer**< **TransactionId** > &**transactionId**)

Static Public Attributes

- static const unsigned char **ID_JOURNALTOPICACK** = 50

Protected Attributes

- **Pointer**< **ActiveMQDestination** > **destination**
- **Pointer**< **MessageId** > **messageld**
- long long **messageSequenceld**
- std::string **subscriptionName**
- std::string **clientId**
- **Pointer**< **TransactionId** > **transactionId**

6.293.1 Constructor & Destructor Documentation

6.293.1.1 **activemq::commands::JournalTopicAck::JournalTopicAck** ()

6.293.1.2 **virtual activemq::commands::JournalTopicAck::~~JournalTopicAck** ()
[virtual]

6.293.2 Member Function Documentation

6.293.2.1 **virtual JournalTopicAck* activemq::commands::JournalTopicAck::cloneDataStructure** () const
[virtual]

Clone this object and return a new instance that the caller now owns, this will be an exact copy of this one.

Returns

new copy of this object.

Implements **activemq::commands::DataStructure** (p. 1133).

6.293.2.2 `virtual void activemq::commands::JournalTopicAck::copyDataStructure (const DataStructure * src) [virtual]`

Copy the contents of the passed object into this objects members, overwriting any existing data.

Returns

src - Source Object

Implements **activemq::commands::DataStructure** (p. 1134).

6.293.2.3 `virtual bool activemq::commands::JournalTopicAck::equals (const DataStructure * value) const [virtual]`

Compares the **DataStructure** (p. 1133) passed in to this one, and returns if they are equivalent.

Equivalent here means that they are of the same type, and that each element of the objects are the same.

Returns

true if **DataStructure** (p. 1133)'s are Equal.

Implements **activemq::commands::DataStructure** (p. 1135).

6.293.2.4 `virtual const std::string& activemq::commands::JournalTopicAck::getClientId () const [virtual]`

6.293.2.5 `virtual std::string& activemq::commands::JournalTopicAck::getClientId () [virtual]`

6.293.2.6 `virtual unsigned char activemq::commands::JournalTopicAck::getDataStructureType () const [virtual]`

Get the **DataStructure** (p. 1133) Type as defined in CommandTypes.h.

Returns

The type of the data structure

Implements **activemq::commands::DataStructure** (p. 1137).

6.293.2.7 `virtual const Pointer<ActiveMQDestination>& activemq::commands::JournalTopicAck::getDestination () const [virtual]`

- 6.293.2.8 `virtual Pointer<ActiveMQDestination>& activemq-
::commands::JournalTopicAck::getDestination ()
[virtual]`
- 6.293.2.9 `virtual const Pointer<Messageld>& activemq-
::commands::JournalTopicAck::getMessageld () const
[virtual]`
- 6.293.2.10 `virtual Pointer<Messageld>& activemq::commands::JournalTopicAck-
::getMessageld () [virtual]`
- 6.293.2.11 `virtual long long activemq::commands::JournalTopicAck::getMessage-
Sequenceld () const [virtual]`
- 6.293.2.12 `virtual const std::string& activemq::commands::-
JournalTopicAck::getSubscriptionName () const
[virtual]`
- 6.293.2.13 `virtual std::string& activemq::commands::JournalTopicAck::get-
SubscriptionName () [virtual]`
- 6.293.2.14 `virtual const Pointer<TransactionId>& activemq-
::commands::JournalTopicAck::getTransactionId () const
[virtual]`
- 6.293.2.15 `virtual Pointer<TransactionId>& activemq::commands::JournalTopic-
Ack::getTransactionId () [virtual]`
- 6.293.2.16 `virtual void activemq::commands::JournalTopicAck::setClientId (const
std::string & clientId) [virtual]`
- 6.293.2.17 `virtual void activemq::commands::JournalTopicAck::setDestination (
const Pointer< ActiveMQDestination > & destination) [virtual]`
- 6.293.2.18 `virtual void activemq::commands::JournalTopicAck::setMessageld (
const Pointer< Messageld > & messageld) [virtual]`
- 6.293.2.19 `virtual void activemq::commands::JournalTopicAck::set-
MessageSequenceld (long long messageSequenceld)
[virtual]`
- 6.293.2.20 `virtual void activemq::commands::JournalTopicAck::set-
SubscriptionName (const std::string & subscriptionName)
[virtual]`
- 6.293.2.21 `virtual void activemq::commands::JournalTopicAck::setTransactionId (
const Pointer< TransactionId > & transactionId) [virtual]`

6.294 activemq::wireformat::openwire::marshal::generated::JournalTopicAck-Marshaller Class

Reference

1577

6.293.2.22 virtual std::string activemq::commands::JournalTopicAck::toString ()
const [virtual]

Returns a string containing the information for this **DataSet** (p. 1133) such as its type and value of its elements.

Returns

formatted string useful for debugging.

Reimplemented from **activemq::commands::BaseDataSet** (p. 531).

6.293.3 Field Documentation

6.293.3.1 std::string activemq::commands::JournalTopicAck::clientId
[protected]

6.293.3.2 Pointer<ActiveMQDestination> activemq::commands::JournalTopicAck::destination [protected]

6.293.3.3 const unsigned char activemq::commands::JournalTopicAck::ID_JOURNALTOPICACK = 50 [static]

6.293.3.4 Pointer<MessageId> activemq::commands::JournalTopicAck::messageId [protected]

6.293.3.5 long long activemq::commands::JournalTopicAck::messageSequenceId [protected]

6.293.3.6 std::string activemq::commands::JournalTopicAck::subscriptionName [protected]

6.293.3.7 Pointer<TransactionId> activemq::commands::JournalTopicAck::transactionId [protected]

The documentation for this class was generated from the following file:

- src/main/activemq/commands/**JournalTopicAck.h**

6.294 activemq::wireformat::openwire::marshal::generated::JournalTopicAckMarshaller Class Reference

Marshaling code for Open Wire Format for **JournalTopicAckMarshaller** (p. 1572).

```
#include <src/main/activemq/wireformat/openwire/marshal/generated/-  
JournalTopicAckMarshaller.h>
```

Inheritance diagram for activemq::wireformat::openwire::marshal::generated::JournalTopicAckMarshaller:

Public Member Functions

- **JournalTopicAckMarshaller** ()
- virtual **~JournalTopicAckMarshaller** ()
- virtual **commands::DataStructure * createObject** () const
Creates a new instance of the class that this class is a marshaling director for.
- virtual unsigned char **getDataStructureType** () const
Gets the DataStructureType that this class marshals/unmarshals.
- virtual void **tightUnmarshal** (**OpenWireFormat *wireFormat**, **commands::DataStructure *dataStructure**, **decaf::io::DataInputStream *dataIn**, **utils::BooleanStream *bs**)
Tight Un-marhsal to the given stream.
- virtual int **tightMarshal1** (**OpenWireFormat *wireFormat**, **commands::DataStructure *dataStructure**, **utils::BooleanStream *bs**)
Tight Marhsal to the given stream.
- virtual void **tightMarshal2** (**OpenWireFormat *wireFormat**, **commands::DataStructure *dataStructure**, **decaf::io::DataOutputStream *dataOut**, **utils::BooleanStream *bs**)
Tight Marhsal to the given stream.
- virtual void **looseUnmarshal** (**OpenWireFormat *wireFormat**, **commands::DataStructure *dataStructure**, **decaf::io::DataInputStream *dataIn**)
Loose Un-marhsal to the given stream.
- virtual void **looseMarshal** (**OpenWireFormat *wireFormat**, **commands::DataStructure *dataStructure**, **decaf::io::DataOutputStream *dataOut**)
Tight Marhsal to the given stream.

6.294.1 Detailed Description

Marshaling code for Open Wire Format for **JournalTopicAckMarshaller** (p. 1572).

NOTE!: This file is auto generated - do not modify! if you need to make a change, please see the Java Classes in the activemq-openwire-generator module

6.294.2 Constructor & Destructor Documentation

- 6.294.2.1 **activemq::wireformat::openwire::marshal::generated::JournalTopicAckMarshaller::JournalTopicAckMarshaller** ()
[inline]

6.294 activemq::wireformat::openwire::marshal::generated::JournalTopicAck-Marshaller Class

Reference

1579

6.294.2.2 `virtual activemq::wireformat::openwire::marshal::generated::JournalTopicAckMarshaller::~JournalTopicAckMarshaller () [inline, virtual]`

6.294.3 Member Function Documentation

6.294.3.1 `virtual commands::DataStructure* activemq::wireformat::openwire::marshal::generated::JournalTopicAckMarshaller::createObject () const [virtual]`

Creates a new instance of the class that this class is a marshaling director for.

Returns

newly allocated Command

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1120).

6.294.3.2 `virtual unsigned char activemq::wireformat::openwire::marshal::generated::JournalTopicAckMarshaller::getDataStructureType () const [virtual]`

Gets the DataStructureType that this class marshals/unmarshals.

Returns

byte Id of this classes DataStructureType

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1122).

6.294.3.3 `virtual void activemq::wireformat::openwire::marshal::generated::JournalTopicAckMarshaller::looseMarshal (OpenWireFormat * format, commands::DataStructure * command, decaf::io::DataOutputStream * ds) [virtual]`

Tight Marshal to the given stream.

Parameters

<i>format</i>	- The OpenWireFormat properties
<i>command</i>	- the object to Marshal
<i>ds</i>	- DataOutputStream to marshal to

Exceptions

<i>IOException</i>	if an error occurs.
--------------------	---------------------

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1123).

6.294.3.4 `virtual void activemq::wireformat::openwire::marshal::generated::JournalTopicAckMarshaller::looseUnmarshal (OpenWireFormat * format, commands::DataStructure * command, decaf::io::DataInputStream * dis) [virtual]`

Loose Un-marhsal to the given stream.

Parameters

<i>format</i>	- The OpenWireFormat properties
<i>command</i>	- the object to Un-Marshal
<i>dis</i>	- the DataInputStream to Un-Marshal from

Exceptions

<i>IOException</i>	if an error occurs.
--------------------	---------------------

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1125).

6.294.3.5 `virtual int activemq::wireformat::openwire::marshal::generated::JournalTopicAckMarshaller::tightMarshal1 (OpenWireFormat * format, commands::DataStructure * command, utils::BooleanStream * bs) [virtual]`

Tight Marhsal to the given stream.

Parameters

<i>format</i>	The OpenWireFormat properties
<i>command</i>	The object to Marshal
<i>bs</i>	The boolean stream to marshal to.

Exceptions

<i>IOException</i>	if an error occurs.
--------------------	---------------------

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1127).

6.294 activemq::wireformat::openwire::marshal::generated::JournalTopicAck-Marshaller Class

Reference

1581

6.294.3.6 virtual void **activemq::wireformat::openwire::marshal::generated::JournalTopicAckMarshaller::tightMarshal2** (**OpenWireFormat** * *format*, **commands::DataStructure** * *command*, **decaf::io::DataOutputStream** * *ds*, **utils::BooleanStream** * *bs*) [virtual]

Tight Marhsal to the given stream.

Parameters

<i>format</i>	- The OpenWireFormat properties
<i>command</i>	- the object to Marshal
<i>ds</i>	- the DataOutputStream to Marshal to
<i>bs</i>	- boolean stream to marshal to.

Exceptions

<i>IOException</i>	if an error occurs.
--------------------	---------------------

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1129).

6.294.3.7 virtual void **activemq::wireformat::openwire::marshal::generated::JournalTopicAckMarshaller::tightUnmarshal** (**OpenWireFormat** * *format*, **commands::DataStructure** * *command*, **decaf::io::DataInputStream** * *dis*, **utils::BooleanStream** * *bs*) [virtual]

Tight Un-marhsal to the given stream.

Parameters

<i>format</i>	- The OpenWireFormat properties
<i>command</i>	- the object to Un-Marshal
<i>dis</i>	- the DataInputStream to Un-Marshal from
<i>bs</i>	- boolean stream to unmarshal from.

Exceptions

<i>IOException</i>	if an error occurs.
--------------------	---------------------

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1131).

The documentation for this class was generated from the following file:

- src/main/activemq/wireformat/openwire/marshal/generated/**JournalTopicAck-Marshaller.h**

6.295 activemq::commands::JournalTrace Class Reference

```
#include <src/main/activemq/commands/JournalTrace.h>
```

Inheritance diagram for activemq::commands::JournalTrace:

Public Member Functions

- **JournalTrace** ()
- virtual **~JournalTrace** ()
- virtual unsigned char **getDataStructureType** () const
*Get the **DataStructure** (p. 1133) Type as defined in CommandTypes.h.*
- virtual **JournalTrace * cloneDataStructure** () const
Clone this obbject and return a new instance that the caller now owns, this will be an exact copy of this one.
- virtual void **copyDataStructure** (const **DataStructure** *src)
Copy the contents of the passed object into this objects members, overwriting any existing data.
- virtual std::string **toString** () const
*Returns a string containing the information for this **DataStructure** (p. 1133) such as its type and value of its elements.*
- virtual bool **equals** (const **DataStructure** *value) const
*Compares the **DataStructure** (p. 1133) passed in to this one, and returns if they are equivalent.*
- virtual const std::string & **getMessage** () const
- virtual std::string & **getMessage** ()
- virtual void **setMessage** (const std::string &message)

Static Public Attributes

- static const unsigned char **ID_JOURNALTRACE** = 53

Protected Attributes

- std::string **message**

6.295.1 Constructor & Destructor Documentation

6.295.1.1 **activemq::commands::JournalTrace::JournalTrace** ()

6.295.1.2 **virtual activemq::commands::JournalTrace::~~JournalTrace** ()
[virtual]

6.295.2 Member Function Documentation

6.295.2.1 virtual **JournalTrace*** **activemq::commands::JournalTrace::cloneDataStructure** () const [virtual]

Clone this obbject and return a new instance that the caller now owns, this will be an exact copy of this one.

Returns

new copy of this object.

Implements **activemq::commands::DataStructure** (p. 1133).

6.295.2.2 virtual void **activemq::commands::JournalTrace::copyDataStructure** (const **DataStructure** * *src*) [virtual]

Copy the contents of the passed object into this objects members, overwriting any existing data.

Returns

src - Source Object

Implements **activemq::commands::DataStructure** (p. 1134).

6.295.2.3 virtual bool **activemq::commands::JournalTrace::equals** (const **DataStructure** * *value*) const [virtual]

Compares the **DataStructure** (p. 1133) passed in to this one, and returns if they are equivalent.

Equivalent here means that they are of the same type, and that each element of the objects are the same.

Returns

true if **DataStructure** (p. 1133)'s are Equal.

Implements **activemq::commands::DataStructure** (p. 1135).

6.295.2.4 virtual unsigned char **activemq::commands::JournalTrace::getDataStructureType** () const [virtual]

Get the **DataStructure** (p. 1133) Type as defined in CommandTypes.h.

Returns

The type of the data structure

Implements **activemq::commands::DataStructure** (p. 1137).

6.295.2.5 `virtual const std::string& activemq::commands::JournalTrace::getMessage () const [virtual]`

6.295.2.6 `virtual std::string& activemq::commands::JournalTrace::getMessage () [virtual]`

6.295.2.7 `virtual void activemq::commands::JournalTrace::setMessage (const std::string & message) [virtual]`

6.295.2.8 `virtual std::string activemq::commands::JournalTrace::toString () const [virtual]`

Returns a string containing the information for this **DataStructure** (p. 1133) such as its type and value of its elements.

Returns

formatted string useful for debugging.

Reimplemented from **activemq::commands::BaseDataStructure** (p. 531).

6.295.3 Field Documentation

6.295.3.1 `const unsigned char activemq::commands::JournalTrace::ID_JOURNALTRACE = 53 [static]`

6.295.3.2 `std::string activemq::commands::JournalTrace::message [protected]`

The documentation for this class was generated from the following file:

- `src/main/activemq/commands/JournalTrace.h`

6.296 **activemq::wireformat::openwire::marshal::generated::JournalTraceMarshaller Class Reference**

Marshaling code for Open Wire Format for **JournalTraceMarshaller** (p. 1579).

```
#include <src/main/activemq/wireformat/openwire/marshal/generated/-
JournalTraceMarshaller.h>
```

Inheritance diagram for `activemq::wireformat::openwire::marshal::generated::JournalTraceMarshaller`:

Public Member Functions

- **JournalTraceMarshaller** ()
- virtual \sim **JournalTraceMarshaller** ()
- virtual **commands::DataStructure** * **createObject** () const
Creates a new instance of the class that this class is a marshaling director for.
- virtual unsigned char **getDataStructureType** () const
Gets the DataStructureType that this class marshals/unmarshals.
- virtual void **tightUnmarshal** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataInputStream** *dataIn, **utils::BooleanStream** *bs)
Tight Un-marhsal to the given stream.
- virtual int **tightMarshal1** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **utils::BooleanStream** *bs)
Tight Marshal to the given stream.
- virtual void **tightMarshal2** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataOutputStream** *dataOut, **utils::BooleanStream** *bs)
Tight Marshal to the given stream.
- virtual void **looseUnmarshal** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataInputStream** *dataIn)
Loose Un-marhsal to the given stream.
- virtual void **looseMarshal** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataOutputStream** *dataOut)
Tight Marshal to the given stream.

6.296.1 Detailed Description

Marshaling code for Open Wire Format for **JournalTraceMarshaller** (p. 1579).

NOTE!: This file is auto generated - do not modify! if you need to make a change, please see the Java Classes in the activemq-openwire-generator module

6.296.2 Constructor & Destructor Documentation

6.296.2.1 **activemq::wireformat::openwire::marshal::generated::JournalTraceMarshaller::JournalTraceMarshaller** ()
[inline]

6.296.2.2 virtual **activemq::wireformat::openwire::marshal::generated::JournalTraceMarshaller::~JournalTraceMarshaller** () [inline, virtual]

6.296.3 Member Function Documentation

6.296.3.1 **virtual commands::DataStructure* activemq::wireformat::openwire-
::marshal::generated::JournalTraceMarshaller::createObject () const**
[virtual]

Creates a new instance of the class that this class is a marshaling director for.

Returns

newly allocated Command

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller**
(p. 1120).

6.296.3.2 **virtual unsigned char activemq::wireformat::openwire::marshal-
::generated::JournalTraceMarshaller::getDataStructureType () const**
[virtual]

Gets the DataStructureType that this class marshals/unmarshals.

Returns

byte Id of this classes DataStructureType

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller**
(p. 1122).

6.296.3.3 **virtual void activemq::wireformat::openwire::marshal::generated-
::JournalTraceMarshaller::looseMarshal (OpenWireFormat * *format*,
commands::DataStructure * *command*, decaf::io::DataOutputStream * *ds*
)** [virtual]

Tight Marshal to the given stream.

Parameters

<i>format</i>	- The OpenWireFormat properties
<i>command</i>	- the object to Marshal
<i>ds</i>	- DataOutputStream to marshal to

Exceptions

<i>IOException</i>	if an error occurs.
--------------------	---------------------

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller**
(p. 1123).

6.296

activemq::wireformat::openwire::marshal::generated::JournalTraceMarshaller

Class Reference

1587

6.296.3.4 **virtual void activemq::wireformat::openwire::marshal::generated::-**
JournalTraceMarshaller::looseUnmarshal (OpenWireFormat * *format*,
commands::DataStructure * *command*, decaf::io::DataInputStream * *dis*)
[virtual]

Loose Un-marhsal to the given stream.

Parameters

<i>format</i>	- The OpenWireFormat properties
<i>command</i>	- the object to Un-Marshal
<i>dis</i>	- the DataInputStream to Un-Marshal from

Exceptions

<i>IOException</i>	if an error occurs.
--------------------	---------------------

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller**
(p. 1125).

6.296.3.5 **virtual int activemq::wireformat::openwire::marshal::generated::-**
JournalTraceMarshaller::tightMarshal1 (OpenWireFormat * *format*,
commands::DataStructure * *command*, utils::BooleanStream * *bs*)
[virtual]

Tight Marhsal to the given stream.

Parameters

<i>format</i>	The OpenWireFormat properties
<i>command</i>	The object to Marshal
<i>bs</i>	The boolean stream to marshal to.

Exceptions

<i>IOException</i>	if an error occurs.
--------------------	---------------------

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller**
(p. 1127).

6.296.3.6 **virtual void activemq::wireformat::openwire::marshal::generated::-**
JournalTraceMarshaller::tightMarshal2 (OpenWireFormat * *format*,
commands::DataStructure * *command*, decaf::io::DataOutputStream * *ds*,
utils::BooleanStream * *bs*) [virtual]

Tight Marhsal to the given stream.

Parameters

<i>format</i>	- The OpenwireFormat properties
<i>command</i>	- the object to Marshal
<i>ds</i>	- the DataOutputStream to Marshal to
<i>bs</i>	- boolean stream to marshal to.

Exceptions

<i>IOException</i>	if an error occurs.
--------------------	---------------------

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1129).

6.296.3.7 virtual void activemq::wireformat::openwire::marshal::generated::JournalTraceMarshaller::tightUnmarshal (OpenWireFormat * *format*, commands::DataStructure * *command*, decaf::io::DataInputStream * *dis*, utils::BooleanStream * *bs*) [virtual]

Tight Un-marhsal to the given stream.

Parameters

<i>format</i>	- The OpenwireFormat properties
<i>command</i>	- the object to Un-Marshal
<i>dis</i>	- the DataInputStream to Un-Marshal from
<i>bs</i>	- boolean stream to unmarshal from.

Exceptions

<i>IOException</i>	if an error occurs.
--------------------	---------------------

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1131).

The documentation for this class was generated from the following file:

- src/main/activemq/wireformat/openwire/marshal/generated/**JournalTraceMarshaller.h**

6.297 activemq::commands::JournalTransaction Class Reference

```
#include <src/main/activemq/commands/JournalTransaction.h>
```

Inheritance diagram for activemq::commands::JournalTransaction:

Public Member Functions

- **JournalTransaction** ()
- virtual **~JournalTransaction** ()
- virtual unsigned char **getDataStructureType** () const
*Get the **DataSetructure** (p. 1133) Type as defined in CommandTypes.h.*
- virtual **JournalTransaction * cloneDataSetructure** () const
Clone this object and return a new instance that the caller now owns, this will be an exact copy of this one.
- virtual void **copyDataSetructure** (const **DataSetructure** *src)
Copy the contents of the passed object into this objects members, overwriting any existing data.
- virtual std::string **toString** () const
*Returns a string containing the information for this **DataSetructure** (p. 1133) such as its type and value of its elements.*
- virtual bool **equals** (const **DataSetructure** *value) const
*Compares the **DataSetructure** (p. 1133) passed in to this one, and returns if they are equivalent.*
- virtual const **Pointer** < **TransactionId** > & **getTransactionId** () const
- virtual **Pointer** < **TransactionId** > & **getTransactionId** ()
- virtual void **setTransactionId** (const **Pointer** < **TransactionId** > &transactionId)
- virtual unsigned char **getType** () const
- virtual void **setType** (unsigned char type)
- virtual bool **getWasPrepared** () const
- virtual void **setWasPrepared** (bool wasPrepared)

Static Public Attributes

- static const unsigned char **ID_JOURNALTRANSACTION** = 54

Protected Attributes

- **Pointer** < **TransactionId** > transactionId
- unsigned char type
- bool wasPrepared

6.297.1 Constructor & Destructor Documentation

6.297.1.1 **activemq::commands::JournalTransaction::JournalTransaction** ()

6.297.1.2 **virtual activemq::commands::JournalTransaction::~~JournalTransaction**
() [virtual]

6.297.2 Member Function Documentation

6.297.2.1 `virtual JournalTransaction* activemq::commands::JournalTransaction::cloneDataStructure () const`
[virtual]

Clone this object and return a new instance that the caller now owns, this will be an exact copy of this one.

Returns

new copy of this object.

Implements **activemq::commands::DataStructure** (p. 1133).

6.297.2.2 `virtual void activemq::commands::JournalTransaction::copyDataStructure (const DataStructure * src)`
[virtual]

Copy the contents of the passed object into this objects members, overwriting any existing data.

Returns

src - Source Object

Implements **activemq::commands::DataStructure** (p. 1134).

6.297.2.3 `virtual bool activemq::commands::JournalTransaction::equals (const DataStructure * value) const` [virtual]

Compares the **DataStructure** (p. 1133) passed in to this one, and returns if they are equivalent.

Equivalent here means that they are of the same type, and that each element of the objects are the same.

Returns

true if **DataStructure** (p. 1133)'s are Equal.

Implements **activemq::commands::DataStructure** (p. 1135).

6.297.2.4 `virtual unsigned char activemq::commands::JournalTransaction::getDataStructureType () const`
[virtual]

Get the **DataStructure** (p. 1133) Type as defined in CommandTypes.h.

Returns

The type of the data structure

Implements **activemq::commands::DataStructure** (p. 1137).

6.297.2.5 `virtual const Pointer<TransactionId>& activemq::commands::JournalTransaction::getTransactionId () const`
[virtual]

6.297.2.6 `virtual Pointer<TransactionId>& activemq::commands::JournalTransaction::getTransactionId ()`
[virtual]

6.297.2.7 `virtual unsigned char activemq::commands::JournalTransaction::getType () const` [virtual]

6.297.2.8 `virtual bool activemq::commands::JournalTransaction::getWasPrepared () const` [virtual]

6.297.2.9 `virtual void activemq::commands::JournalTransaction::setTransactionId (const Pointer< TransactionId > & transactionId)` [virtual]

6.297.2.10 `virtual void activemq::commands::JournalTransaction::setType (unsigned char type)` [virtual]

6.297.2.11 `virtual void activemq::commands::JournalTransaction::setWasPrepared (bool wasPrepared)` [virtual]

6.297.2.12 `virtual std::string activemq::commands::JournalTransaction::toString () const` [virtual]

Returns a string containing the information for this **DataStructure** (p. 1133) such as its type and value of its elements.

Returns

formatted string useful for debugging.

Reimplemented from **activemq::commands::BaseDataStructure** (p. 531).

6.297.3 Field Documentation

6.297.3.1 `const unsigned char activemq::commands::JournalTransaction::ID_JOURNALTRANSACTION = 54`
[static]

6.297.3.2 **Pointer<TransactionId> activemq::commands::JournalTransaction-
::transactionId** [protected]

6.297.3.3 **unsigned char activemq::commands::JournalTransaction::type**
[protected]

6.297.3.4 **bool activemq::commands::JournalTransaction::wasPrepared**
[protected]

The documentation for this class was generated from the following file:

- src/main/activemq/commands/**JournalTransaction.h**

6.298 **activemq::wireformat::openwire::marshal::generated::- JournalTransactionMarshaller Class Reference**

Marshaling code for Open Wire Format for **JournalTransactionMarshaller** (p. 1587).

```
#include <src/main/activemq/wireformat/openwire/marshal/generated/-  
JournalTransactionMarshaller.h>
```

Inheritance diagram for activemq::wireformat::openwire::marshal::generated::JournalTransactionMarshaller:

Public Member Functions

- **JournalTransactionMarshaller ()**
- virtual **~JournalTransactionMarshaller ()**
- virtual **commands::DataStructure * createObject ()** const
Creates a new instance of the class that this class is a marshaling director for.
- virtual unsigned char **getDataStructureType ()** const
Gets the DataStructureType that this class marshals/unmarshals.
- virtual void **tightUnmarshal (OpenWireFormat *wireFormat, commands::DataStructure *dataStructure, decaf::io::DataInputStream *dataIn, utils::BooleanStream *bs)**
Tight Un-marhsal to the given stream.
- virtual int **tightMarshal1 (OpenWireFormat *wireFormat, commands::DataStructure *dataStructure, utils::BooleanStream *bs)**
Tight Marhsal to the given stream.
- virtual void **tightMarshal2 (OpenWireFormat *wireFormat, commands::DataStructure *dataStructure, decaf::io::DataOutputStream *dataOut, utils::BooleanStream *bs)**
Tight Marhsal to the given stream.

6.298 activemq::wireformat::openwire::marshal::generated::JournalTransaction-Marshaller Class

Reference

1593

- virtual void **looseUnmarshal** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataInputStream** *dataIn)
Loose Un-marshal to the given stream.
- virtual void **looseMarshal** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataOutputStream** *dataOut)
Tight Marshal to the given stream.

6.298.1 Detailed Description

Marshaling code for Open Wire Format for **JournalTransactionMarshaller** (p. 1587).

NOTE!: This file is auto generated - do not modify! if you need to make a change, please see the Java Classes in the activemq-openwire-generator module

6.298.2 Constructor & Destructor Documentation

6.298.2.1 **activemq::wireformat::openwire::marshal::generated::JournalTransactionMarshaller::JournalTransactionMarshaller** ()
[inline]

6.298.2.2 **virtual activemq::wireformat::openwire::marshal::generated::JournalTransactionMarshaller::~~JournalTransactionMarshaller** ()
[inline, virtual]

6.298.3 Member Function Documentation

6.298.3.1 **virtual commands::DataStructure* activemq::wireformat::openwire::marshal::generated::JournalTransactionMarshaller::createObject** ()
const [virtual]

Creates a new instance of the class that this class is a marshaling director for.

Returns

newly allocated Command

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1120).

6.298.3.2 **virtual unsigned char activemq::wireformat::openwire::marshal::generated::JournalTransactionMarshaller::getDataStructureType** () **const**
[virtual]

Gets the DataStructureType that this class marshals/unmarshals.

Returns

byte Id of this classes DataStructureType

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller**
(p. 1122).

6.298.3.3 **virtual void activemq::wireformat::openwire::marshal::generated::JournalTransactionMarshaller::looseMarshal (OpenWireFormat * *format*, commands::DataStructure * *command*, decaf::io::DataOutputStream * *ds*) [virtual]**

Tight Marshal to the given stream.

Parameters

<i>format</i>	- The OpenWireFormat properties
<i>command</i>	- the object to Marshal
<i>ds</i>	- DataOutputStream to marshal to

Exceptions

<i>IOException</i>	if an error occurs.
--------------------	---------------------

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller**
(p. 1123).

6.298.3.4 **virtual void activemq::wireformat::openwire::marshal::generated::JournalTransactionMarshaller::looseUnmarshal (OpenWireFormat * *format*, commands::DataStructure * *command*, decaf::io::DataInputStream * *dis*) [virtual]**

Loose Un-marshal to the given stream.

Parameters

<i>format</i>	- The OpenWireFormat properties
<i>command</i>	- the object to Un-Marshal
<i>dis</i>	- the DataInputStream to Un-Marshal from

Exceptions

<i>IOException</i>	if an error occurs.
--------------------	---------------------

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller**
(p. 1125).

6.298 activemq::wireformat::openwire::marshal::generated::JournalTransaction-Marshaller Class

Reference

1595

```
6.298.3.5 virtual int activemq::wireformat::openwire::marshal::generated::-  
    JournalTransactionMarshaller::tightMarshal1 ( OpenWireFormat *  
        format, commands::DataStructure * command, utils::BooleanStream * bs  
    ) [virtual]
```

Tight Marhsal to the given stream.

Parameters

<i>format</i>	The OpenwireFormat properties
<i>command</i>	The object to Marshal
<i>bs</i>	The boolean stream to marshal to.

Exceptions

<i>IOException</i>	if an error occurs.
--------------------	---------------------

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller**
(p. 1127).

```
6.298.3.6 virtual void activemq::wireformat::openwire::marshal-  
    ::generated::JournalTransactionMarshaller::tightMarshal2 (   
    OpenWireFormat * format, commands::DataStructure * command,  
    decaf::io::DataOutputStream * ds, utils::BooleanStream * bs )  
    [virtual]
```

Tight Marhsal to the given stream.

Parameters

<i>format</i>	- The OpenwireFormat properties
<i>command</i>	- the object to Marshal
<i>ds</i>	- the DataOutputStream to Marshal to
<i>bs</i>	- boolean stream to marshal to.

Exceptions

<i>IOException</i>	if an error occurs.
--------------------	---------------------

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller**
(p. 1129).

```
6.298.3.7 virtual void activemq::wireformat::openwire::marshal-
::generated::JournalTransactionMarshaller::tightUnmarshal (
OpenWireFormat * format, commands::DataStructure * command,
decaf::io::DataInputStream * dis, utils::BooleanStream * bs )
[virtual]
```

Tight Un-marhsal to the given stream.

Parameters

<i>format</i>	- The OpenwireFormat properties
<i>command</i>	- the object to Un-Marshal
<i>dis</i>	- the DataInputStream to Un-Marshal from
<i>bs</i>	- boolean stream to unmarshal from.

Exceptions

<i>IOException</i>	if an error occurs.
--------------------	---------------------

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1131).

The documentation for this class was generated from the following file:

- src/main/activemq/wireformat/openwire/marshal/generated/**JournalTransactionMarshaller.h**

6.299 activemq::commands::KeepAliveInfo Class Reference

```
#include <src/main/activemq/commands/KeepAliveInfo.h>
```

Inheritance diagram for activemq::commands::KeepAliveInfo:

Public Member Functions

- **KeepAliveInfo** ()
- virtual **~KeepAliveInfo** ()
- virtual unsigned char **getDataStructureType** () const
*Get the **DataStructure** (p. 1133) Type as defined in CommandTypes.h.*
- virtual **KeepAliveInfo** * **cloneDataStructure** () const
Clone this obbect and return a new instance that the caller now owns, this will be an exact copy of this one.
- virtual void **copyDataStructure** (const **DataStructure** *src)
Copy the contents of the passed object into this objects members, overwriting any existing data.

- virtual std::string **toString** () const
*Returns a string containing the information for this **DataStructure** (p. 1133) such as its type and value of its elements.*
- virtual bool **equals** (const **DataStructure** *value) const
*Compares the **DataStructure** (p. 1133) passed in to this one, and returns if they are equivalent.*
- virtual bool **isKeepAliveInfo** () const
- virtual **Pointer**< **Command** > **visit** (activemq::state::CommandVisitor *visitor)
Allows a Visitor to visit this command and return a response to the command based on the command type being visited.

Static Public Attributes

- static const unsigned char **ID_KEEPAVAILABLEINFO** = 10

6.299.1 Constructor & Destructor Documentation

6.299.1.1 **activemq::commands::KeepAliveInfo::KeepAliveInfo** ()

6.299.1.2 **virtual activemq::commands::KeepAliveInfo::~~KeepAliveInfo** ()
[virtual]

6.299.2 Member Function Documentation

6.299.2.1 **virtual KeepAliveInfo* activemq::commands::KeepAliveInfo::cloneData-Structure** () const [virtual]

Clone this object and return a new instance that the caller now owns, this will be an exact copy of this one.

Returns

new copy of this object.

Implements **activemq::commands::DataStructure** (p. 1133).

6.299.2.2 **virtual void activemq::commands::KeepAliveInfo::copyDataStructure** (const **DataStructure** * src) [virtual]

Copy the contents of the passed object into this objects members, overwriting any existing data.

Returns

src - Source Object

Reimplemented from **activemq::commands::BaseCommand** (p. 493).

6.299.2.3 `virtual bool activemq::commands::KeepAliveInfo::equals (const
DataStructure * value) const [virtual]`

Compares the **DataStructure** (p. 1133) passed in to this one, and returns if they are equivalent.

Equivalent here means that they are of the same type, and that each element of the objects are the same.

Returns

true if **DataStructure** (p. 1133)'s are Equal.

Reimplemented from **activemq::commands::BaseCommand** (p. 494).

6.299.2.4 `virtual unsigned char activemq::commands::KeepAliveInfo::getData-
StructureType () const [virtual]`

Get the **DataStructure** (p. 1133) Type as defined in CommandTypes.h.

Returns

The type of the data structure

Implements **activemq::commands::DataStructure** (p. 1137).

6.299.2.5 `virtual bool activemq::commands::KeepAliveInfo::isKeepAliveInfo ()
const [inline, virtual]`

Returns

an answer of true to the **isKeepAliveInfo()** (p. 1593) query.

Reimplemented from **activemq::commands::BaseCommand** (p. 495).

6.299.2.6 `virtual std::string activemq::commands::KeepAliveInfo::toString () const
[virtual]`

Returns a string containing the information for this **DataStructure** (p. 1133) such as its type and value of its elements.

Returns

formatted string useful for debugging.

Reimplemented from **activemq::commands::BaseCommand** (p. 498).

6.300

activemq::wireformat::openwire::marshal::generated::KeepAliveInfoMarshaller

Class Reference

1599

6.299.2.7 virtual **Pointer**<**Command**> **activemq::commands::KeepAliveInfo::visit**(
 activemq::state::CommandVisitor * *visitor*) [virtual]

Allows a Visitor to visit this command and return a response to the command based on the command type being visited.

The command will call the proper processXXX method in the visitor.

Returns

a **Response** (p. 2298) to the visitor being called or NULL if no response.

Implements **activemq::commands::Command** (p. 871).

6.299.3 Field Documentation

6.299.3.1 const unsigned char **activemq::commands::KeepAliveInfo::ID_KEEPALIVEINFO** = 10 [static]

The documentation for this class was generated from the following file:

- src/main/activemq/commands/**KeepAliveInfo.h**

6.300 **activemq::wireformat::openwire::marshal::generated::KeepAliveInfoMarshaller** Class Reference

Marshaling code for Open Wire Format for **KeepAliveInfoMarshaller** (p. 1594).

```
#include <src/main/activemq/wireformat/openwire/marshal/generated/-  
KeepAliveInfoMarshaller.h>
```

Inheritance diagram for **activemq::wireformat::openwire::marshal::generated::KeepAliveInfoMarshaller**:

Public Member Functions

- **KeepAliveInfoMarshaller** ()
- virtual **~KeepAliveInfoMarshaller** ()
- virtual **commands::DataStructure** * **createObject** () const
 Creates a new instance of the class that this class is a marshaling director for.
- virtual unsigned char **getDataStructureType** () const
 Gets the DataStructureType that this class marshals/unmarshals.
- virtual void **tightUnmarshal** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataInputStream** *dataIn, **utils::BooleanStream** *bs)

Tight Un-marhsal to the given stream.

- virtual int **tightMarshal1** (**OpenWireFormat** *wireFormat, **commands::Data-Structure** *dataStructure, **utils::BooleanStream** *bs)

Tight Marhsal to the given stream.

- virtual void **tightMarshal2** (**OpenWireFormat** *wireFormat, **commands::Data-Structure** *dataStructure, **decaf::io::DataOutputStream** *dataOut, **utils::BooleanStream** *bs)

Tight Marhsal to the given stream.

- virtual void **looseUnmarshal** (**OpenWireFormat** *wireFormat, **commands::Data-Structure** *dataStructure, **decaf::io::DataInputStream** *dataIn)

Loose Un-marhsal to the given stream.

- virtual void **looseMarshal** (**OpenWireFormat** *wireFormat, **commands::Data-Structure** *dataStructure, **decaf::io::DataOutputStream** *dataOut)

Tight Marhsal to the given stream.

6.300.1 Detailed Description

Marshaling code for Open Wire Format for **KeepAliveInfoMarshaller** (p. 1594).

NOTE!: This file is auto generated - do not modify! if you need to make a change, please see the Java Classes in the activemq-openwire-generator module

6.300.2 Constructor & Destructor Documentation

6.300.2.1 **activemq::wireformat::openwire::marshal::generated::KeepAliveInfoMarshaller::KeepAliveInfoMarshaller ()**
[inline]

6.300.2.2 **virtual activemq::wireformat::openwire::marshal::generated::KeepAliveInfoMarshaller::~~KeepAliveInfoMarshaller ()** [inline, virtual]

6.300.3 Member Function Documentation

6.300.3.1 **virtual commands::DataStructure* activemq::wireformat::openwire::marshal::generated::KeepAliveInfoMarshaller::createObject ()** const
[virtual]

Creates a new instance of the class that this class is a marshaling director for.

Returns

newly allocated Command

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1120).

6.300

activemq::wireformat::openwire::marshal::generated::KeepAliveInfoMarshaller
Class Reference 1601

6.300.3.2 `virtual unsigned char activemq::wireformat::openwire::marshal-
::generated::KeepAliveInfoMarshaller::getDataStructureType () const
[virtual]`

Gets the DataStructureType that this class marshals/unmarshals.

Returns

byte Id of this classes DataStructureType

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller**
(p. 1122).

6.300.3.3 `virtual void activemq::wireformat::openwire::marshal::generated::-
KeepAliveInfoMarshaller::looseMarshal (OpenWireFormat * format,
commands::DataStructure * command, decaf::io::DataOutputStream * ds
) [virtual]`

Tight Marhsal to the given stream.

Parameters

<i>format</i>	- The OpenWireFormat properties
<i>command</i>	- the object to Marshal
<i>ds</i>	- DataOutputStream to marshal to

Exceptions

<i>IOException</i>	if an error occurs.
--------------------	---------------------

Reimplemented from **activemq::wireformat::openwire::marshal::generated::Base-
CommandMarshaller** (p. 500).

6.300.3.4 `virtual void activemq::wireformat::openwire::marshal::generated::Keep-
AliveInfoMarshaller::looseUnmarshal (OpenWireFormat * format,
commands::DataStructure * command, decaf::io::DataInputStream * dis)
[virtual]`

Loose Un-marhsal to the given stream.

Parameters

<i>format</i>	- The OpenWireFormat properties
<i>command</i>	- the object to Un-Marshal
<i>dis</i>	- the DataInputStream to Un-Marshal from

Exceptions

<i>IOException</i>	if an error occurs.
--------------------	---------------------

Reimplemented from **activemq::wireformat::openwire::marshal::generated::Base-CommandMarshaller** (p. 501).

6.300.3.5 **virtual int activemq::wireformat::openwire::marshal::generated::KeepAliveInfoMarshaller::tightMarshal1 (OpenWireFormat * *format*, commands::DataStructure * *command*, utils::BooleanStream * *bs*)**
[virtual]

Tight Marshal to the given stream.

Parameters

<i>format</i>	The OpenWireFormat properties
<i>command</i>	The object to Marshal
<i>bs</i>	The boolean stream to marshal to.

Exceptions

<i>IOException</i>	if an error occurs.
--------------------	---------------------

Reimplemented from **activemq::wireformat::openwire::marshal::generated::Base-CommandMarshaller** (p. 503).

6.300.3.6 **virtual void activemq::wireformat::openwire::marshal::generated::KeepAliveInfoMarshaller::tightMarshal2 (OpenWireFormat * *format*, commands::DataStructure * *command*, decaf::io::DataOutputStream * *ds*, utils::BooleanStream * *bs*)** [virtual]

Tight Marshal to the given stream.

Parameters

<i>format</i>	- The OpenWireFormat properties
<i>command</i>	- the object to Marshal
<i>ds</i>	- the DataOutputStream to Marshal to
<i>bs</i>	- boolean stream to marshal to.

Exceptions

<i>IOException</i>	if an error occurs.
--------------------	---------------------

Reimplemented from **activemq::wireformat::openwire::marshal::generated::Base-CommandMarshaller** (p. 504).

6.300.3.7 virtual void **activemq::wireformat::openwire::marshal::generated::KeepAliveInfoMarshaller::tightUnmarshal** (**OpenWireFormat** * *format*, **commands::DataStructure** * *command*, **decaf::io::DataInputStream** * *dis*, **utils::BooleanStream** * *bs*) [virtual]

Tight Un-marhsal to the given stream.

Parameters

<i>format</i>	- The OpenWireFormat properties
<i>command</i>	- the object to Un-Marshal
<i>dis</i>	- the DataInputStream to Un-Marshal from
<i>bs</i>	- boolean stream to unmarshal from.

Exceptions

<i>IOException</i>	if an error occurs.
--------------------	---------------------

Reimplemented from **activemq::wireformat::openwire::marshal::generated::BaseCommandMarshaller** (p. 505).

The documentation for this class was generated from the following file:

- src/main/activemq/wireformat/openwire/marshal/generated/**KeepAliveInfoMarshaller.h**

6.301 decaf::security::Key Class Reference

The **Key** (p. 1598) interface is the top-level interface for all keys.

```
#include <src/main/decaf/security/Key.h>
```

Inheritance diagram for decaf::security::Key:

Public Member Functions

- virtual **~Key** ()
- virtual std::string **getAlgorithm** () const =0
Returns the standard algorithm name for this key.
- virtual void **getEncoded** (std::vector< unsigned char > &output) const =0
Provides the key in its primary encoding format, or nothing if this key does not support encoding.
- virtual std::string **getFormat** () const =0
Returns the name of the primary encoding format of this key, or an empty string if this key does not support encoding.

6.301.1 Detailed Description

The **Key** (p. 1598) interface is the top-level interface for all keys.

It defines the functionality shared by all key objects. All keys have three characteristics:

An Algorithm

This is the key algorithm for that key. The key algorithm is usually an encryption or asymmetric operation algorithm (such as DSA or RSA), which will work with those algorithms and with related algorithms (such as MD5 with RSA, SHA-1 with RSA, Raw DSA, etc.) The name of the algorithm of a key is obtained using the `getAlgorithm` method.

An Encoded Form

This is an external encoded form for the key used when a standard representation of the key is needed outside the application, as when transmitting the key to some other party. The key is encoded according to a standard format (such as X.509 SubjectPublicKeyInfo or PKCS#8), and is returned using the `getEncoded` method. Note: The syntax of the ASN.1 type SubjectPublicKeyInfo is defined as follows:

```
SubjectPublicKeyInfo ::= SEQUENCE {
    algorithm AlgorithmIdentifier,
    subjectPublicKey BIT STRING }

AlgorithmIdentifier ::= SEQUENCE {
    algorithm OBJECT IDENTIFIER,
    parameters ANY DEFINED BY algorithm OPTIONAL }
```

For more information, see RFC 2459: Internet X.509 Public **Key** (p. 1598) Infrastructure Certificate and CRL Profile.

A Format

This is the name of the format of the encoded key. It is returned by the `getFormat` method.

6.301.2 Constructor & Destructor Documentation

6.301.2.1 `virtual decaf::security::Key::~~Key() [inline, virtual]`

6.301.3 Member Function Documentation

6.301.3.1 `virtual std::string decaf::security::Key::getAlgorithm() const [pure virtual]`

Returns the standard algorithm name for this key.

For example, "DSA" would indicate that this key is a DSA key.

Returns

the name of the algorithm associated with this key.

6.301.3.2 `virtual void decaf::security::Key::getEncoded (std::vector< unsigned char > & output) const [pure virtual]`

Provides the key in its primary encoding format, or nothing if this key does not support encoding.

Parameters

<i>output</i>	Receives the encoded key, or nothing if the key does not support encoding.
---------------	--

6.301.3.3 `virtual std::string decaf::security::Key::getFormat () const [pure virtual]`

Returns the name of the primary encoding format of this key, or an empty string if this key does not support encoding.

The primary encoding format is named in terms of the appropriate ASN.1 data format, if an ASN.1 specification for this key exists. For example, the name of the ASN.1 data format for public keys is SubjectPublicKeyInfo, as defined by the X.509 standard; in this case, the returned format is "X.509". Similarly, the name of the ASN.1 data format for private keys is PrivateKeyInfo, as defined by the PKCS #8 standard; in this case, the returned format is "PKCS#8".

Returns

the primary encoding format of the key.

The documentation for this class was generated from the following file:

- src/main/decaf/security/**Key.h**

6.302 decaf::security::KeyException Class Reference

```
#include <src/main/decaf/security/KeyException.h>
```

Inheritance diagram for decaf::security::KeyException:

Public Member Functions

- **KeyException** () throw ()

Default Constructor.

- **KeyException** (const **decaf::lang::Exception** &ex) throw ()

Conversion Constructor from some other Exception.

- **KeyException** (const **KeyException** &ex) throw ()

Copy Constructor.

- **KeyException** (const char *file, const int lineNumber, const std::exception ***cause**, const char *msg,...) throw ()

Constructor - Initializes the file name and line number where this message occurred.

- **KeyException** (const std::exception ***cause**) throw ()

Constructor.

- **KeyException** (const char *file, const int lineNumber, const char *msg,...) throw ()

Constructor - Initializes the file name and line number where this message occurred.

- virtual **KeyException** * **clone** () const

Clones this exception.

- virtual ~**KeyException** () throw ()

6.302.1 Constructor & Destructor Documentation

6.302.1.1 **decaf::security::KeyException::KeyException () throw ()** [inline]

Default Constructor.

6.302.1.2 **decaf::security::KeyException::KeyException (const decaf::lang::Exception & ex) throw ()** [inline]

Conversion Constructor from some other Exception.

Parameters

<i>ex</i>	An exception that should become this type of Exception
-----------	--

6.302.1.3 **decaf::security::KeyException::KeyException (const KeyException & ex) throw ()** [inline]

Copy Constructor.

Parameters

<i>ex</i>	An exception that should become this type of Exception
-----------	--

6.302.1.4 **decaf::security::KeyException::KeyException** (const char * *file*, const int *lineNumber*, const std::exception * *cause*, const char * *msg*, ...) throw ()
[inline]

Constructor - Initializes the file name and line number where this message occurred.

Sets the message to report, using an optional list of arguments to parse into the message

Parameters

<i>file</i>	The file name where exception occurs
<i>lineNumber</i>	The line number where the exception occurred.
<i>cause</i>	The exception that was the cause for this one to be thrown.
<i>msg</i>	The message to report
...	list of primitives that are formatted into the message

6.302.1.5 **decaf::security::KeyException::KeyException** (const std::exception * *cause*) throw () [inline]

Constructor.

Parameters

<i>cause</i>	Pointer to the exception that caused this one to be thrown, the object is cloned caller retains ownership.
--------------	--

6.302.1.6 **decaf::security::KeyException::KeyException** (const char * *file*, const int *lineNumber*, const char * *msg*, ...) throw () [inline]

Constructor - Initializes the file name and line number where this message occurred.

Sets the message to report, using an optional list of arguments to parse into the message

Parameters

<i>file</i>	name where exception occurs
<i>lineNumber</i>	line number where the exception occurred.
<i>msg</i>	message to report
...	list of primitives that are formatted into the message

6.302.1.7 **virtual decaf::security::KeyException::~KeyException** () throw ()
[inline, virtual]

6.302.2 Member Function Documentation

6.302.2.1 **virtual KeyException* decaf::security::KeyException::clone () const**
 [inline, virtual]

Clones this exception.

This is useful for cases where you need to preserve the type of the original exception as well as the message. All subclasses should override.

Returns

A deep copy of this exception.

Reimplemented from **decaf::security::GeneralSecurityException** (p. 1397).

Reimplemented in **decaf::security::InvalidKeyException** (p. 1538), and **decaf::security::KeyManagementException** (p. 1605).

The documentation for this class was generated from the following file:

- src/main/decaf/security/**KeyException.h**

6.303 decaf::security::KeyManagementException Class Reference

```
#include <src/main/decaf/security/KeyManagementException.h>
```

Inheritance diagram for decaf::security::KeyManagementException:

Public Member Functions

- **KeyManagementException** () throw ()
Default Constructor.
- **KeyManagementException** (const **Exception** &ex) throw ()
Conversion Constructor from some other Exception.
- **KeyManagementException** (const **KeyManagementException** &ex) throw ()
Copy Constructor.
- **KeyManagementException** (const char *file, const int lineNumber, const std::exception *cause, const char *msg,...) throw ()
Constructor - Initializes the file name and line number where this message occurred.
- **KeyManagementException** (const std::exception *cause) throw ()
Constructor.
- **KeyManagementException** (const char *file, const int lineNumber, const char *msg,...) throw ()
Constructor - Initializes the file name and line number where this message occurred.
- virtual **KeyManagementException** * **clone** () const
Clones this exception.
- virtual ~**KeyManagementException** () throw ()

6.303.1 Constructor & Destructor Documentation

6.303.1.1 **decaf::security::KeyManagementException::KeyManagementException**
() throw () [inline]

Default Constructor.

6.303.1.2 **decaf::security::KeyManagementException::KeyManagementException**
(const Exception & ex) throw () [inline]

Conversion Constructor from some other Exception.

Parameters

<i>ex</i>	An exception that should become this type of Exception
-----------	--

6.303.1.3 **decaf::security::KeyManagementException::KeyManagementException**
(const KeyManagementException & ex) throw () [inline]

Copy Constructor.

Parameters

<i>ex</i>	An exception that should become this type of Exception
-----------	--

6.303.1.4 **decaf::security::KeyManagementException::KeyManagementException**
(const char * *file*, const int *lineNumber*, const std::exception * *cause*, const char * *msg*, ...) throw () [inline]

Constructor - Initializes the file name and line number where this message occurred.

Sets the message to report, using an optional list of arguments to parse into the message

Parameters

<i>file</i>	The file name where exception occurs
<i>lineNumber</i>	The line number where the exception occurred.
<i>cause</i>	The exception that was the cause for this one to be thrown.
<i>msg</i>	The message to report
...	list of primitives that are formatted into the message

6.303.1.5 **decaf::security::KeyManagementException::KeyManagementException**
(const std::exception * *cause*) throw () [inline]

Constructor.

Parameters

<i>cause</i>	Pointer to the exception that caused this one to be thrown, the object is cloned caller retains ownership.
--------------	--

6.303.1.6 `decaf::security::KeyManagementException::KeyManagementException (const char * file, const int lineNumber, const char * msg, ...) throw () [inline]`

Constructor - Initializes the file name and line number where this message occurred.

Sets the message to report, using an optional list of arguments to parse into the message

Parameters

<i>file</i>	name where exception occurs
<i>lineNumber</i>	line number where the exception occurred.
<i>msg</i>	message to report
<i>...</i>	list of primitives that are formatted into the message

6.303.1.7 `virtual decaf::security::KeyManagementException::~~KeyManagementException () throw () [inline, virtual]`

6.303.2 Member Function Documentation

6.303.2.1 `virtual KeyManagementException* decaf::security::KeyManagementException::clone () const [inline, virtual]`

Clones this exception.

This is useful for cases where you need to preserve the type of the original exception as well as the message. All subclasses should override.

Returns

A deep copy of this exception.

Reimplemented from **decaf::security::KeyException** (p. 1603).

The documentation for this class was generated from the following file:

- `src/main/decaf/security/KeyManagementException.h`

6.304 activemq::commands::LastPartialCommand Class Reference

```
#include <src/main/activemq/commands/LastPartialCommand.h>
```

Inheritance diagram for activemq::commands::LastPartialCommand:

Public Member Functions

- **LastPartialCommand** ()
- virtual **~LastPartialCommand** ()
- virtual unsigned char **getDataStructureType** () const
*Get the **DataStructure** (p. 1133) Type as defined in CommandTypes.h.*
- virtual **LastPartialCommand * cloneDataStructure** () const
Clone this object and return a new instance that the caller now owns, this will be an exact copy of this one.
- virtual void **copyDataStructure** (const **DataStructure** *src)
Copy the contents of the passed object into this objects members, overwriting any existing data.
- virtual std::string **toString** () const
*Returns a string containing the information for this **DataStructure** (p. 1133) such as its type and value of its elements.*
- virtual bool **equals** (const **DataStructure** *value) const
*Compares the **DataStructure** (p. 1133) passed in to this one, and returns if they are equivalent.*

Static Public Attributes

- static const unsigned char **ID_LASTPARTIALCOMMAND** = 61

6.304.1 Constructor & Destructor Documentation

6.304.1.1 **activemq::commands::LastPartialCommand::LastPartialCommand** ()

6.304.1.2 **virtual activemq::commands::LastPartialCommand::~~LastPartialCommand** () [virtual]

6.304.2 Member Function Documentation

6.304.2.1 **virtual LastPartialCommand* activemq::commands::LastPartialCommand::cloneDataStructure** () const [virtual]

Clone this object and return a new instance that the caller now owns, this will be an exact copy of this one.

Returns

new copy of this object.

Reimplemented from **activemq::commands::PartialCommand** (p. 2077).

```
6.304.2.2  virtual void activemq::commands::LastPartialCommand-  
           ::copyDataStructure ( const DataStructure * src )  
           [virtual]
```

Copy the contents of the passed object into this objects members, overwriting any existing data.

Returns

src - Source Object

Reimplemented from **activemq::commands::PartialCommand** (p. 2077).

```
6.304.2.3  virtual bool activemq::commands::LastPartialCommand::equals ( const  
           DataStructure * value ) const [virtual]
```

Compares the **DataStructure** (p. 1133) passed in to this one, and returns if they are equivalent.

Equivalent here means that they are of the same type, and that each element of the objects are the same.

Returns

true if **DataStructure** (p. 1133)'s are Equal.

Reimplemented from **activemq::commands::PartialCommand** (p. 2078).

```
6.304.2.4  virtual unsigned char activemq::commands::Last-  
           PartialCommand::getDataStructureType ( ) const  
           [virtual]
```

Get the **DataStructure** (p. 1133) Type as defined in CommandTypes.h.

Returns

The type of the data structure

Reimplemented from **activemq::commands::PartialCommand** (p. 2078).

6.305 activemq::wireformat::openwire::marshal::generated::LastPartialCommandMarshaller Class

Reference

1613

6.304.2.5 virtual std::string activemq::commands::LastPartialCommand::toString()
) const [virtual]

Returns a string containing the information for this **DataSet** (p. 1133) such as its type and value of its elements.

Returns

formatted string useful for debugging.

Reimplemented from **activemq::commands::PartialCommand** (p. 2078).

6.304.3 Field Documentation

6.304.3.1 const unsigned char activemq::commands::LastPartialCommand::ID_LASTPARTIALCOMMAND = 61
[static]

The documentation for this class was generated from the following file:

- src/main/activemq/commands/**LastPartialCommand.h**

6.305 activemq::wireformat::openwire::marshal::generated::LastPartialCommandMarshaller Class Reference

Marshaling code for Open Wire Format for **LastPartialCommandMarshaller** (p. 1608).

```
#include <src/main/activemq/wireformat/openwire/marshal/generated/-  
LastPartialCommandMarshaller.h>
```

Inheritance diagram for activemq::wireformat::openwire::marshal::generated::LastPartialCommandMarshaller:

Public Member Functions

- **LastPartialCommandMarshaller** ()
- virtual ~**LastPartialCommandMarshaller** ()
- virtual **commands::DataSet** * **createObject** () const
Creates a new instance of the class that this class is a marshaling director for.
- virtual unsigned char **getDataSetType** () const
Gets the DataSetType that this class marshals/unmarshals.
- virtual void **tightUnmarshal** (**OpenWireFormat** *wireFormat, **commands::DataSet** *dataStructure, **decaf::io::DataInputStream** *dataIn, **utils::BooleanStream** *bs)

Tight Un-marhsal to the given stream.

- virtual int **tightMarshal1** (**OpenWireFormat** *wireFormat, **commands::Data-Structure** *dataStructure, **utils::BooleanStream** *bs)

Tight Marhsal to the given stream.

- virtual void **tightMarshal2** (**OpenWireFormat** *wireFormat, **commands::Data-Structure** *dataStructure, **decaf::io::DataOutputStream** *dataOut, **utils::BooleanStream** *bs)

Tight Marhsal to the given stream.

- virtual void **looseUnmarshal** (**OpenWireFormat** *wireFormat, **commands::Data-Structure** *dataStructure, **decaf::io::DataInputStream** *dataIn)

Loose Un-marhsal to the given stream.

- virtual void **looseMarshal** (**OpenWireFormat** *wireFormat, **commands::Data-Structure** *dataStructure, **decaf::io::DataOutputStream** *dataOut)

Tight Marhsal to the given stream.

6.305.1 Detailed Description

Marshaling code for Open Wire Format for **LastPartialCommandMarshaller** (p. 1608).

NOTE!: This file is auto generated - do not modify! if you need to make a change, please see the Java Classes in the activemq-openwire-generator module

6.305.2 Constructor & Destructor Documentation

6.305.2.1 **activemq::wireformat::openwire::marshal::generated::Last-PartialCommandMarshaller::LastPartialCommandMarshaller** ()
[inline]

6.305.2.2 **virtual activemq::wireformat::openwire::marshal::generated::Last-PartialCommandMarshaller::~~LastPartialCommandMarshaller** ()
[inline, virtual]

6.305.3 Member Function Documentation

6.305.3.1 **virtual commands::DataStructure* activemq::wireformat::openwire-::marshal::generated::LastPartialCommandMarshaller::createObject** ()
const [virtual]

Creates a new instance of the class that this class is a marshaling director for.

Returns

newly allocated Command

Reimplemented from **activemq::wireformat::openwire::marshal::generated::-PartialCommandMarshaller** (p. 2080).

6.305 activemq::wireformat::openwire::marshal::generated::LastPartialCommandMarshaller Class

Reference

1615

6.305.3.2 `virtual unsigned char activemq::wireformat::openwire::marshal::generated::LastPartialCommandMarshaller::getDataType () const`
[virtual]

Gets the DataType that this class marshals/unmarshals.

Returns

byte Id of this classes DataType

Reimplemented from **activemq::wireformat::openwire::marshal::generated::LastPartialCommandMarshaller** (p. 2081).

6.305.3.3 `virtual void activemq::wireformat::openwire::marshal::generated::LastPartialCommandMarshaller::looseMarshal (OpenWireFormat * format, commands::DataStructure * command, decaf::io::DataOutputStream * ds)` [virtual]

Tight Marshal to the given stream.

Parameters

<i>format</i>	- The OpenWireFormat properties
<i>command</i>	- the object to Marshal
<i>ds</i>	- DataOutputStream to marshal to

Exceptions

<i>IOException</i>	if an error occurs.
--------------------	---------------------

Reimplemented from **activemq::wireformat::openwire::marshal::generated::LastPartialCommandMarshaller** (p. 2081).

6.305.3.4 `virtual void activemq::wireformat::openwire::marshal::generated::LastPartialCommandMarshaller::looseUnmarshal (OpenWireFormat * format, commands::DataStructure * command, decaf::io::DataInputStream * dis)` [virtual]

Loose Un-marshal to the given stream.

Parameters

<i>format</i>	- The OpenWireFormat properties
<i>command</i>	- the object to Un-Marshal
<i>dis</i>	- the DataInputStream to Un-Marshal from

Exceptions

<i>IOException</i>	if an error occurs.
--------------------	---------------------

Reimplemented from **activemq::wireformat::openwire::marshal::generated::PartialCommandMarshaller** (p. 2081).

6.305.3.5 **virtual int activemq::wireformat::openwire::marshal::generated::LastPartialCommandMarshaller::tightMarshal1 (OpenWireFormat * *format*, commands::DataStructure * *command*, utils::BooleanStream * *bs*)**
[virtual]

Tight Marshal to the given stream.

Parameters

<i>format</i>	The OpenWireFormat properties
<i>command</i>	The object to Marshal
<i>bs</i>	The boolean stream to marshal to.

Exceptions

<i>IOException</i>	if an error occurs.
--------------------	---------------------

Reimplemented from **activemq::wireformat::openwire::marshal::generated::PartialCommandMarshaller** (p. 2082).

6.305.3.6 **virtual void activemq::wireformat::openwire::marshal::generated::LastPartialCommandMarshaller::tightMarshal2 (OpenWireFormat * *format*, commands::DataStructure * *command*, decaf::io::DataOutputStream * *ds*, utils::BooleanStream * *bs*)** [virtual]

Tight Marshal to the given stream.

Parameters

<i>format</i>	- The OpenWireFormat properties
<i>command</i>	- the object to Marshal
<i>ds</i>	- the DataOutputStream to Marshal to
<i>bs</i>	- boolean stream to marshal to.

Exceptions

<i>IOException</i>	if an error occurs.
--------------------	---------------------

Reimplemented from **activemq::wireformat::openwire::marshal::generated::PartialCommandMarshaller** (p. 2082).

6.305.3.7 virtual void activemq::wireformat::openwire::marshal::generated::LastPartialCommandMarshaller::tightUnmarshal (OpenWireFormat * *format*, commands::DataStructure * *command*, decaf::io::DataInputStream * *dis*, utils::BooleanStream * *bs*) [virtual]

Tight Un-marhsal to the given stream.

Parameters

<i>format</i>	- The OpenWireFormat properties
<i>command</i>	- the object to Un-Marshal
<i>dis</i>	- the DataInputStream to Un-Marshal from
<i>bs</i>	- boolean stream to unmarshal from.

Exceptions

<i>IOException</i>	if an error occurs.
--------------------	---------------------

Reimplemented from **activemq::wireformat::openwire::marshal::generated::LastPartialCommandMarshaller** (p. 2083).

The documentation for this class was generated from the following file:

- src/main/activemq/wireformat/openwire/marshal/generated/**LastPartialCommandMarshaller.h**

6.306 decaf::util::comparators::Less< E > Class Template - Reference

Simple **Less** (p. 1612) **Comparator** (p. 888) that compares to elements to determine if the first is less than the second.

```
#include <src/main/decaf/util/comparators/Less.h>
```

Inheritance diagram for decaf::util::comparators::Less< E >:

Public Member Functions

- **Less** ()
- virtual ~**Less** ()
- virtual bool **operator()** (const E &left, const E &right) const
Implementation of the Binary function interface as a means of allowing a **Comparator** (p. 888) to be passed to an STL **Map** (p. 1768) for use as the sorting criteria.
- virtual int **compare** (const E &o1, const E &o2) const
Compares its two arguments for order.

6.306.1 Detailed Description

`template<typename E>class decaf::util::comparators::Less< E >`

Simple **Less** (p. 1612) **Comparator** (p. 888) that compares to elements to determine if the first is less than the second.

This can be used in **Collection** (p. 851) classes to sort elements according to their natural ordering. By design the **Comparator** (p. 888)'s compare function return more information about comparison than the STL binary function's boolean compare operator. In this case the compare method will return

Since

1.0

6.306.2 Constructor & Destructor Documentation

6.306.2.1 `template<typename E > decaf::util::comparators::Less< E >::Less ()`
[inline]

6.306.2.2 `template<typename E > virtual decaf::util::comparators::Less< E >::~Less ()` [inline, virtual]

6.306.3 Member Function Documentation

6.306.3.1 `template<typename E > virtual int decaf::util::comparators::Less< E >::compare (const E & o1, const E & o2) const` [inline, virtual]

Compares its two arguments for order.

Returns a negative integer, zero, or a positive integer as the first argument is less than, equal to, or greater than the second.

The implementor must ensure that `sgn(compare(x, y)) == -sgn(compare(y, x))` for all `x` and `y`. (This implies that `compare(x, y)` must throw an exception if and only if `compare(y, x)` throws an exception.)

The implementor must also ensure that the relation is transitive: `((compare(x, y)>0) && (compare(y, z)>0))` implies `compare(x, z)>0`.

Finally, the implementer must ensure that `compare(x, y)==0` implies that `sgn(compare(x, z))==sgn(compare(y, z))` for all `z`.

It is generally the case, but not strictly required that `(compare(x, y)==0) == (x == y)`. Generally speaking, any comparator that violates this condition should clearly indicate this fact. The recommended language is "Note: this comparator imposes orderings that are inconsistent with equals."

Parameters

<code>o1</code>	The first object to be compared
<code>o2</code>	The second object to be compared

6.307 `std::less< decaf::lang::ArrayPointer< T > >` Struct Template Reference 619

Returns

a negative integer, zero, or a positive integer as the first argument is less than, equal to, or greater than the second.

Implements `decaf::util::Comparator< E >` (p. 889).

6.306.3.2 `template<typename E> virtual bool decaf::util::comparators::Less< E >::operator()(const E & left, const E & right) const [inline, virtual]`

Implementation of the Binary function interface as a means of allowing a **Comparator** (p. 888) to be passed to an STL **Map** (p. 1768) for use as the sorting criteria.

Parameters

<i>left</i>	The Left hand side operand.
<i>right</i>	The Right hand side operand.

Returns

true if the vale of left is less than the value of right.

Implements `decaf::util::Comparator< E >` (p. 890).

The documentation for this class was generated from the following file:

- `src/main/decaf/util/comparators/Less.h`

6.307 `std::less< decaf::lang::ArrayPointer< T > >` Struct - Template Reference

An override of the less function object so that the Pointer objects can be stored in STL Maps, etc.

```
#include <src/main/decaf/lang/ArrayPointer.h>
```

Public Member Functions

- `bool operator() (const decaf::lang::ArrayPointer< T > &left, const decaf::lang::ArrayPointer< T > &right) const`

6.307.1 Detailed Description

```
template<typename T>struct std::less< decaf::lang::ArrayPointer< T > >
```

An override of the less function object so that the Pointer objects can be stored in STL Maps, etc.

6.307.2 Member Function Documentation

6.307.2.1 `template<typename T> bool std::less< decaf::lang::ArrayPointer< T > >::operator() (const decaf::lang::ArrayPointer< T > & left, const decaf::lang::ArrayPointer< T > & right) const [inline]`

References `decaf::lang::ArrayPointer< T, REFCOUNTER >::get()`.

The documentation for this struct was generated from the following file:

- `src/main/decaf/lang/ArrayPointer.h`

6.308 `std::less< decaf::lang::Pointer< T > >` Struct Template - Reference

An override of the less function object so that the Pointer objects can be stored in STL Maps, etc.

```
#include <src/main/decaf/lang/Pointer.h>
```

Public Member Functions

- `bool operator() (const decaf::lang::Pointer< T > &left, const decaf::lang::Pointer< T > &right) const`

6.308.1 Detailed Description

```
template<typename T>struct std::less< decaf::lang::Pointer< T > >
```

An override of the less function object so that the Pointer objects can be stored in STL Maps, etc.

6.308.2 Member Function Documentation

6.308.2.1 `template<typename T> bool std::less< decaf::lang::Pointer< T > >::operator() (const decaf::lang::Pointer< T > & left, const decaf::lang::Pointer< T > & right) const [inline]`

References `decaf::lang::Pointer< T, REFCOUNTER >::get()`.

The documentation for this struct was generated from the following file:

- `src/main/decaf/lang/Pointer.h`

6.309 decaf::util::logging::Level Class Reference

The **Level** (p. 1616) class defines a set of standard logging levels that can be used to control logging output.

```
#include <src/main/decaf/util/logging/Level.h>
```

Inheritance diagram for decaf::util::logging::Level:

Public Member Functions

- virtual **~Level** ()
- int **intValue** () const
- std::string **getName** () const
- std::string **toString** () const
- virtual int **compareTo** (const **Level** &value) const
- virtual bool **equals** (const **Level** &value) const
- virtual bool **operator==** (const **Level** &value) const
- virtual bool **operator<** (const **Level** &value) const

Static Public Member Functions

- static **Level parse** (const std::string &name)
*Parse a level name string into a **Level** (p. 1616).*

Static Public Attributes

- static const **Level INHERIT**
*NULL is a special level that indicates that the **Logger** (p. 1693) should get its **Level** (p. 1616) from its parent **Logger** (p. 1693), the value is initialized as zero.*
- static const **Level OFF**
OFF is a special level that can be used to turn off logging.
- static const **Level SEVERE**
SEVERE is a message level indicating a serious failure.
- static const **Level WARNING**
WARNING is a message level indicating a potential problem.
- static const **Level INFO**
INFO is a message level for informational messages.
- static const **Level DEBUG**
DEBUG is a level for more verbose informative messages.
- static const **Level CONFIG**
CONFIG is a message level for static configuration messages.
- static const **Level FINE**

FINE is a message level providing tracing information.

- static const **Level FINER**

FINER indicates a fairly detailed tracing message.

- static const **Level FINEST**

FINEST indicates a highly detailed tracing message.

- static const **Level ALL**

ALL indicates that all messages should be logged.

Protected Member Functions

- **Level** (const std::string &name, int value)

*Create a named **Level** (p. 1616) with a given integer value.*

6.309.1 Detailed Description

The **Level** (p. 1616) class defines a set of standard logging levels that can be used to control logging output.

The logging **Level** (p. 1616) objects are ordered and are specified by ordered integers. Enabling logging at a given level also enables logging at all higher levels.

Clients should normally use the predefined **Level** (p. 1616) constants such as **Level.SEVERE** (p. 1620).

The levels in descending order are:

* SEVERE (highest value) * WARNING * INFO * DEBUG * CONFIG * FINE * FINER
* FINEST (lowest value)

In addition there is a level OFF that can be used to turn off logging, and a level ALL that can be used to enable logging of all messages.

It is possible for third parties to define additional logging levels by subclassing **Level** (p. 1616). In such cases subclasses should take care to chose unique integer level values.

Since

1.0

6.309.2 Constructor & Destructor Documentation

6.309.2.1 decaf::util::logging::Level::Level (const std::string & name, int value) [protected]

Create a named **Level** (p. 1616) with a given integer value.

Parameters

<i>name</i>	Name of the level, e.g. SEVERE
<i>value</i>	Unique integer value of this level, e.g. 100

6.309.2.2 virtual `decaf::util::logging::Level::~~Level ()` `[virtual]`

6.309.3 Member Function Documentation

6.309.3.1 virtual `int decaf::util::logging::Level::compareTo (const Level & value)`
`const` `[virtual]`

6.309.3.2 virtual `bool decaf::util::logging::Level::equals (const Level & value)` `const`
`[virtual]`

6.309.3.3 `std::string decaf::util::logging::Level::getName ()` `const` `[inline]`

Returns

the name of this **Level** (p. 1616) instance.

6.309.3.4 `int decaf::util::logging::Level::intValue ()` `const` `[inline]`

Returns

the integer value of this level instance.

6.309.3.5 virtual `bool decaf::util::logging::Level::operator< (const Level & value)` `const`
`[virtual]`

6.309.3.6 virtual `bool decaf::util::logging::Level::operator== (const Level & value)` `const`
`[virtual]`

6.309.3.7 `static Level decaf::util::logging::Level::parse (const std::string & name)`
`[static]`

Parse a level name string into a **Level** (p. 1616).

The argument string may consist of either a level name or an integer value.

For example:

* "SEVERE" * "1000"

Parameters

<i>name</i>	- The name or int value of the desired Level (p. 1616)
-------------	---

Returns

the parsed **Level** (p. 1616) value, passing in a level name that is an int value that is not one of the known **Level** (p. 1616) values will result in a new **Level** (p. 1616) that has been initialized with that int value and name as the string form of the int.

Exceptions

<i>IllegalArgument-Exception</i>	if the value is not valid, validity means that the string is either a valid int (between Integer::MIN_VALUE and Integer::MAX_VALUE or is one of the known level names.
----------------------------------	--

6.309.3.8 `std::string decaf::util::logging::Level::toString () const` `[inline]`

Returns

the string value of this **Level** (p. 1616), e.g. "SEVERE".

6.309.4 Field Documentation

6.309.4.1 `const Level decaf::util::logging::Level::ALL` `[static]`

ALL indicates that all messages should be logged.

This level is initialized to Integer::MIN_VALUE.

6.309.4.2 `const Level decaf::util::logging::Level::CONFIG` `[static]`

CONFIG is a message level for static configuration messages.

CONFIG messages are intended to provide a variety of static configuration information, to assist in debugging problems that may be associated with particular configurations. For example, CONFIG message might include the CPU type, the System properties, etc. This level is initialized to 600.

6.309.4.3 `const Level decaf::util::logging::Level::DEBUG` `[static]`

DEBUG is a level for more verbose informative messages.

DEBUG messages are intended to provide a more detailed message intended for use by developers in tracking the behavior of a client. DEBUG messages typically contain more implementation specific information that might not be significant to end users or system admins. This level is initialized to 700.

6.309.4.4 `const Level decaf::util::logging::Level::FINE` `[static]`

FINE is a message level providing tracing information.

All of FINE, FINER, and FINEST are intended for relatively detailed tracing. The exact meaning of the three levels will vary between subsystems, but in general, FINEST should be used for the most detailed output, FINER for somewhat less detailed output, and FINE for the lowest volume (and most important) messages.

In general the FINE level should be used for information that will be broadly interesting to developers who do not have a specialized interest in the specific subsystem.

FINE messages might include things like minor (recoverable) failures. Issues indicating potential performance problems are also worth logging as FINE. This level is initialized to 500.

6.309.4.5 `const Level decaf::util::logging::Level::FINER` [static]

FINER indicates a fairly detailed tracing message.

By default logging calls for entering, returning, or throwing an exception are traced at this level. This level is initialized to 400.

6.309.4.6 `const Level decaf::util::logging::Level::FINEST` [static]

FINEST indicates a highly detailed tracing message.

This level is initialized to 300.

6.309.4.7 `const Level decaf::util::logging::Level::INFO` [static]

INFO is a message level for informational messages.

Typically INFO messages will be written to the console or its equivalent. So the INFO level should only be used for reasonably significant messages that will make sense to end users and system admins. This level is initialized to 800.

6.309.4.8 `const Level decaf::util::logging::Level::INHERIT` [static]

NULL is a special level that indicates that the **Logger** (p. 1693) should get its **Level** (p. 1616) from its parent **Logger** (p. 1693), the value is initialized as zero.

6.309.4.9 `const Level decaf::util::logging::Level::OFF` [static]

OFF is a special level that can be used to turn off logging.

This level is initialized to `Integer::MAX_VALUE`

6.309.4.10 `const Level decaf::util::logging::Level::SEVERE` [static]

SEVERE is a message level indicating a serious failure.

In general SEVERE messages should describe events that are of considerable importance and which will prevent normal program execution. They should be reasonably intelligible to end users and to system administrators. This level is initialized to 1000.

6.309.4.11 `const Level decaf::util::logging::Level::WARNING` [static]

WARNING is a message level indicating a potential problem.

In general WARNING messages should describe events that will be of interest to end users or system managers, or which indicate potential problems. This level is initialized to 900.

The documentation for this class was generated from the following file:

- `src/main/decaf/util/logging/Level.h`

6.310 `decaf::util::concurrent::LinkedBlockingQueue< E >` Class - Template Reference

A **BlockingQueue** (p. 538) derivative that allows for a bound to be placed on the number of elements that can be enqueued at any one time.

```
#include <src/main/decaf/util/concurrent/LinkedBlockingQueue.h>
```

Inheritance diagram for `decaf::util::concurrent::LinkedBlockingQueue< E >`:

Data Structures

- class **ConstLinkedIterator**
- class **LinkedIterator**
- class **QueueNode**
- class **TotalLock**

Public Member Functions

- **LinkedBlockingQueue** ()
Create a new instance with a Capacity of Integer::MAX_VALUE.
- **LinkedBlockingQueue** (int capacity)
Create a new instance with the given initial capacity value.
- **LinkedBlockingQueue** (const **Collection**< E > &collection)
*Create a new instance with a Capacity of Integer::MAX_VALUE and adds all the values contained in the specified collection to this **Queue** (p. 2222).*
- virtual **~LinkedBlockingQueue** ()
- **LinkedBlockingQueue**< E > & **operator=** (const **LinkedBlockingQueue**< E > &queue)
- **LinkedBlockingQueue**< E > & **operator=** (const **Collection**< E > &collection)
- virtual int **size** () const
Returns the number of elements in this collection.
- virtual void **clear** ()

Removes all of the elements from this collection (optional operation).

The collection will be empty after this method returns.

*This implementation iterates over this collection, removing each element using the - **Iterator.remove** (p. 1560) operation. Most implementations will probably choose to override this method for efficiency.*

Note that this implementation will throw an UnsupportedOperationException if the iterator returned by this collection's iterator method does not implement the remove method and this collection is non-empty.

Exceptions

UnsupportedOperation-Exception	if the clear operation is not supported by this collection
--------------------------------	--

This implementation repeatedly invokes poll until it returns false.

- virtual int **remainingCapacity** () const

Returns the number of additional elements that this queue can ideally (in the absence of memory or resource constraints) accept without blocking, or Integer::MAX_VALUE if there is no intrinsic limit.

- virtual void **put** (const E &value)

Inserts the specified element into this queue, waiting if necessary for space to become available.

- virtual bool **offer** (const E &value, long long timeout, const TimeUnit &unit)

Inserts the specified element into this queue, waiting up to the specified wait time if necessary for space to become available.

- virtual bool **offer** (const E &value)

Inserts the specified element into the queue provided that the condition allows such an operation.

- virtual E **take** ()

Retrieves and removes the head of this queue, waiting if necessary until an element becomes available.

- virtual bool **poll** (E &result, long long timeout, const TimeUnit &unit)

Retrieves and removes the head of this queue, waiting up to the specified wait time if necessary for an element to become available.

- virtual bool **poll** (E &result)

Gets and removes the element in the head of the queue.

- virtual bool **peek** (E &result) const

Gets but not removes the element in the head of the queue.

- virtual bool **remove** (const E &value)

Removes a single instance of the specified element from the collection.

More formally, removes an element e such that (value == NULL ? e == NULL : value == e), if this collection contains one or more such elements. Returns true if this collection contained the specified element (or equivalently, if this collection changed as a result of the call).

Parameters

value	The reference to the element to remove from this Collection (p. 851).
-------	--

Returns

true if the collection was changed, false otherwise.

Exceptions

UnsupportedOperation-Exception	<i>if this is an unmodifiable collection.</i>
NullPointerException	<i>if the Collection (p. 851) is a container of pointers and does not allow NULL values.</i>

This implementation iterates over the collection looking for the specified element. - If it finds the element, it removes the element from the collection using the iterator's remove method.

Note that this implementation throws an UnsupportedOperationException if the iterator returned by this collection's iterator method does not implement the remove method and this collection contains the specified object.

- virtual std::vector< E > **toArray** () const

*Answers an STL vector containing copies of all elements contained in this **Collection** (p. 851).*

- virtual std::string **toString** () const
- virtual int **drainTo** (**Collection**< E > &c)

Removes all available elements from this queue and adds them to the given collection.

- virtual int **drainTo** (**Collection**< E > &sink, int maxElements)

Removes at most the given number of available elements from this queue and adds them to the given collection.

- virtual **decaf::util::Iterator** < E > * **iterator** ()
- virtual **decaf::util::Iterator** < E > * **iterator** () const

6.310.1 Detailed Description

```
template<typename E>class decaf::util::concurrent::LinkedBlockingQueue< E >
```

A **BlockingQueue** (p. 538) derivative that allows for a bound to be placed on the number of elements that can be enqueued at any one time.

Elements are inserted and removed in FIFO order. The internal structure of the queue is based on a linked nodes which provides for better performance over their array based versions but the performance is less predictable.

The capacity bound of this class default to Integer::MAX_VALUE.

Since

1.0

6.310.2 Constructor & Destructor Documentation

6.310.2.1 `template<typename E> decaf::util::concurrent::LinkedBlockingQueue< E >::LinkedBlockingQueue () [inline]`

Create a new instance with a Capacity of Integer::MAX_VALUE.

6.310 decaf::util::concurrent::LinkedBlockingQueue< E > Class Template Reference 1629

6.310.2.2 `template<typename E> decaf::util::concurrent::LinkedBlockingQueue< E >::LinkedBlockingQueue (int capacity) [inline]`

Create a new instance with the given initial capacity value.

Parameters

<i>capacity</i>	The initial capacity value to assign to this Queue (p. 2222).
-----------------	--

Exceptions

<i>IllegalArgumentException</i>	if the specified capacity is not greater than zero.
---------------------------------	---

6.310.2.3 `template<typename E> decaf::util::concurrent::LinkedBlockingQueue< E >::LinkedBlockingQueue (const Collection< E > & collection) [inline]`

Create a new instance with a Capacity of Integer::MAX_VALUE and adds all the values contained in the specified collection to this **Queue** (p. 2222).

Parameters

<i>collection</i>	The Collection (p. 851) whose elements are to be copied to this Queue (p. 2222).
-------------------	--

Exceptions

<i>IllegalStateException</i>	if the number of elements in the collection exceeds this Queue (p. 2222)'s capacity.
------------------------------	---

References DECAF_CATCH_RETHROW, DECAF_CATCHALL_THROW, and decaf::lang::Iterable< E >::iterator().

6.310.2.4 `template<typename E> virtual decaf::util::concurrent::LinkedBlockingQueue< E >::~~LinkedBlockingQueue () [inline, virtual]`

6.310.3 Member Function Documentation

6.310.3.1 `template<typename E> virtual void decaf::util::concurrent::LinkedBlockingQueue< E >::clear () [inline, virtual]`

Removes all of the elements from this collection (optional operation).

The collection will be empty after this method returns.

This implementation iterates over this collection, removing each element using the **Iterator.remove** (p. 1560) operation. Most implementations will probably choose to override this method for efficiency.

Note that this implementation will throw an `UnsupportedOperationException` if the iterator returned by this collection's iterator method does not implement the remove method and this collection is non-empty.

Exceptions

<i>Unsupported- OperationException</i>	if the clear operation is not supported by this collection
--	--

This implementation repeatedly invokes poll until it returns false.

This implementation repeatedly invokes poll until it returns false.

Reimplemented from **decaf::util::AbstractQueue< E >** (p. 141).

References `decaf::util::concurrent::atomic::AtomicInteger::getAndSet()`, `decaf::util::AbstractCollection< E >::lock()`, `decaf::util::concurrent::Mutex::notify()`, and `decaf::util::concurrent::atomic::AtomicInteger::set()`.

Referenced by `decaf::util::concurrent::LinkedBlockingQueue< E >::operator=()`.

6.310.3.2 `template<typename E> virtual int decaf::util::concurrent::Linked-
BlockingQueue< E >::drainTo (Collection< E > & c) [inline,
virtual]`

Removes all available elements from this queue and adds them to the given collection.

This operation may be more efficient than repeatedly polling this queue. A failure encountered while attempting to add elements to collection `c` may result in elements being in neither, either or both collections when the associated exception is thrown. Attempts to drain a queue to itself result in `IllegalArgumentException`. Further, the behavior of this operation is undefined if the specified collection is modified while the operation is in progress.

Parameters

<code>c</code>	the collection to transfer elements into
----------------	--

Returns

the number of elements transferred

Exceptions

<i>Unsupported- OperationException</i>	if addition of elements is not supported by the specified collection
<i>IllegalArgumentException</i>	if the specified collection is this queue, or some property of an element of this queue prevents it from being added to the specified collection

6.310 decaf::util::concurrent::LinkedBlockingQueue< E > Class Template Reference 1631

Implements **decaf::util::concurrent::BlockingQueue< E >** (p. 541).

References `decaf::lang::Integer::MAX_VALUE`.

6.310.3.3 `template<typename E> virtual int decaf::util::concurrent::Linked-BlockingQueue< E >::drainTo (Collection< E > & c, int maxElements)`
[inline, virtual]

Removes at most the given number of available elements from this queue and adds them to the given collection.

A failure encountered while attempting to add elements to collection `c` may result in elements being in neither, either or both collections when the associated exception is thrown. Attempts to drain a queue to itself result in `IllegalArgumentException`. Further, the behavior of this operation is undefined if the specified collection is modified while the operation is in progress.

Parameters

<i>c</i>	the collection to transfer elements into
<i>max-Elements</i>	the maximum number of elements to transfer

Returns

the number of elements transferred

Exceptions

<i>Unsupported-OperationException</i>	if addition of elements is not supported by the specified collection
<i>IllegalArgumentException</i>	if the specified collection is this queue, or some property of an element of this queue prevents it from being added to the specified collection

Implements **decaf::util::concurrent::BlockingQueue< E >** (p. 542).

References `decaf::util::Collection< E >::add()`, `decaf::util::concurrent::atomic::AtomicInteger::get()`, `decaf::util::concurrent::atomic::AtomicInteger::getAndAdd()`, and `decaf::lang::Math::min()`.

6.310.3.4 `template<typename E> virtual decaf::util::Iterator<E>*`
`decaf::util::concurrent::LinkedBlockingQueue< E >::iterator ()`
[inline, virtual]

Returns

an iterator over a set of elements of type `T`.

Implements **decaf::lang::Iterable< E >** (p. 1557).

```
6.310.3.5  template<typename E> virtual decaf::util::Iterator<E>*
           decaf::util::concurrent::LinkedBlockingQueue< E >::iterator ( ) const
           [inline, virtual]
```

Implements **decaf::lang::Iterable< E >** (p. 1558).

```
6.310.3.6  template<typename E> virtual bool decaf::util::concurrent::LinkedBlocking-
           Queue< E >::offer ( const E & e, long long timeout, const TimeUnit & unit )
           [inline, virtual]
```

Inserts the specified element into this queue, waiting up to the specified wait time if necessary for space to become available.

Parameters

<i>e</i>	the element to add
<i>timeout</i>	how long to wait before giving up, in units of <i>unit</i>
<i>unit</i>	a TimeUnit (p. 2756) determining how to interpret the <i>timeout</i> parameter

Returns

`true` if successful, or `false` if the specified waiting time elapses before space is available

Exceptions

<i>InterruptedException</i>	if interrupted while waiting
<i>NullPointerException</i>	if the specified element is null
<i>IllegalArgumentException</i>	if some property of the specified element prevents it from being added to this queue

Implements **decaf::util::concurrent::BlockingQueue< E >** (p. 543).

References `decaf::util::concurrent::atomic::AtomicInteger::get()`, `decaf::util::concurrent::atomic::AtomicInteger::getAndIncrement()`, `decaf::util::concurrent::Mutex::notify()`, `decaf::util::concurrent::TimeUnit::toMillis()`, and `decaf::util::concurrent::Mutex::wait()`.

```
6.310.3.7  template<typename E> virtual bool decaf::util::concurrent::Linked-
           BlockingQueue< E >::offer ( const E & value ) [inline,
           virtual]
```

Inserts the specified element into the queue provided that the condition allows such an operation.

The method is generally preferable to the `collection.add(E)`, since the latter might throw an exception if the operation fails.

6.310 decaf::util::concurrent::LinkedBlockingQueue< E > Class Template Reference

1633

Parameters

<i>value</i>	the specified element to insert into the queue.
--------------	---

Returns

true if the operation succeeds and false if it fails.

Exceptions

<i>NullPointerException</i>	if the Queue (p. 2222) implementation does not allow Null values to be inserted into the Queue (p. 2222).
<i>IllegalArgument-Exception</i>	if some property of the specified element prevents it from being added to this queue

Implements **decaf::util::Queue< E >** (p. 2224).

References **decaf::util::concurrent::atomic::AtomicInteger::get()**, **decaf::util::concurrent::atomic::AtomicInteger::getAndIncrement()**, and **decaf::util::concurrent::Mutex::notify()**.

6.310.3.8 `template<typename E> LinkedBlockingQueue<E>&
decaf::util::concurrent::LinkedBlockingQueue< E >::operator= (const
LinkedBlockingQueue< E > & queue) [inline]`

References **decaf::util::AbstractQueue< E >::addAll()**, and **decaf::util::concurrent::LinkedBlockingQueue< E >::clear()**.

6.310.3.9 `template<typename E> LinkedBlockingQueue<E>&
decaf::util::concurrent::LinkedBlockingQueue< E >::operator= (const
Collection< E > & collection) [inline]`

References **decaf::util::AbstractQueue< E >::addAll()**, and **decaf::util::concurrent::LinkedBlockingQueue< E >::clear()**.

6.310.3.10 `template<typename E> virtual bool decaf::util::concurrent::Linked-
BlockingQueue< E >::peek (E & result) const [inline,
virtual]`

Gets but not removes the element in the head of the queue.

The result if successful is assigned to the result parameter.

Parameters

<i>result</i>	Reference to an instance of the contained type to assigned the removed value to.
---------------	--

Returns

true if the element at the head of the queue was removed and assigned to the result parameter.

Implements **decaf::util::Queue< E >** (p. 2225).

References **decaf::util::concurrent::atomic::AtomicInteger::get()**, and **decaf::lang::Pointer< T, REFCOUNTER >::get()**.

6.310.3.11 `template<typename E> virtual bool decaf::util::concurrent::LinkedBlockingQueue< E >::poll (E & result, long long timeout, const TimeUnit & unit) [inline, virtual]`

Retrieves and removes the head of this queue, waiting up to the specified wait time if necessary for an element to become available.

Parameters

<i>result</i>	the referenced value that will be assigned the value retrieved from the Queue (p. 2222). Undefined if this methods returned false.
<i>timeout</i>	how long to wait before giving up, in units of <i>unit</i>
<i>unit</i>	a TimeUnit (p. 2756) determining how to interpret the <i>timeout</i> parameter.

Returns

true if successful or false if the specified waiting time elapses before an element is available.

Exceptions

<i>InterruptedException</i>	if interrupted while waiting
-----------------------------	------------------------------

Implements **decaf::util::concurrent::BlockingQueue< E >** (p. 543).

References **decaf::util::concurrent::atomic::AtomicInteger::get()**, **decaf::util::concurrent::atomic::AtomicInteger::getAndDecrement()**, **decaf::util::concurrent::Mutex::notify()**, **decaf::util::concurrent::TimeUnit::toMillis()**, and **decaf::util::concurrent::Mutex::wait()**.

6.310.3.12 `template<typename E> virtual bool decaf::util::concurrent::LinkedBlockingQueue< E >::poll (E & result) [inline, virtual]`

Gets and removes the element in the head of the queue.

If the operation succeeds the value of the element at the head of the **Queue** (p. 2222) is assigned to the result parameter and the method returns true. If the operation fails the method returns false and the value of the result parameter is undefined.

6.310 decaf::util::concurrent::LinkedBlockingQueue< E > Class Template Reference 1635

Parameters

<i>result</i>	Reference to an instance of the contained type to assigned the removed value to.
---------------	--

Returns

true if the element at the head of the queue was removed and assigned to the result parameter.

Implements **decaf::util::Queue< E >** (p. 2225).

References **decaf::util::concurrent::atomic::AtomicInteger::get()**, **decaf::util::concurrent::atomic::AtomicInteger::getAndDecrement()**, and **decaf::util::concurrent::Mutex::notify()**.

6.310.3.13 `template<typename E> virtual void decaf::util::concurrent::Linked-BlockingQueue< E >::put (const E & value) [inline, virtual]`

Inserts the specified element into this queue, waiting if necessary for space to become available.

Parameters

<i>value</i>	the element to add
--------------	--------------------

Exceptions

<i>InterruptedException</i>	if interrupted while waiting
<i>NullPointerException</i>	if the specified element is null
<i>IllegalArgumentException</i>	if some property of the specified element prevents it from being added to this queue

Implements **decaf::util::concurrent::BlockingQueue< E >** (p. 544).

References **decaf::util::concurrent::atomic::AtomicInteger::get()**, **decaf::util::concurrent::atomic::AtomicInteger::getAndIncrement()**, **decaf::util::concurrent::Mutex::notify()**, and **decaf::util::concurrent::Mutex::wait()**.

6.310.3.14 `template<typename E> virtual int decaf::util::concurrent::Linked-BlockingQueue< E >::remainingCapacity () const [inline, virtual]`

Returns the number of additional elements that this queue can ideally (in the absence of memory or resource constraints) accept without blocking, or `Integer::MAX_VALUE` if there is no intrinsic limit.

Note that you *cannot* always tell if an attempt to insert an element will succeed by inspecting `remainingCapacity` because it may be the case that another thread is

about to insert or remove an element.

Returns

the remaining capacity

Implements **decaf::util::concurrent::BlockingQueue< E >** (p. 544).

References **decaf::util::concurrent::atomic::AtomicInteger::get()**.

6.310.3.15 `template<typename E> virtual bool decaf::util::concurrent::Linked-BlockingQueue< E >::remove (const E & value) [inline, virtual]`

Removes a single instance of the specified element from the collection.

More formally, removes an element *e* such that (*value* == NULL ? *e* == NULL : *value* == *e*), if this collection contains one or more such elements. Returns true if this collection contained the specified element (or equivalently, if this collection changed as a result of the call).

Parameters

<i>value</i>	The reference to the element to remove from this Collection (p. 851).
--------------	--

Returns

true if the collection was changed, false otherwise.

Exceptions

<i>UnsupportedOperationException</i>	if this is an unmodifiable collection.
<i>NullPointerException</i>	if the Collection (p. 851) is a container of pointers and does not allow NULL values.

This implementation iterates over the collection looking for the specified element. If it finds the element, it removes the element from the collection using the iterator's remove method.

Note that this implementation throws an `UnsupportedOperationException` if the iterator returned by this collection's iterator method does not implement the remove method and this collection contains the specified object.

This implementation iterates over the collection looking for the specified element. If it finds the element, it removes the element from the collection using the iterator's remove method.

Note that this implementation throws an `UnsupportedOperationException` if the iterator returned by this collection's iterator method does not implement the remove method and this collection contains the specified object.

6.310 decaf::util::concurrent::LinkedBlockingQueue< E > Class Template Reference 1637

Reimplemented from **decaf::util::AbstractCollection< E >** (p. 116).

References decaf::util::AbstractCollection< E >::lock().

6.310.3.16 `template<typename E> virtual int decaf::util::concurrent::-
LinkedBlockingQueue< E >::size () const [inline,
virtual]`

Returns the number of elements in this collection.

If this collection contains more than Integer::MAX_VALUE elements, returns Integer::MAX_VALUE.

Returns

the number of elements in this collection

Implements **decaf::util::Collection< E >** (p. 864).

References decaf::util::concurrent::atomic::AtomicInteger::get().

Referenced by decaf::util::concurrent::LinkedBlockingQueue< E >::toArray().

6.310.3.17 `template<typename E> virtual E decaf::util::concurrent-
::LinkedBlockingQueue< E >::take () [inline,
virtual]`

Retrieves and removes the head of this queue, waiting if necessary until an element becomes available.

Returns

the head of this queue

Exceptions

<i>InterruptedException</i>	if interrupted while waiting
-----------------------------	------------------------------

Implements **decaf::util::concurrent::BlockingQueue< E >** (p. 545).

References decaf::util::concurrent::atomic::AtomicInteger::get(), decaf::util::concurrent::atomic::AtomicInteger::getAndDecrement(), decaf::util::concurrent::Mutex::notify(), and decaf::util::concurrent::Mutex::wait().

6.310.3.18 `template<typename E> virtual std::vector<E> decaf::util::concurrent-
::LinkedBlockingQueue< E >::toArray () const [inline,
virtual]`

Answers an STL vector containing copies of all elements contained in this **Collection** (p. 851).

All the elements in the array will not be referenced by the collection. The elements in the returned array will be sorted to the same order as those returned by the iterator of this collection itself if the collection guarantees the order.

Returns

an vector of copies of all the elements from this **Collection** (p. 851)

Reimplemented from **decaf::util::AbstractCollection**< **E** > (p. 119).

References `decaf::util::concurrent::atomic::AtomicInteger::get()`, `decaf::util::AbstractCollection`< **E** >::`lock()`, and `decaf::util::concurrent::LinkedBlockingQueue`< **E** >::`size()`.

```
6.310.3.19  template<typename E> virtual std::string decaf::util::concurrent::-
             LinkedBlockingQueue< E >::toString ( ) const [inline,
             virtual]
```

References `decaf::util::concurrent::atomic::AtomicInteger::get()`.

The documentation for this class was generated from the following file:

- `src/main/decaf/util/concurrent/LinkedBlockingQueue.h`

6.311 **decaf::util::LinkedList**< **E** > Class Template Reference

A complete implementation of the **List** (p. 1658) interface using a doubly linked list data structure.

```
#include <src/main/decaf/util/LinkedList.h>
```

Inheritance diagram for `decaf::util::LinkedList`< **E** >:

Data Structures

- class **ConstLinkedListIterator**
- class **ConstReverselIterator**
- class **LinkedListIterator**
- class **ListNode**
- class **ReverselIterator**

Public Member Functions

- **LinkedList** ()
- **LinkedList** (const **LinkedList**< **E** > &list)

- **LinkedList** (const **Collection**< E > &collection)
- virtual ~**LinkedList** ()
- **LinkedList**< E > & **operator=** (const **LinkedList**< E > &list)
- **LinkedList**< E > & **operator=** (const **Collection**< E > &collection)
- virtual E **get** (int index) const

Gets the element contained at position passed.

Parameters

index	<i>The position to get.</i>
-------	-----------------------------

Returns

value at index specified.

Exceptions

IndexOutOfBoundsException	<i>if the index given is less than zero or greater than the List (p. 1658) size.</i>
---------------------------	---

*This implementation first gets a list iterator pointing to the indexed element (with list-iterator(index)). Then, it gets the element using **ListIterator.next** (p. 1560) and returns it.*

- virtual E **set** (int index, const E &element)

Replaces the element at the specified position in this list with the specified element.

Parameters

index	<i>The index of the element to replace.</i>
element	<i>The element to be stored at the specified position.</i>

Returns

the element previously at the specified position.

Exceptions

IndexOutOfBoundsException	<i>if the index given is less than zero or greater than the List (p. 1658) size.</i>
UnsupportedOperationException	<i>if this is an unmodifiable collection.</i>
NullPointerException	<i>if the Collection (p. 851) is a container of pointers and does not allow NULL values.</i>
IllegalArgumentException	<i>if some property of the element prevents it from being added to this collection</i>
IllegalStateException	<i>if the element cannot be added at this time due to insertion restrictions.</i>

*This implementation first gets a list iterator pointing to the indexed element (with list-iterator(index)). Then, it gets the current element using **ListIterator.next** (p. 1560) and replaces it with **ListIterator.set** (p. 1674).*

- virtual bool **add** (const E &value)

Returns true if this collection changed as a result of the call.

- virtual void **add** (int index, const E &value)

Inserts the specified element at the specified position in this list.

Shifts the element currently at that position (if any) and any subsequent elements to the right (adds one to their indices).

Parameters

index	<i>The index at which the specified element is to be inserted.</i>
element	<i>The element to be inserted in this List (p. 1658).</i>

Exceptions

IndexOutOfBoundsException	<i>if the index is greater than size of the List (p. 1658).</i>
UnsupportedOperationException	<i>if this is an unmodifiable collection.</i>
NullPointerException	<i>if the Collection (p. 851) is a container of pointers and does not allow NULL values.</i>
IllegalArgumentException	<i>if some property of the element prevents it from being added to this collection</i>
IllegalStateException	<i>if the element cannot be added at this time due to insertion restrictions.</i>

This implementation first gets a list iterator pointing to the indexed element (with `listIterator(index)`). Then, it inserts the specified element with `ListIterator.add` (p. 1672).

- virtual bool **addAll** (const **Collection**< E > &collection)

Adds all of the elements in the specified collection to this collection.

The behavior of this operation is undefined if the specified collection is modified while the operation is in progress. (This implies that the behavior of this call is undefined if the specified collection is this collection, and this collection is nonempty.)

Parameters

collection	<i>The Collection (p. 851) whose elements are added to this one.</i>
------------	---

Returns

true if this collection changed as a result of the call

Exceptions

UnsupportedOperationException	<i>if this is an unmodifiable collection.</i>
NullPointerException	<i>if the Collection (p. 851) is a container of pointers and does not allow NULL values.</i>
IllegalArgumentException	<i>if some property of an element prevents it from being added to this collection</i>
IllegalStateException	<i>if an element cannot be added at this time due to insertion restrictions.</i>

This implementation iterates over the specified collection, and adds each object returned by the iterator to this collection, in turn.

Note that this implementation will throw an `UnsupportedOperationException` unless `add` is overridden (assuming the specified collection is non-empty).

- virtual bool **addAll** (int index, const **Collection**< E > &collection)

Inserts all of the elements in the specified collection into this list at the specified position (optional operation).

Shifts the element currently at that position (if any) and any subsequent elements to the right (increases their indices). The new elements will appear in this list in the order that they are returned by the specified collection's iterator. The behavior of this operation is undefined if the specified collection is modified while the operation is in progress. (Note that this will occur if the specified collection is this list, and it's nonempty.)

Parameters

index	<i>The index at which to insert the first element from the specified collection</i>
source	<i>The Collection (p. 851) containing elements to be added to this list</i>

Returns

true if this list changed as a result of the call

Exceptions

IndexOutOfBoundsException	<i>if the index given is less than zero or greater than the List (p. 1658) size.</i>
UnsupportedOperationException	<i>if this is an unmodifiable collection.</i>
NullPointerException	<i>if the Collection (p. 851) is a container of pointers and does not allow NULL values.</i>
IllegalArgumentException	<i>if some property of the element prevents it from being added to this collection</i>
IllegalStateException	<i>if the element cannot be added at this time due to insertion restrictions.</i>

*This implementation gets an iterator over the specified collection and a list iterator over this list pointing to the indexed element (with listIterator(index)). Then, it iterates over the specified collection, inserting the elements obtained from the iterator into this list, one at a time, using **ListIterator.add** (p. 1672) (to skip over the added element).*

- virtual void **copy** (const **Collection**< E > &collection)

*Renders this **Collection** (p. 851) as a Copy of the given **Collection** (p. 851).*

- virtual bool **remove** (const E &value)

*Removes a single instance of the specified element from the collection.
More formally, removes an element e such that (value == NULL ? e == NULL : value == e), if this collection contains one or more such elements. Returns true if this collection contained the specified element (or equivalently, if this collection changed as a result of the call).*

Parameters

value	<i>The reference to the element to remove from this Collection (p. 851).</i>
-------	---

Returns

true if the collection was changed, false otherwise.

Exceptions

UnsupportedOperationException	<i>if this is an unmodifiable collection.</i>
NullPointerException	<i>if the Collection (p. 851) is a container of pointers and does not allow NULL values.</i>

This implementation iterates over the collection looking for the specified element. - If it finds the element, it removes the element from the collection using the iterator's remove method.

Note that this implementation throws an UnsupportedOperationException if the iterator returned by this collection's iterator method does not implement the remove method and this collection contains the specified object.

- virtual bool **isEmpty** () const

Returns true if this collection contains no elements.

- virtual int **size** () const

Returns the number of elements in this collection.

- virtual void **clear** ()

Removes all of the elements from this collection (optional operation).

- virtual bool **contains** (const E &value) const

Returns true if this collection contains the specified element.

More formally, returns true if and only if this collection contains at least one element e such that (value == NULL ? e == NULL : value == e).

Parameters

value	<i>The value to check for presence in the collection.</i>
-------	---

Returns

true if there is at least one of the elements in the collection

Exceptions

NullPointerException	<i>if the Collection (p. 851) contains pointers and the Collection (p. 851) does not allow for NULL elements (optional check).</i>
----------------------	--

This implementation iterates over the elements in the collection, checking each element in turn for equality with the specified element.

- virtual int **indexOf** (const E &value) const

Returns the index of the first occurrence of the specified element in this list, or -1 if this list does not contain the element.

- virtual int **lastIndexOf** (const E &value) const

Returns the index of the last occurrence of the specified element in this list, or -1 if this list does not contain the element.

- virtual std::vector< E > **toArray** () const

*Answers an STL vector containing copies of all elements contained in this **Collection** (p. 851).*

- virtual bool **offer** (const E &value)

Inserts the specified element into the queue provided that the condition allows such an operation.

- virtual bool **poll** (E &result)

Gets and removes the element in the head of the queue.

- virtual E **remove** ()

Gets and removes the element in the head of the queue.

- virtual bool **peek** (E &result) const

Gets but not removes the element in the head of the queue.

- virtual E **element** () const

Gets but not removes the element in the head of the queue.

- virtual void **addFirst** (const E &value)

*Inserts an element onto the front of the **Deque** (p. 1196) if possible without violating the implementations capacity restrictions.*

- virtual void **addLast** (const E &value)

*Inserts an element onto the end of the **Deque** (p. 1196) if possible without violating the implementations capacity restrictions.*

- virtual E & **getFirst** ()

*Attempts to fetch a reference to the first element in the **Deque** (p. 1196).*

- virtual const E & **getFirst** () const
- virtual E & **getLast** ()
 - Attempts to fetch a reference to the last element in the **Deque** (p. 1196).*
- virtual const E & **getLast** () const
- virtual bool **offerFirst** (const E &element)
 - This method attempts to insert the given element into the **Deque** (p. 1196) at the front end.*
- virtual bool **offerLast** (const E &element)
 - This method attempts to insert the given element into the **Deque** (p. 1196) at the end.*
- virtual E **removeFirst** ()
 - Removes the topmost element from the **Deque** (p. 1196) and returns it.*
- virtual E **removeLast** ()
 - Removes the last element from the **Deque** (p. 1196) and returns it.*
- virtual bool **pollFirst** (E &result)
 - Removes the first element from the **Deque** (p. 1196) assigns it to the element reference passed.*
- virtual bool **pollLast** (E &result)
 - Removes the last element from the **Deque** (p. 1196) assigns it to the element reference passed.*
- virtual bool **peekFirst** (E &result) const
 - Retrieves the first element contained in this **Deque** (p. 1196) and assigns its value to the reference value passed the value however is not removed from the **Deque** (p. 1196).*
- virtual bool **peekLast** (E &result) const
 - Retrieves the last element contained in this **Deque** (p. 1196) and assigns its value to the reference value passed the value however is not removed from the **Deque** (p. 1196).*
- virtual E **pop** ()
 - Treats this **Deque** (p. 1196) as a stack and attempts to pop an element off the top.*
- virtual void **push** (const E &element)
 - Pushes an element onto the stack represented by this deque (in other words, at the head of this deque) if it is possible to do so immediately without violating capacity restrictions, otherwise it throwing an **IllegalStateException** if no space is currently available.*
- virtual bool **removeFirstOccurrence** (const E &value)
 - Removes the first occurrence of the specified element from this **Deque** (p. 1196).*
- virtual bool **removeLastOccurrence** (const E &value)
 - Removes the last occurrence of the specified element from this **Deque** (p. 1196).*
- virtual **ListIterator**< E > * **listIterator** (int index)
- virtual **ListIterator**< E > * **listIterator** (int index) const
- virtual **Iterator**< E > * **descendingIterator** ()
 - Provides an **Iterator** (p. 1559) over this **Collection** (p. 851) that traverses the element in reverse order.*
- virtual **Iterator**< E > * **descendingIterator** () const

6.311.1 Detailed Description

```
template<typename E>class decaf::util::LinkedList< E >
```

A complete implementation of the **List** (p. 1658) interface using a doubly linked list data structure.

This class also implements the **Deque** (p. 1196) interface providing a common interface for additions into the list at the front and end as well as allowing insertions anywhere in between. This class can be used then to implement other data structures such as Stacks, **Queue** (p. 2222)'s or double ended **Queue** (p. 2222)'s.

The operations on this **List** (p. 1658) object that index a particular element involve iterating over the links of the list from beginning to end, starting from whichever end is closer to the location the operation is to be performed on.

Since

1.0

6.311.2 Constructor & Destructor Documentation

```
6.311.2.1 template<typename E> decaf::util::LinkedList< E >::LinkedList ( )
[inline]
```

```
6.311.2.2 template<typename E> decaf::util::LinkedList< E >::LinkedList ( const
LinkedList< E > & list ) [inline]
```

```
6.311.2.3 template<typename E> decaf::util::LinkedList< E >::LinkedList ( const
Collection< E > & collection ) [inline]
```

```
6.311.2.4 template<typename E> virtual decaf::util::LinkedList< E >::~~LinkedList ( )
[inline, virtual]
```

6.311.3 Member Function Documentation

```
6.311.3.1 template<typename E> virtual bool decaf::util::LinkedList< E >::add ( const E
& value ) [inline, virtual]
```

Returns true if this collection changed as a result of the call.

(Returns false if this collection does not permit duplicates and already contains the specified element.)

Collections that support this operation may place limitations on what elements may be added to this collection. In particular, some collections will refuse to add null elements, and others will impose restrictions on the type of elements that may be added. **Collection** (p. 851) classes should clearly specify in their documentation any restrictions on what elements may be added.

If a collection refuses to add a particular element for any reason other than that it already

contains the element, it must throw an exception (rather than returning false). This preserves the invariant that a collection always contains the specified element after this call returns.

For non-pointer values, i.e. class instances or string's the object will be copied into the collection, thus the object must support being copied, must not hide the copy constructor and assignment operator.

Parameters

<i>value</i>	The reference to the element to add to this Collection (p. 851).
--------------	---

Returns

true if the element was added to this **Collection** (p. 851).

Exceptions

<i>UnsupportedOperationException</i>	if this is an unmodifiable collection.
<i>NullPointerException</i>	if the Collection (p. 851) is a container of pointers and does not allow NULL values.
<i>IllegalArgumentException</i>	if some property of the element prevents it from being added to this collection
<i>IllegalStateException</i>	if the element cannot be added at this time due to insertion restrictions.

Reimplemented from **decaf::util::AbstractList< E >** (p. 125).

6.311.3.2 `template<typename E> virtual void decaf::util::LinkedList< E >::add (int index , const E & element) [inline, virtual]`

Inserts the specified element at the specified position in this list.

Shifts the element currently at that position (if any) and any subsequent elements to the right (adds one to their indices).

Parameters

<i>index</i>	The index at which the specified element is to be inserted.
<i>element</i>	The element to be inserted in this List (p. 1658).

Exceptions

<i>IndexOutOfBoundsException</i>	if the index is greater than size of the List (p. 1658).
<i>UnsupportedOperationException</i>	if this is an unmodifiable collection.
<i>NullPointerException</i>	if the Collection (p. 851) is a container of pointers and does not allow NULL values.

<i>IllegalArgument-Exception</i>	if some property of the element prevents it from being added to this collection
<i>IllegalStateException</i>	if the element cannot be added at this time due to insertion restrictions.

This implementation first gets a list iterator pointing to the indexed element (with `listIterator(index)`). Then, it inserts the specified element with `ListIterator.add` (p. 1672).

This implementation first gets a list iterator pointing to the indexed element (with `listIterator(index)`). Then, it inserts the specified element with `ListIterator.add` (p. 1672).

Reimplemented from `decaf::util::AbstractSequentialList< E >` (p. 146).

6.311.3.3 `template<typename E> virtual bool decaf::util::LinkedList< E >::addAll (const Collection< E > & collection) [inline, virtual]`

Adds all of the elements in the specified collection to this collection.

The behavior of this operation is undefined if the specified collection is modified while the operation is in progress. (This implies that the behavior of this call is undefined if the specified collection is this collection, and this collection is nonempty.)

Parameters

<i>collection</i>	The Collection (p. 851) whose elements are added to this one.
-------------------	--

Returns

true if this collection changed as a result of the call

Exceptions

<i>UnsupportedOperationException</i>	if this is an unmodifiable collection.
<i>NullPointerException</i>	if the Collection (p. 851) is a container of pointers and does not allow NULL values.
<i>IllegalArgument-Exception</i>	if some property of an element prevents it from being added to this collection
<i>IllegalStateException</i>	if an element cannot be added at this time due to insertion restrictions.

This implementation iterates over the specified collection, and adds each object returned by the iterator to this collection, in turn.

Note that this implementation will throw an `UnsupportedOperationException` unless `add` is overridden (assuming the specified collection is non-empty).

This implementation iterates over the specified collection, and adds each object returned by the iterator to this collection, in turn.

Note that this implementation will throw an `UnsupportedOperationException` unless `add` is overridden (assuming the specified collection is non-empty).

Reimplemented from **decaf::util::AbstractCollection< E >** (p. 110).

6.311.3.4 `template<typename E> virtual bool decaf::util::LinkedList< E >::addAll (int index, const Collection< E > & source) [inline, virtual]`

Inserts all of the elements in the specified collection into this list at the specified position (optional operation).

Shifts the element currently at that position (if any) and any subsequent elements to the right (increases their indices). The new elements will appear in this list in the order that they are returned by the specified collection's iterator. The behavior of this operation is undefined if the specified collection is modified while the operation is in progress. (Note that this will occur if the specified collection is this list, and it's nonempty.)

Parameters

<i>index</i>	The index at which to insert the first element from the specified collection
<i>source</i>	The Collection (p. 851) containing elements to be added to this list

Returns

true if this list changed as a result of the call

Exceptions

<i>IndexOutOfBoundsException</i>	if the index given is less than zero or greater than the List (p. 1658) size.
<i>UnsupportedOperationException</i>	if this is an unmodifiable collection.
<i>NullPointerException</i>	if the Collection (p. 851) is a container of pointers and does not allow NULL values.
<i>IllegalArgumentException</i>	if some property of the element prevents it from being added to this collection
<i>IllegalStateException</i>	if the element cannot be added at this time due to insertion restrictions.

This implementation gets an iterator over the specified collection and a list iterator over this list pointing to the indexed element (with `listIterator(index)`). Then, it iterates over the specified collection, inserting the elements obtained from the iterator into this list, one at a time, using **ListIterator.add** (p. 1672) (to skip over the added element).

This implementation gets an iterator over the specified collection and a list iterator over this list pointing to the indexed element (with `listIterator(index)`). Then, it iterates over the specified collection, inserting the elements obtained from the iterator into this list, one at a time, using **ListIterator.add** (p. 1672) (to skip over the added element).

Reimplemented from **decaf::util::AbstractSequentialList**< **E** > (p. 147).

6.311.3.5 `template<typename E> virtual void decaf::util::LinkedList< E >::addFirst (const E & element) [inline, virtual]`

Inserts an element onto the front of the **Deque** (p. 1196) if possible without violating the implementations capacity restrictions.

For a capacity restricted **Deque** (p. 1196) it is preferable to call `offerFirst` instead.

Parameters

<i>element</i>	The element to be placed at the front of the Deque (p. 1196).
----------------	--

Exceptions

<i>IllegalStateException</i>	if the element cannot be added at this time due to capacity restrictions
<i>NullPointerException</i>	if the specified element is NULL and this deque is a Collection (p. 851) of pointers and does not permit null elements.
<i>IllegalArgumentException</i>	if some property of the specified element prevents it from being added to this deque.

Implements **decaf::util::Deque**< **E** > (p. 1198).

6.311.3.6 `template<typename E> virtual void decaf::util::LinkedList< E >::addLast (const E & element) [inline, virtual]`

Inserts an element onto the end of the **Deque** (p. 1196) if possible without violating the implementations capacity restrictions.

For a capacity restricted **Deque** (p. 1196) it is preferable to call `offerLast` instead.

Parameters

<i>element</i>	The element to be placed at the end of the Deque (p. 1196).
----------------	--

Exceptions

<i>IllegalStateException</i>	if the element cannot be added at this time due to capacity restrictions
<i>NullPointerException</i>	if the specified element is NULL and this deque is a Collection (p. 851) of pointers and does not permit null elements.
<i>IllegalArgumentException</i>	if some property of the specified element prevents it from being added to this deque.

Implements **decaf::util::Deque**< **E** > (p. 1199).

Referenced by `decaf::util::LinkedList< cms::Connection * >::offer()`.

6.311.3.7 `template<typename E> virtual void decaf::util::LinkedList< E >::clear ()`
`[inline, virtual]`

Removes all of the elements from this collection (optional operation).

The collection will be empty after this method returns.

This implementation iterates over this collection, removing each element using the - **Iterator.remove** (p. 1560) operation. Most implementations will probably choose to override this method for efficiency.

Note that this implementation will throw an `UnsupportedOperationException` if the iterator returned by this collection's iterator method does not implement the remove method and this collection is non-empty.

Exceptions

<i>Unsupported- OperationException</i>	if the clear operation is not supported by this collection
--	--

Reimplemented from **decaf::util::AbstractList< E >** (p. 127).

Referenced by `decaf::util::LinkedList< cms::Connection * >::copy()`, and `decaf::util::LinkedList< cms::Connection * >::operator=()`.

6.311.3.8 `template<typename E> virtual bool decaf::util::LinkedList< E >::contains (`
`const E & value) const [inline, virtual]`

Returns true if this collection contains the specified element.

More formally, returns true if and only if this collection contains at least one element *e* such that `(value == NULL ? e == NULL : value == e)`.

Parameters

<i>value</i>	The value to check for presence in the collection.
--------------	--

Returns

true if there is at least one of the elements in the collection

Exceptions

<i>NullPointerException</i>	if the Collection (p. 851) contains pointers and the Collection (p. 851) does not allow for NULL elements (optional check).
-----------------------------	---

This implementation iterates over the elements in the collection, checking each element in turn for equality with the specified element.

This implementation iterates over the elements in the collection, checking each element in turn for equality with the specified element.

Reimplemented from **decaf::util::AbstractCollection**< **E** > (p. 112).

6.311.3.9 `template<typename E> virtual void decaf::util::LinkedList< E >::copy (const Collection< E > & collection) [inline, virtual]`

Renders this **Collection** (p. 851) as a Copy of the given **Collection** (p. 851).

The default implementation iterates over the contents of the given collection adding each to this collection after first calling this **Collection** (p. 851)'s clear method.

Parameters

<i>collection</i>	The collection to mirror.
-------------------	---------------------------

Exceptions

<i>UnsupportedOperationException</i>	if this is an unmodifiable collection.
<i>IllegalStateException</i>	if the elements cannot be added at this time due to insertion restrictions.

Reimplemented from **decaf::util::AbstractCollection**< **E** > (p. 113).

6.311.3.10 `template<typename E> virtual Iterator<E>* decaf::util::LinkedList< E >::descendingIterator () [inline, virtual]`

Provides an **Iterator** (p. 1559) over this **Collection** (p. 851) that traverses the element in reverse order.

Returns

a new **Iterator** (p. 1559) instance that moves from last to first.

Implements **decaf::util::Deque**< **E** > (p. 1199).

Referenced by `decaf::util::LinkedList< cms::Connection * >::removeLastOccurrence()`.

6.311.3.11 `template<typename E> virtual Iterator<E>* decaf::util::LinkedList< E >::descendingIterator () const [inline, virtual]`

Implements **decaf::util::Deque**< **E** > (p. 1200).

6.311.3.12 `template<typename E> virtual E decaf::util::LinkedList< E >::element () const [inline, virtual]`

Gets but not removes the element in the head of the queue.

Throws a **NoSuchElementException** (p. 1984) if there is no element in the queue.

Returns

the element in the head of the queue.

Exceptions

<i>NoSuchElementException</i> (p. 1984)	if there is no element in the queue.
---	--------------------------------------

Implements **decaf::util::Queue< E >** (p. 2223).

Referenced by **decaf::util::LinkedList< cms::Connection * >::set()**.

6.311.3.13 `template<typename E> virtual E decaf::util::LinkedList< E >::get (int index) const [inline, virtual]`

Gets the element contained at position passed.

Parameters

<i>index</i>	The position to get.
--------------	----------------------

Returns

value at index specified.

Exceptions

<i>IndexOutOfBoundsException</i>	if the index given is less than zero or greater than the List (p. 1658) size.
---	--

This implementation first gets a list iterator pointing to the indexed element (with list-iterator(index)). Then, it gets the element using **ListIterator.next** (p. 1560) and returns it.

This implementation first gets a list iterator pointing to the indexed element (with list-iterator(index)). Then, it gets the element using **ListIterator.next** (p. 1560) and returns it.

Reimplemented from **decaf::util::AbstractSequentialList< E >** (p. 148).

6.311.3.14 `template<typename E> virtual E& decaf::util::LinkedList< E >::getFirst () [inline, virtual]`

Attempts to fetch a reference to the first element in the **Deque** (p. 1196).

This method does not remove the element from the **Deque** (p. 1196) but simply returns a reference to it.

Returns

reference to the first element in the **Deque** (p. 1196).

Exceptions

<i>NoSuchElementException</i> (p. 1984)	if the Deque (p. 1196) is empty.
---	---

Implements **decaf::util::Deque< E >** (p. 1200).

6.311.3.15 `template<typename E> virtual const E& decaf::util::LinkedList< E >::getFirst () const [inline, virtual]`

Implements **decaf::util::Deque< E >** (p. 1201).

6.311.3.16 `template<typename E> virtual E& decaf::util::LinkedList< E >::getLast () [inline, virtual]`

Attempts to fetch a reference to the last element in the **Deque** (p. 1196).

This method does not remove the element from the **Deque** (p. 1196) but simply returns a reference to it.

Returns

reference to the last element in the **Deque** (p. 1196).

Exceptions

<i>NoSuchElementException</i> (p. 1984)	if the Deque (p. 1196) is empty.
---	---

Implements **decaf::util::Deque< E >** (p. 1201).

6.311.3.17 `template<typename E> virtual const E& decaf::util::LinkedList< E >::getLast () const [inline, virtual]`

Implements **decaf::util::Deque< E >** (p. 1202).

6.311.3.18 `template<typename E> virtual int decaf::util::LinkedList< E >::indexOf (const E & value) const [inline, virtual]`

Returns the index of the first occurrence of the specified element in this list, or -1 if this list does not contain the element.

More formally, returns the lowest index *i* such that `get(i) == value`, or -1 if there is no such index.

Parameters

<i>value</i>	The element to search for in this List (p. 1658).
--------------	--

Returns

the index of the first occurrence of the specified element in this list,

Exceptions

<i>NullPointerException</i>	if the Collection (p. 851) is a container of pointers and does not allow NULL values.
-----------------------------	--

Reimplemented from **decaf::util::AbstractList< E >** (p. 128).

Referenced by `decaf::util::LinkedList< cms::Connection * >::contains()`.

6.311.3.19 `template<typename E> virtual bool decaf::util::LinkedList< E >::isEmpty () const [inline, virtual]`

Returns true if this collection contains no elements.

This implementation returns `size() (p. 1658) == 0`.

Returns

true if the size method return 0.

Reimplemented from **decaf::util::AbstractCollection< E >** (p. 114).

6.311.3.20 `template<typename E> virtual int decaf::util::LinkedList< E >::lastIndexOf (const E & value) const [inline, virtual]`

Returns the index of the last occurrence of the specified element in this list, or -1 if this list does not contain the element.

More formally, returns the highest index *i* such that `get(i) == value` or -1 if there is no such index.

Parameters

<i>value</i>	The element to search for in this List (p. 1658).
--------------	--

Returns

the index of the last occurrence of the specified element in this list.

Exceptions

<i>NullPointerException</i>	if the Collection (p. 851) is a container of pointers and does not allow NULL values.
-----------------------------	--

Reimplemented from **decaf::util::AbstractList< E >** (p. 129).

6.311.3.21 `template<typename E> virtual ListIterator<E>* decaf::util::LinkedList< E >::listIterator (int index) [inline, virtual]`

Parameters

<i>index</i>	index of first element to be returned from the list iterator (by a call to the next method).
--------------	--

Returns

a list iterator of the elements in this list (in proper sequence), starting at the specified position in this list. The specified index indicates the first element that would be returned by an initial call to next. An initial call to previous would return the element with the specified index minus one.

Exceptions

<i>IndexOutOfBoundsException</i>	if the index is out of range (<code>index < 0 index > size()</code> (p. 1658))
----------------------------------	--

Reimplemented from **decaf::util::AbstractSequentialList< E >** (p. 149).

6.311.3.22 `template<typename E> virtual ListIterator<E>* decaf::util::LinkedList< E >::listIterator (int index) const [inline, virtual]`

Reimplemented from **decaf::util::AbstractSequentialList< E >** (p. 150).

6.311.3.23 `template<typename E> virtual bool decaf::util::LinkedList< E >::offer (const E & value) [inline, virtual]`

Inserts the specified element into the queue provided that the condition allows such an operation.

The method is generally preferable to the `collection.add(E)`, since the latter might throw an exception if the operation fails.

Parameters

<i>value</i>	the specified element to insert into the queue.
--------------	---

Returns

true if the operation succeeds and false if it fails.

Exceptions

<i>NullPointerException</i>	if the Queue (p. 2222) implementation does not allow Null values to be inserted into the Queue (p. 2222).
<i>IllegalArgumentException</i>	if some property of the specified element prevents it from being added to this queue

Implements **decaf::util::Queue< E >** (p. 2224).

6.311.3.24 `template<typename E> virtual bool decaf::util::LinkedList< E >::offerFirst (const E & element) [inline, virtual]`

This method attempts to insert the given element into the **Deque** (p. 1196) at the front end.

Unlike the addFirst method that throws an exception if it cannot insert the element due to capacity restrictions etc this method returns false if it cannot insert the element.

Parameters

<i>element</i>	The element to add to this Deque (p. 1196).
----------------	--

Returns

true if the element was added, false otherwise.

Exceptions

<i>NullPointerException</i>	if the specified element is NULL and this deque is a Collection (p. 851) of pointers and does not permit null elements.
<i>IllegalArgumentException</i>	if some property of the specified element prevents it from being added to this deque.

Implements **decaf::util::Deque< E >** (p. 1202).

6.311.3.25 `template<typename E> virtual bool decaf::util::LinkedList< E >::offerLast (const E & element) [inline, virtual]`

This method attempts to insert the given element into the **Deque** (p. 1196) at the end.

Unlike the addLast method that throws an exception if it cannot insert the element due to capacity restrictions etc this method returns false if it cannot insert the element.

Parameters

<i>element</i>	The element to add to this Deque (p. 1196).
----------------	--

Returns

true if the element was added, false otherwise.

Exceptions

<i>NullPointerException</i>	if the specified element is NULL and this deque is a Collection (p. 851) of pointers and does not permit null elements.
<i>IllegalArgumentException</i>	if some property of the specified element prevents it from being added to this deque.

Implements **decaf::util::Deque< E >** (p. 1203).

6.311.3.26 `template<typename E> LinkedList<E>& decaf::util::LinkedList< E >::operator= (const LinkedList< E > & list) [inline]`

6.311.3.27 `template<typename E> LinkedList<E>& decaf::util::LinkedList< E >::operator= (const Collection< E > & collection) [inline]`

6.311.3.28 `template<typename E> virtual bool decaf::util::LinkedList< E >::peek (E & result) const [inline, virtual]`

Gets but not removes the element in the head of the queue.

The result if successful is assigned to the result parameter.

Parameters

<i>result</i>	Reference to an instance of the contained type to assigned the removed value to.
---------------	--

Returns

true if the element at the head of the queue was removed and assigned to the result parameter.

Implements **decaf::util::Queue< E >** (p. 2225).

6.311.3.29 `template<typename E> virtual bool decaf::util::LinkedList< E >::peekFirst (E & value) const [inline, virtual]`

Retrieves the first element contained in this **Deque** (p. 1196) and assigns its value to the reference value passed the value however is not removed from the **Deque** (p. 1196).

If this call is successful it returns true. Unlike `getFirst` this method does not throw an exception if the **Deque** (p. 1196) is empty.

Returns

true if an element was assigned to the reference passed, false otherwise.

Implements **decaf::util::Deque< E >** (p. 1204).

6.311.3.30 `template<typename E> virtual bool decaf::util::LinkedList< E >::peekLast (E & value) const [inline, virtual]`

Retrieves the last element contained in this **Deque** (p. 1196) and assigns its value to the reference value passed the value however is not removed from the **Deque** (p. 1196).

If this call is successful it returns true. Unlike getLast this method does not throw an exception if the **Deque** (p. 1196) is empty.

Returns

true if an element was assigned to the reference passed, false otherwise.

Implements **decaf::util::Deque< E >** (p. 1204).

6.311.3.31 `template<typename E> virtual bool decaf::util::LinkedList< E >::poll (E & result) [inline, virtual]`

Gets and removes the element in the head of the queue.

If the operation succeeds the value of the element at the head of the **Queue** (p. 2222) is assigned to the result parameter and the method returns true. If the operation fails the method returns false and the value of the result parameter is undefined.

Parameters

<i>result</i>	Reference to an instance of the contained type to assigned the removed value to.
---------------	--

Returns

true if the element at the head of the queue was removed and assigned to the result parameter.

Implements **decaf::util::Queue< E >** (p. 2225).

6.311.3.32 `template<typename E> virtual bool decaf::util::LinkedList< E >::pollFirst (E & element) [inline, virtual]`

Removes the first element from the **Deque** (p. 1196) assigns it to the element reference passed.

Parameters

<i>element</i>	Reference to an variable that can be assigned the value of the head of this Deque (p. 1196).
----------------	---

Returns

true if an element was available to remove, false otherwise.

Implements **decaf::util::Deque**< **E** > (p. 1205).

6.311.3.33 `template<typename E> virtual bool decaf::util::LinkedList< E >::pollLast (E & element) [inline, virtual]`

Removes the last element from the **Deque** (p. 1196) assigns it to the element reference passed.

Parameters

<i>element</i>	Reference to an variable that can be assigned the value of the tail of this Deque (p. 1196).
----------------	---

Returns

true if an element was available to remove, false otherwise.

Implements **decaf::util::Deque**< **E** > (p. 1205).

6.311.3.34 `template<typename E> virtual E decaf::util::LinkedList< E >::pop () [inline, virtual]`

Treats this **Deque** (p. 1196) as a stack and attempts to pop an element off the top.

If there's no element to pop then a **NuSuchElementException** is thrown, otherwise the top element is removed and assigned to the reference passed.

This operation performs the same basic function as the **removeFirst** method.

Returns

the element at the front of this deque which would be the top of a stack.

Exceptions

NoSuchElementException (p. 1984)	if there is nothing on the top of the stack.
--	--

Implements **decaf::util::Deque**< **E** > (p. 1206).

6.311.3.35 `template<typename E> virtual void decaf::util::LinkedList< E >::push (const E & element) [inline, virtual]`

Pushes an element onto the stack represented by this deque (in other words, at the head of this deque) if it is possible to do so immediately without violating capacity restrictions, otherwise it throwing an `IllegalStateException` if no space is currently available.

This method performs the same basic operation as the `addFirst` method.

Parameters

<i>element</i>	The element to be pushed onto the Deque (p. 1196).
----------------	---

Exceptions

<i>IllegalStateException</i>	if the element cannot be added at this time due to capacity restrictions
<i>NullPointerException</i>	if the specified element is NULL and this deque is a Collection (p. 851) of pointers and does not permit null elements.
<i>IllegalArgumentException</i>	if some property of the specified element prevents it from being added to this deque.

Implements `decaf::util::Deque< E >` (p. 1206).

6.311.3.36 `template<typename E> virtual bool decaf::util::LinkedList< E >::remove (const E & value) [inline, virtual]`

Removes a single instance of the specified element from the collection.

More formally, removes an element `e` such that `(value == NULL ? e == NULL : value == e)`, if this collection contains one or more such elements. Returns true if this collection contained the specified element (or equivalently, if this collection changed as a result of the call).

Parameters

<i>value</i>	The reference to the element to remove from this Collection (p. 851).
--------------	--

Returns

true if the collection was changed, false otherwise.

Exceptions

<i>UnsupportedOperationException</i>	if this is an unmodifiable collection.
<i>NullPointerException</i>	if the Collection (p. 851) is a container of pointers and does not allow NULL values.

This implementation iterates over the collection looking for the specified element. If it finds the element, it removes the element from the collection using the iterator's remove method.

Note that this implementation throws an `UnsupportedOperationException` if the iterator returned by this collection's iterator method does not implement the remove method and this collection contains the specified object.

This implementation iterates over the collection looking for the specified element. If it finds the element, it removes the element from the collection using the iterator's remove method.

Note that this implementation throws an `UnsupportedOperationException` if the iterator returned by this collection's iterator method does not implement the remove method and this collection contains the specified object.

Reimplemented from **decaf::util::AbstractCollection**< E > (p. 116).

6.311.3.37 `template<typename E> virtual E decaf::util::LinkedList< E >::remove ()`
`[inline, virtual]`

Gets and removes the element in the head of the queue.

Throws a **NoSuchElementException** (p. 1984) if there is no element in the queue.

Returns

the element in the head of the queue.

Exceptions

NoSuchElementException (p. 1984)	if there is no element in the queue.
--	--------------------------------------

Implements **decaf::util::Queue**< E > (p. 2226).

6.311.3.38 `template<typename E> virtual E decaf::util::LinkedList< E >::removeFirst ()`
`[inline, virtual]`

Removes the topmost element from the **Deque** (p. 1196) and returns it.

Unlike the pollFirst method this method throws a `NoSuchElementException` if the **Deque** (p. 1196) is empty.

Returns

the element at the Head of the **Deque** (p. 1196).

Exceptions

NoSuchElementException (p. 1984)	if the Deque (p. 1196) is empty.
--	---

Implements **decaf::util::Deque< E >** (p. 1207).

6.311.3.39 `template<typename E> virtual bool decaf::util::LinkedList< E >::removeFirstOccurrence (const E & value) [inline, virtual]`

Removes the first occurrence of the specified element from this **Deque** (p. 1196).

If there is no matching element then the **Deque** (p. 1196) is left unchanged.

Parameters

<i>value</i>	The value to be removed from this Deque (p. 1196).
--------------	---

Returns

true if the **Deque** (p. 1196) was modified as a result of this operation.

Exceptions

<i>NullPointerException</i>	if the specified element is NULL and this deque is a Collection (p. 851) of pointers and does not permit null elements.
-----------------------------	--

Implements **decaf::util::Deque< E >** (p. 1208).

Referenced by `decaf::util::LinkedList< cms::Connection * >::remove()`.

6.311.3.40 `template<typename E> virtual E decaf::util::LinkedList< E >::removeLast () [inline, virtual]`

Removes the last element from the **Deque** (p. 1196) and returns it.

Unlike the pollLast method this method throws a NoSuchElementException if the **Deque** (p. 1196) is empty.

Returns

the element at the Tail of the **Deque** (p. 1196).

Exceptions

NoSuchElementException (p. 1984)	if the Deque (p. 1196) is empty.
--	---

Implements **decaf::util::Deque**< **E** > (p. 1208).

6.311.3.41 `template<typename E> virtual bool decaf::util::LinkedList< E
>::removeLastOccurrence (const E & value) [inline, virtual]`

Removes the last occurrence of the specified element from this **Deque** (p. 1196).

If there is no matching element then the **Deque** (p. 1196) is left unchanged.

Parameters

<i>value</i>	The value to be removed from this Deque (p. 1196).
--------------	---

Returns

true if the **Deque** (p. 1196) was modified as a result of this operation.

Exceptions

<i>NullPointerException</i>	if the specified element is NULL and this deque is a Collection (p. 851) of pointers and does not permit null elements.
-----------------------------	--

Implements **decaf::util::Deque**< **E** > (p. 1209).

6.311.3.42 `template<typename E> virtual E decaf::util::LinkedList< E >::set (int index ,
const E & element) [inline, virtual]`

Replaces the element at the specified position in this list with the specified element.

Parameters

<i>index</i>	The index of the element to replace.
<i>element</i>	The element to be stored at the specified position.

Returns

the element previously at the specified position.

Exceptions

<i>IndexOutOfBoundsException</i>	if the index given is less than zero or greater than the List (p. 1658) size.
<i>UnsupportedOperationException</i>	if this is an unmodifiable collection.
<i>NullPointerException</i>	if the Collection (p. 851) is a container of pointers and does not allow NULL values.
<i>IllegalArgumentException</i>	if some property of the element prevents it from being added to this collection

<i>IllegalStateException</i>	if the element cannot be added at this time due to insertion restrictions.
------------------------------	--

This implementation first gets a list iterator pointing to the indexed element (with `listIterator(index)`). Then, it gets the current element using **ListIterator.next** (p. 1560) and replaces it with **ListIterator.set** (p. 1674).

This implementation first gets a list iterator pointing to the indexed element (with `listIterator(index)`). Then, it gets the current element using **ListIterator.next** (p. 1560) and replaces it with **ListIterator.set** (p. 1674).

Reimplemented from **decaf::util::AbstractSequentialList< E >** (p. 151).

6.311.3.43 `template<typename E> virtual int decaf::util::LinkedList< E >::size ()`
`const [inline, virtual]`

Returns the number of elements in this collection.

If this collection contains more than `Integer::MAX_VALUE` elements, returns `Integer::MAX_VALUE`.

Returns

the number of elements in this collection

Implements **decaf::util::Collection< E >** (p. 864).

6.311.3.44 `template<typename E> virtual std::vector<E> decaf::util::LinkedList< E`
`>::toArray () const [inline, virtual]`

Answers an STL vector containing copies of all elements contained in this **Collection** (p. 851).

All the elements in the array will not be referenced by the collection. The elements in the returned array will be sorted to the same order as those returned by the iterator of this collection itself if the collection guarantees the order.

Returns

an vector of copies of all the elements from this **Collection** (p. 851)

Reimplemented from **decaf::util::AbstractCollection< E >** (p. 119).

The documentation for this class was generated from the following file:

- `src/main/decaf/util/LinkedList.h`

6.312 decaf::util::List< E > Class Template Reference

An ordered collection (also known as a sequence).

```
#include <src/main/decaf/util/List.h>
```

Inheritance diagram for decaf::util::List< E >:

Public Member Functions

- **List** ()
- virtual **~List** ()
- virtual **ListIterator**< E > * **listIterator** ()=0
- virtual **ListIterator**< E > * **listIterator** () const =0
- virtual **ListIterator**< E > * **listIterator** (int index)=0
- virtual **ListIterator**< E > * **listIterator** (int index) const =0
- virtual int **indexOf** (const E &value) const =0

Returns the index of the first occurrence of the specified element in this list, or -1 if this list does not contain the element.
- virtual int **lastIndexOf** (const E &value) const =0

Returns the index of the last occurrence of the specified element in this list, or -1 if this list does not contain the element.
- virtual E **get** (int index) const =0

Gets the element contained at position passed.
- virtual E **set** (int index, const E &element)=0

Replaces the element at the specified position in this list with the specified element.
- virtual void **add** (int index, const E &element)=0

Inserts the specified element at the specified position in this list.
- virtual bool **addAll** (int index, const **Collection**< E > &source)=0

Inserts all of the elements in the specified collection into this list at the specified position (optional operation).
- virtual E **removeAt** (int index)=0

Removes the element at the specified position in this list.

6.312.1 Detailed Description

```
template<typename E>class decaf::util::List< E >
```

An ordered collection (also known as a sequence).

The user of this interface has precise control over where in the list each element is inserted. The user can access elements by their integer index (position in the list), and search for elements in the list.

Unlike sets, lists typically allow duplicate elements. More formally, lists typically allow pairs of elements e1 and e2 such that e1.equals(e2), and they typically allow multiple null elements if they allow null elements at all. It is not inconceivable that someone might wish to implement a list that prohibits duplicates, by throwing runtime exceptions when the user attempts to insert them, but we expect this usage to be rare.

6.312.2 Constructor & Destructor Documentation

6.312.2.1 `template<typename E> decaf::util::List< E >::List () [inline]`

6.312.2.2 `template<typename E> virtual decaf::util::List< E >::~~List () [inline, virtual]`

6.312.3 Member Function Documentation

6.312.3.1 `template<typename E> virtual void decaf::util::List< E >::add (int index, const E & element) [pure virtual]`

Inserts the specified element at the specified position in this list.

Shifts the element currently at that position (if any) and any subsequent elements to the right (adds one to their indices).

Parameters

<i>index</i>	The index at which the specified element is to be inserted.
<i>element</i>	The element to be inserted in this List (p. 1658).

Exceptions

<i>IndexOutOfBoundsException</i>	if the index is greater than size of the List (p. 1658).
<i>UnsupportedOperationException</i>	if this is an unmodifiable collection.
<i>NullPointerException</i>	if the Collection (p. 851) is a container of pointers and does not allow NULL values.
<i>IllegalArgumentException</i>	if some property of the element prevents it from being added to this collection
<i>IllegalStateException</i>	if the element cannot be added at this time due to insertion restrictions.

Implemented in **decaf::util::concurrent::CopyOnWriteArrayList< E >** (p. 1034), **decaf::util::concurrent::CopyOnWriteArrayList< ServiceListener * >** (p. 1034), **decaf::util::StlList< E >** (p. 2542), **decaf::util::ArrayList< E >** (p. 449), **decaf::util::LinkedList< E >** (p. 1640), **decaf::util::LinkedList< Pointer< Transport > >** (p. 1640), **decaf::util::LinkedList< cms::MessageConsumer * >** (p. 1640), **decaf::util::LinkedList< CompositeTask * >** (p. 1640), **decaf::util::LinkedList< URI >** (p. 1640), **decaf::util::LinkedList< Pointer< MessageDispatch > >** (p. 1640), **decaf::util::LinkedList< Pointer< DestinationInfo > >** (p. 1640), **decaf::util::LinkedList< PrimitiveValueNode >** (p. 1640), **decaf::util::LinkedList< Pointer< Command > >** (p. 1640), **decaf::util::LinkedList< Pointer< BackupTransport > >** (p. 1640), **decaf::util::LinkedList< cms::MessageProducer * >** (p. 1640), **decaf::util::LinkedList< cms::Destination * >** (p. 1640), **decaf::util::LinkedList< cms::Session * >** (p. 1640), **decaf::util::LinkedList< cms::Connection * >** (p. 1640), **decaf::util::AbstractSequentialList< E >** (p. 146), **decaf::util::Abstract-**

SequentialList< **Pointer**< **Transport** > > (p. 146), **decaf::util::AbstractSequentialList**< **cms::MessageConsumer** * > (p. 146), **decaf::util::AbstractSequentialList**< **CompositeTask** * > (p. 146), **decaf::util::AbstractSequentialList**< **URI** > (p. 146), **decaf::util::AbstractSequentialList**< **Pointer**< **MessageDispatch** > > (p. 146), **decaf::util::AbstractSequentialList**< **Pointer**< **DestinationInfo** > > (p. 146), **decaf::util::AbstractSequentialList**< **PrimitiveValueNode** > (p. 146), **decaf::util::AbstractSequentialList**< **Pointer**< **Command** > > (p. 146), **decaf::util::AbstractSequentialList**< **Pointer**< **BackupTransport** > > (p. 146), **decaf::util::AbstractSequentialList**< **cms::MessageProducer** * > (p. 146), **decaf::util::AbstractSequentialList**< **cms::Destination** * > (p. 146), **decaf::util::AbstractSequentialList**< **cms::Session** * > (p. 146), and **decaf::util::AbstractSequentialList**< **cms::Connection** * > (p. 146).

6.312.3.2 `template<typename E> virtual bool decaf::util::List< E >::addAll(int index, const Collection< E > & source) [pure virtual]`

Inserts all of the elements in the specified collection into this list at the specified position (optional operation).

Shifts the element currently at that position (if any) and any subsequent elements to the right (increases their indices). The new elements will appear in this list in the order that they are returned by the specified collection's iterator. The behavior of this operation is undefined if the specified collection is modified while the operation is in progress. (Note that this will occur if the specified collection is this list, and it's nonempty.)

Parameters

<i>index</i>	The index at which to insert the first element from the specified collection
<i>source</i>	The Collection (p. 851) containing elements to be added to this list

Returns

true if this list changed as a result of the call

Exceptions

<i>IndexOutOfBoundsException</i>	if the index given is less than zero or greater than the List (p. 1658) size.
<i>UnsupportedOperationException</i>	if this is an unmodifiable collection.
<i>NullPointerException</i>	if the Collection (p. 851) is a container of pointers and does not allow NULL values.
<i>IllegalArgumentException</i>	if some property of the element prevents it from being added to this collection
<i>IllegalStateException</i>	if the element cannot be added at this time due to insertion restrictions.

Implemented in **decaf::util::concurrent::CopyOnWriteArrayList**< **E** > (p. 1035),

decaf::util::concurrent::CopyOnWriteArrayList< ServiceListener * > (p. 1035), decaf::util::StlList< E > (p. 2544), decaf::util::AbstractList< E > (p. 126), decaf::util::AbstractList< Pointer< Transport > > (p. 126), decaf::util::AbstractList< cms::MessageConsumer * > (p. 126), decaf::util::AbstractList< CompositeTask * > (p. 126), decaf::util::AbstractList< URI > (p. 126), decaf::util::AbstractList< Pointer< MessageDispatch > > (p. 126), decaf::util::AbstractList< Pointer< DestinationInfo > > (p. 126), decaf::util::AbstractList< PrimitiveValueNode > (p. 126), decaf::util::AbstractList< Pointer< Command > > (p. 126), decaf::util::AbstractList< Pointer< BackupTransport > > (p. 126), decaf::util::AbstractList< cms::MessageProducer * > (p. 126), decaf::util::AbstractList< cms::Destination * > (p. 126), decaf::util::AbstractList< cms::Session * > (p. 126), decaf::util::AbstractList< cms::Connection * > (p. 126), decaf::util::ArrayList< E > (p. 451), decaf::util::LinkedList< E > (p. 1642), decaf::util::LinkedList< Pointer< Transport > > (p. 1642), decaf::util::LinkedList< cms::MessageConsumer * > (p. 1642), decaf::util::LinkedList< CompositeTask * > (p. 1642), decaf::util::LinkedList< URI > (p. 1642), decaf::util::LinkedList< Pointer< MessageDispatch > > (p. 1642), decaf::util::LinkedList< Pointer< DestinationInfo > > (p. 1642), decaf::util::LinkedList< PrimitiveValueNode > (p. 1642), decaf::util::LinkedList< Pointer< Command > > (p. 1642), decaf::util::LinkedList< Pointer< BackupTransport > > (p. 1642), decaf::util::LinkedList< cms::MessageProducer * > (p. 1642), decaf::util::LinkedList< cms::Destination * > (p. 1642), decaf::util::LinkedList< cms::Session * > (p. 1642), decaf::util::LinkedList< cms::Connection * > (p. 1642), decaf::util::AbstractSequentialList< E > (p. 147), decaf::util::AbstractSequentialList< Pointer< Transport > > (p. 147), decaf::util::AbstractSequentialList< cms::MessageConsumer * > (p. 147), decaf::util::AbstractSequentialList< CompositeTask * > (p. 147), decaf::util::AbstractSequentialList< URI > (p. 147), decaf::util::AbstractSequentialList< Pointer< MessageDispatch > > (p. 147), decaf::util::AbstractSequentialList< Pointer< DestinationInfo > > (p. 147), decaf::util::AbstractSequentialList< PrimitiveValueNode > (p. 147), decaf::util::AbstractSequentialList< Pointer< Command > > (p. 147), decaf::util::AbstractSequentialList< Pointer< BackupTransport > > (p. 147), decaf::util::AbstractSequentialList< cms::MessageProducer * > (p. 147), decaf::util::AbstractSequentialList< cms::Destination * > (p. 147), decaf::util::AbstractSequentialList< cms::Session * > (p. 147), and decaf::util::AbstractSequentialList< cms::Connection * > (p. 147).

6.312.3.3 `template<typename E> virtual E decaf::util::List< E >::get (int index) const`
`[pure virtual]`

Gets the element contained at position passed.

Parameters

<i>index</i>	The position to get.
--------------	----------------------

Returns

value at index specified.

Exceptions

<i>IndexOutOfBoundsException</i>	if the index given is less than zero or greater than the List (p. 1658) size.
----------------------------------	--

Implemented in **decaf::util::concurrent::CopyOnWriteArrayList< E >** (p. 1038), **decaf::util::concurrent::CopyOnWriteArrayList< ServiceListener * >** (p. 1038), **decaf::util::StlList< E >** (p. 2547), **decaf::util::ArrayList< E >** (p. 453), **decaf::util::LinkedList< E >** (p. 1646), **decaf::util::LinkedList< Pointer< Transport > >** (p. 1646), **decaf::util::LinkedList< cms::MessageConsumer * >** (p. 1646), **decaf::util::LinkedList< CompositeTask * >** (p. 1646), **decaf::util::LinkedList< URI >** (p. 1646), **decaf::util::LinkedList< Pointer< MessageDispatch > >** (p. 1646), **decaf::util::LinkedList< Pointer< DestinationInfo > >** (p. 1646), **decaf::util::LinkedList< PrimitiveValueNode >** (p. 1646), **decaf::util::LinkedList< Pointer< Command > >** (p. 1646), **decaf::util::LinkedList< Pointer< BackupTransport > >** (p. 1646), **decaf::util::LinkedList< cms::MessageProducer * >** (p. 1646), **decaf::util::LinkedList< cms::Destination * >** (p. 1646), **decaf::util::LinkedList< cms::Session * >** (p. 1646), **decaf::util::LinkedList< cms::Connection * >** (p. 1646), **decaf::util::AbstractSequentialList< E >** (p. 148), **decaf::util::AbstractSequentialList< Pointer< Transport > >** (p. 148), **decaf::util::AbstractSequentialList< cms::MessageConsumer * >** (p. 148), **decaf::util::AbstractSequentialList< CompositeTask * >** (p. 148), **decaf::util::AbstractSequentialList< URI >** (p. 148), **decaf::util::AbstractSequentialList< Pointer< MessageDispatch > >** (p. 148), **decaf::util::AbstractSequentialList< Pointer< DestinationInfo > >** (p. 148), **decaf::util::AbstractSequentialList< PrimitiveValueNode >** (p. 148), **decaf::util::AbstractSequentialList< Pointer< Command > >** (p. 148), **decaf::util::AbstractSequentialList< Pointer< BackupTransport > >** (p. 148), **decaf::util::AbstractSequentialList< cms::MessageProducer * >** (p. 148), **decaf::util::AbstractSequentialList< cms::Destination * >** (p. 148), **decaf::util::AbstractSequentialList< cms::Session * >** (p. 148), and **decaf::util::AbstractSequentialList< cms::Connection * >** (p. 148).

6.312.3.4 `template<typename E> virtual int decaf::util::List< E >::indexOf (const E & value) const [pure virtual]`

Returns the index of the first occurrence of the specified element in this list, or -1 if this list does not contain the element.

More formally, returns the lowest index *i* such that `get(i) == value`, or -1 if there is no such index.

Parameters

<i>value</i>	The element to search for in this List (p. 1658).
--------------	--

Returns

the index of the first occurrence of the specified element in this list,

Exceptions

<i>NullPointerException</i>	if the Collection (p.851) is a container of pointers and does not allow NULL values.
-----------------------------	---

Implemented in **decaf::util::StlList< E >** (p.2547), **decaf::util::concurrent::CopyOnWriteArrayList< E >** (p.1039), **decaf::util::concurrent::CopyOnWriteArrayList< ServiceListener * >** (p.1039), **decaf::util::AbstractList< E >** (p.128), **decaf::util::AbstractList< Pointer< Transport > >** (p.128), **decaf::util::AbstractList< cms::MessageConsumer * >** (p.128), **decaf::util::AbstractList< CompositeTask * >** (p.128), **decaf::util::AbstractList< URI >** (p.128), **decaf::util::AbstractList< Pointer< MessageDispatch > >** (p.128), **decaf::util::AbstractList< Pointer< DestinationInfo > >** (p.128), **decaf::util::AbstractList< PrimitiveValueNode >** (p.128), **decaf::util::AbstractList< Pointer< Command > >** (p.128), **decaf::util::AbstractList< Pointer< BackupTransport > >** (p.128), **decaf::util::AbstractList< cms::MessageProducer * >** (p.128), **decaf::util::AbstractList< cms::Destination * >** (p.128), **decaf::util::AbstractList< cms::Session * >** (p.128), **decaf::util::AbstractList< cms::Connection * >** (p.128), **decaf::util::ArrayList< E >** (p.453), **decaf::util::LinkedList< E >** (p.1647), **decaf::util::LinkedList< Pointer< Transport > >** (p.1647), **decaf::util::LinkedList< cms::MessageConsumer * >** (p.1647), **decaf::util::LinkedList< CompositeTask * >** (p.1647), **decaf::util::LinkedList< URI >** (p.1647), **decaf::util::LinkedList< Pointer< MessageDispatch > >** (p.1647), **decaf::util::LinkedList< Pointer< DestinationInfo > >** (p.1647), **decaf::util::LinkedList< PrimitiveValueNode >** (p.1647), **decaf::util::LinkedList< Pointer< Command > >** (p.1647), **decaf::util::LinkedList< Pointer< BackupTransport > >** (p.1647), **decaf::util::LinkedList< cms::MessageProducer * >** (p.1647), **decaf::util::LinkedList< cms::Destination * >** (p.1647), **decaf::util::LinkedList< cms::Session * >** (p.1647), and **decaf::util::LinkedList< cms::Connection * >** (p.1647).

6.312.3.5 `template<typename E> virtual int decaf::util::List< E >::lastIndexOf(const E & value) const [pure virtual]`

Returns the index of the last occurrence of the specified element in this list, or -1 if this list does not contain the element.

More formally, returns the highest index *i* such that `get(i) == value` or -1 if there is no such index.

Parameters

<i>value</i>	The element to search for in this List (p.1658).
--------------	---

Returns

the index of the last occurrence of the specified element in this list.

Exceptions

<i>NullPointerException</i>	if the Collection (p.851) is a container of pointers and does not allow NULL values.
-----------------------------	---

Implemented in `decaf::util::StlList< E >` (p. 2548), `decaf::util::concurrent::CopyOnWriteArrayList< E >` (p. 1040), `decaf::util::concurrent::CopyOnWriteArrayList< ServiceListener * >` (p. 1040), `decaf::util::AbstractList< E >` (p. 129), `decaf::util::AbstractList< Pointer< Transport > >` (p. 129), `decaf::util::AbstractList< cms::MessageConsumer * >` (p. 129), `decaf::util::AbstractList< CompositeTask * >` (p. 129), `decaf::util::AbstractList< URI >` (p. 129), `decaf::util::AbstractList< Pointer< MessageDispatch > >` (p. 129), `decaf::util::AbstractList< Pointer< DestinationInfo > >` (p. 129), `decaf::util::AbstractList< PrimitiveValueNode >` (p. 129), `decaf::util::AbstractList< Pointer< Command > >` (p. 129), `decaf::util::AbstractList< Pointer< BackupTransport > >` (p. 129), `decaf::util::AbstractList< cms::MessageProducer * >` (p. 129), `decaf::util::AbstractList< cms::Destination * >` (p. 129), `decaf::util::AbstractList< cms::Session * >` (p. 129), `decaf::util::AbstractList< cms::Connection * >` (p. 129), `decaf::util::ArrayList< E >` (p. 454), `decaf::util::LinkedList< E >` (p. 1648), `decaf::util::LinkedList< Pointer< Transport > >` (p. 1648), `decaf::util::LinkedList< cms::MessageConsumer * >` (p. 1648), `decaf::util::LinkedList< CompositeTask * >` (p. 1648), `decaf::util::LinkedList< URI >` (p. 1648), `decaf::util::LinkedList< Pointer< MessageDispatch > >` (p. 1648), `decaf::util::LinkedList< Pointer< DestinationInfo > >` (p. 1648), `decaf::util::LinkedList< PrimitiveValueNode >` (p. 1648), `decaf::util::LinkedList< Pointer< Command > >` (p. 1648), `decaf::util::LinkedList< Pointer< BackupTransport > >` (p. 1648), `decaf::util::LinkedList< cms::MessageProducer * >` (p. 1648), `decaf::util::LinkedList< cms::Destination * >` (p. 1648), `decaf::util::LinkedList< cms::Session * >` (p. 1648), and `decaf::util::LinkedList< cms::Connection * >` (p. 1648).

6.312.3.6 `template<typename E> virtual ListIterator<E>* decaf::util::List< E >::listIterator () [pure virtual]`

Returns

a list iterator over the elements in this list (in proper sequence).

Implemented in `decaf::util::concurrent::CopyOnWriteArrayList< E >` (p. 1041), `decaf::util::concurrent::CopyOnWriteArrayList< ServiceListener * >` (p. 1041), `decaf::util::AbstractList< E >` (p. 130), `decaf::util::AbstractList< Pointer< Transport > >` (p. 130), `decaf::util::AbstractList< cms::MessageConsumer * >` (p. 130), `decaf::util::AbstractList< CompositeTask * >` (p. 130), `decaf::util::AbstractList< URI >` (p. 130), `decaf::util::AbstractList< Pointer< MessageDispatch > >` (p. 130), `decaf::util::AbstractList< Pointer< DestinationInfo > >` (p. 130), `decaf::util::AbstractList< PrimitiveValueNode >` (p. 130), `decaf::util::AbstractList< Pointer< Command > >` (p. 130), `decaf::util::AbstractList< Pointer< BackupTransport > >` (p. 130), `decaf::util::AbstractList< cms::MessageProducer * >` (p. 130), `decaf::util::AbstractList< cms::Destination * >` (p. 130), `decaf::util::AbstractList< cms::Session * >` (p. 130), `decaf::util::AbstractList< cms::Connection * >` (p. 130), `decaf::util::StlList< E >` (p. 2549), `decaf::util::AbstractSequentialList< E >` (p. 149), `decaf::util::AbstractSequentialList< Pointer< Transport > >` (p. 149), `decaf::util::AbstractSequentialList< cms::MessageConsumer * >` (p. 149), `decaf::util::AbstractSequentialList< CompositeTask * >` (p. 149), `decaf::util::AbstractSequentialList< URI >` (p. 149), `decaf::util::AbstractSequentialList< Pointer< MessageDispatch >`

> (p. 149), **decaf::util::AbstractSequentialList< Pointer< DestinationInfo > >** (p. 149), **decaf::util::AbstractSequentialList< PrimitiveValueNode >** (p. 149), **decaf::util::AbstractSequentialList< Pointer< Command > >** (p. 149), **decaf::util::AbstractSequentialList< Pointer< BackupTransport > >** (p. 149), **decaf::util::AbstractSequentialList< cms::MessageProducer * >** (p. 149), **decaf::util::AbstractSequentialList< cms::Destination * >** (p. 149), **decaf::util::AbstractSequentialList< cms::Session * >** (p. 149), and **decaf::util::AbstractSequentialList< cms::Connection * >** (p. 149).

6.312.3.7 `template<typename E> virtual ListIterator<E>* decaf::util::List< E >::listIterator ()const [pure virtual]`

Implemented in **decaf::util::concurrent::CopyOnWriteArrayList< E >** (p. 1042), **decaf::util::concurrent::CopyOnWriteArrayList< ServiceListener * >** (p. 1042), **decaf::util::AbstractList< E >** (p. 131), **decaf::util::AbstractList< Pointer< Transport > >** (p. 131), **decaf::util::AbstractList< cms::MessageConsumer * >** (p. 131), **decaf::util::AbstractList< CompositeTask * >** (p. 131), **decaf::util::AbstractList< URI >** (p. 131), **decaf::util::AbstractList< Pointer< MessageDispatch > >** (p. 131), **decaf::util::AbstractList< Pointer< DestinationInfo > >** (p. 131), **decaf::util::AbstractList< PrimitiveValueNode >** (p. 131), **decaf::util::AbstractList< Pointer< Command > >** (p. 131), **decaf::util::AbstractList< Pointer< BackupTransport > >** (p. 131), **decaf::util::AbstractList< cms::MessageProducer * >** (p. 131), **decaf::util::AbstractList< cms::Destination * >** (p. 131), **decaf::util::AbstractList< cms::Session * >** (p. 131), **decaf::util::AbstractList< cms::Connection * >** (p. 131), **decaf::util::StlList< E >** (p. 2549), **decaf::util::AbstractSequentialList< E >** (p. 149), **decaf::util::AbstractSequentialList< Pointer< Transport > >** (p. 149), **decaf::util::AbstractSequentialList< cms::MessageConsumer * >** (p. 149), **decaf::util::AbstractSequentialList< CompositeTask * >** (p. 149), **decaf::util::AbstractSequentialList< URI >** (p. 149), **decaf::util::AbstractSequentialList< Pointer< MessageDispatch > >** (p. 149), **decaf::util::AbstractSequentialList< Pointer< DestinationInfo > >** (p. 149), **decaf::util::AbstractSequentialList< PrimitiveValueNode >** (p. 149), **decaf::util::AbstractSequentialList< Pointer< Command > >** (p. 149), **decaf::util::AbstractSequentialList< Pointer< BackupTransport > >** (p. 149), **decaf::util::AbstractSequentialList< cms::MessageProducer * >** (p. 149), **decaf::util::AbstractSequentialList< cms::Destination * >** (p. 149), **decaf::util::AbstractSequentialList< cms::Session * >** (p. 149), and **decaf::util::AbstractSequentialList< cms::Connection * >** (p. 149).

6.312.3.8 `template<typename E> virtual ListIterator<E>* decaf::util::List< E >::listIterator (int index) [pure virtual]`

Parameters

<i>index</i>	index of first element to be returned from the list iterator (by a call to the next method).
--------------	--

Returns

a list iterator of the elements in this list (in proper sequence), starting at the specified position in this list. The specified index indicates the first element that would be returned by an initial call to next. An initial call to previous would return the element with the specified index minus one.

Exceptions

<i>IndexOutOfBoundsException</i>	if the index is out of range (index < 0 index > size() (p. 864))
----------------------------------	--

Implemented in **decaf::util::LinkedList< E >** (p. 1649), **decaf::util::LinkedList< Pointer< Transport > >** (p. 1649), **decaf::util::LinkedList< cms::MessageConsumer * >** (p. 1649), **decaf::util::LinkedList< CompositeTask * >** (p. 1649), **decaf::util::LinkedList< URI >** (p. 1649), **decaf::util::LinkedList< Pointer< MessageDispatch > >** (p. 1649), **decaf::util::LinkedList< Pointer< DestinationInfo > >** (p. 1649), **decaf::util::LinkedList< PrimitiveValueNode >** (p. 1649), **decaf::util::LinkedList< Pointer< Command > >** (p. 1649), **decaf::util::LinkedList< Pointer< BackupTransport > >** (p. 1649), **decaf::util::LinkedList< cms::MessageProducer * >** (p. 1649), **decaf::util::LinkedList< cms::Destination * >** (p. 1649), **decaf::util::LinkedList< cms::Session * >** (p. 1649), **decaf::util::LinkedList< cms::Connection * >** (p. 1649), **decaf::util::concurrent::CopyOnWriteArrayList< E >** (p. 1042), **decaf::util::concurrent::CopyOnWriteArrayList< ServiceListener * >** (p. 1042), **decaf::util::AbstractList< E >** (p. 131), **decaf::util::AbstractList< Pointer< Transport > >** (p. 131), **decaf::util::AbstractList< cms::MessageConsumer * >** (p. 131), **decaf::util::AbstractList< CompositeTask * >** (p. 131), **decaf::util::AbstractList< URI >** (p. 131), **decaf::util::AbstractList< Pointer< MessageDispatch > >** (p. 131), **decaf::util::AbstractList< Pointer< DestinationInfo > >** (p. 131), **decaf::util::AbstractList< PrimitiveValueNode >** (p. 131), **decaf::util::AbstractList< Pointer< Command > >** (p. 131), **decaf::util::AbstractList< Pointer< BackupTransport > >** (p. 131), **decaf::util::AbstractList< cms::MessageProducer * >** (p. 131), **decaf::util::AbstractList< cms::Destination * >** (p. 131), **decaf::util::AbstractList< cms::Session * >** (p. 131), **decaf::util::AbstractList< cms::Connection * >** (p. 131), **decaf::util::StlList< E >** (p. 2549), **decaf::util::AbstractSequentialList< E >** (p. 149), **decaf::util::AbstractSequentialList< Pointer< Transport > >** (p. 149), **decaf::util::AbstractSequentialList< cms::MessageConsumer * >** (p. 149), **decaf::util::AbstractSequentialList< CompositeTask * >** (p. 149), **decaf::util::AbstractSequentialList< URI >** (p. 149), **decaf::util::AbstractSequentialList< Pointer< MessageDispatch > >** (p. 149), **decaf::util::AbstractSequentialList< Pointer< DestinationInfo > >** (p. 149), **decaf::util::AbstractSequentialList< PrimitiveValueNode >** (p. 149), **decaf::util::AbstractSequentialList< Pointer< Command > >** (p. 149), **decaf::util::AbstractSequentialList< Pointer< BackupTransport > >** (p. 149), **decaf::util::AbstractSequentialList< cms::MessageProducer * >** (p. 149), **decaf::util::AbstractSequentialList< cms::Destination * >** (p. 149), **decaf::util::AbstractSequentialList< cms::Session * >** (p. 149), and **decaf::util::AbstractSequentialList< cms::Connection * >** (p. 149).

6.312.3.9 `template<typename E> virtual ListIterator<E>* decaf::util::List< E >::listIterator (int index) const [pure virtual]`

Implemented in `decaf::util::LinkedList< E >` (p. 1649), `decaf::util::LinkedList< Pointer< Transport > >` (p. 1649), `decaf::util::LinkedList< cms::MessageConsumer * >` (p. 1649), `decaf::util::LinkedList< CompositeTask * >` (p. 1649), `decaf::util::LinkedList< URI >` (p. 1649), `decaf::util::LinkedList< Pointer< MessageDispatch > >` (p. 1649), `decaf::util::LinkedList< Pointer< DestinationInfo > >` (p. 1649), `decaf::util::LinkedList< PrimitiveValueNode >` (p. 1649), `decaf::util::LinkedList< Pointer< Command > >` (p. 1649), `decaf::util::LinkedList< Pointer< BackupTransport > >` (p. 1649), `decaf::util::LinkedList< cms::MessageProducer * >` (p. 1649), `decaf::util::LinkedList< cms::Destination * >` (p. 1649), `decaf::util::LinkedList< cms::Session * >` (p. 1649), `decaf::util::LinkedList< cms::Connection * >` (p. 1649), `decaf::util::concurrent::CopyOnWriteArrayList< E >` (p. 1042), `decaf::util::concurrent::CopyOnWriteArrayList< ServiceListener * >` (p. 1042), `decaf::util::AbstractList< E >` (p. 132), `decaf::util::AbstractList< Pointer< Transport > >` (p. 132), `decaf::util::AbstractList< cms::MessageConsumer * >` (p. 132), `decaf::util::AbstractList< CompositeTask * >` (p. 132), `decaf::util::AbstractList< URI >` (p. 132), `decaf::util::AbstractList< Pointer< MessageDispatch > >` (p. 132), `decaf::util::AbstractList< Pointer< DestinationInfo > >` (p. 132), `decaf::util::AbstractList< PrimitiveValueNode >` (p. 132), `decaf::util::AbstractList< Pointer< Command > >` (p. 132), `decaf::util::AbstractList< Pointer< BackupTransport > >` (p. 132), `decaf::util::AbstractList< cms::MessageProducer * >` (p. 132), `decaf::util::AbstractList< cms::Destination * >` (p. 132), `decaf::util::AbstractList< cms::Session * >` (p. 132), `decaf::util::AbstractList< cms::Connection * >` (p. 132), `decaf::util::StlList< E >` (p. 2550), `decaf::util::AbstractSequentialList< E >` (p. 150), `decaf::util::AbstractSequentialList< Pointer< Transport > >` (p. 150), `decaf::util::AbstractSequentialList< cms::MessageConsumer * >` (p. 150), `decaf::util::AbstractSequentialList< CompositeTask * >` (p. 150), `decaf::util::AbstractSequentialList< URI >` (p. 150), `decaf::util::AbstractSequentialList< Pointer< MessageDispatch > >` (p. 150), `decaf::util::AbstractSequentialList< Pointer< DestinationInfo > >` (p. 150), `decaf::util::AbstractSequentialList< PrimitiveValueNode >` (p. 150), `decaf::util::AbstractSequentialList< Pointer< Command > >` (p. 150), `decaf::util::AbstractSequentialList< Pointer< BackupTransport > >` (p. 150), `decaf::util::AbstractSequentialList< cms::MessageProducer * >` (p. 150), `decaf::util::AbstractSequentialList< cms::Destination * >` (p. 150), `decaf::util::AbstractSequentialList< cms::Session * >` (p. 150), and `decaf::util::AbstractSequentialList< cms::Connection * >` (p. 150).

6.312.3.10 `template<typename E> virtual E decaf::util::List< E >::removeAt (int index) [pure virtual]`

Removes the element at the specified position in this list.

Shifts any subsequent elements to the left (subtracts one from their indices). Returns the element that was removed from the list.

Parameters

<i>index</i>	- the index of the element to be removed.
--------------	---

Returns

the element previously at the specified position.

Exceptions

<i>IndexOutOfBoundsException</i>	if the index given is less than zero or greater than the List (p. 1658) size.
<i>UnsupportedOperationException</i>	if this is an unmodifiable collection.

Implemented in **decaf::util::concurrent::CopyOnWriteArrayList< E >** (p. 1045), **decaf::util::concurrent::CopyOnWriteArrayList< ServiceListener * >** (p. 1045), **decaf::util::StlList< E >** (p. 2550), **decaf::util::AbstractList< E >** (p. 133), **decaf::util::AbstractList< Pointer< Transport > >** (p. 133), **decaf::util::AbstractList< cms::MessageConsumer * >** (p. 133), **decaf::util::AbstractList< CompositeTask * >** (p. 133), **decaf::util::AbstractList< URI >** (p. 133), **decaf::util::AbstractList< Pointer< MessageDispatch > >** (p. 133), **decaf::util::AbstractList< Pointer< DestinationInfo > >** (p. 133), **decaf::util::AbstractList< PrimitiveValueNode >** (p. 133), **decaf::util::AbstractList< Pointer< Command > >** (p. 133), **decaf::util::AbstractList< Pointer< BackupTransport > >** (p. 133), **decaf::util::AbstractList< cms::MessageProducer * >** (p. 133), **decaf::util::AbstractList< cms::Destination * >** (p. 133), **decaf::util::AbstractList< cms::Session * >** (p. 133), **decaf::util::AbstractList< cms::Connection * >** (p. 133), **decaf::util::ArrayList< E >** (p. 456), **decaf::util::AbstractSequentialList< E >** (p. 150), **decaf::util::AbstractSequentialList< Pointer< Transport > >** (p. 150), **decaf::util::AbstractSequentialList< cms::MessageConsumer * >** (p. 150), **decaf::util::AbstractSequentialList< CompositeTask * >** (p. 150), **decaf::util::AbstractSequentialList< URI >** (p. 150), **decaf::util::AbstractSequentialList< Pointer< MessageDispatch > >** (p. 150), **decaf::util::AbstractSequentialList< Pointer< DestinationInfo > >** (p. 150), **decaf::util::AbstractSequentialList< PrimitiveValueNode >** (p. 150), **decaf::util::AbstractSequentialList< Pointer< Command > >** (p. 150), **decaf::util::AbstractSequentialList< Pointer< BackupTransport > >** (p. 150), **decaf::util::AbstractSequentialList< cms::MessageProducer * >** (p. 150), **decaf::util::AbstractSequentialList< cms::Destination * >** (p. 150), **decaf::util::AbstractSequentialList< cms::Session * >** (p. 150), and **decaf::util::AbstractSequentialList< cms::Connection * >** (p. 150).

```
6.312.3.11  template<typename E> virtual E decaf::util::List< E >::set ( int index, const E
            & element ) [pure virtual]
```

Replaces the element at the specified position in this list with the specified element.

Parameters

<i>index</i>	The index of the element to replace.
<i>element</i>	The element to be stored at the specified position.

Returns

the element previously at the specified position.

Exceptions

<i>IndexOutOfBoundsException</i>	if the index given is less than zero or greater than the List (p. 1658) size.
<i>UnsupportedOperationException</i>	if this is an unmodifiable collection.
<i>NullPointerException</i>	if the Collection (p. 851) is a container of pointers and does not allow NULL values.
<i>IllegalArgumentException</i>	if some property of the element prevents it from being added to this collection
<i>IllegalStateException</i>	if the element cannot be added at this time due to insertion restrictions.

Implemented in **decaf::util::concurrent::CopyOnWriteArrayList< E >** (p. 1046), **decaf::util::concurrent::CopyOnWriteArrayList< ServiceListener * >** (p. 1046), **decaf::util::StlList< E >** (p. 2551), **decaf::util::ArrayList< E >** (p. 456), **decaf::util::LinkedList< E >** (p. 1657), **decaf::util::LinkedList< Pointer< Transport > >** (p. 1657), **decaf::util::LinkedList< cms::MessageConsumer * >** (p. 1657), **decaf::util::LinkedList< CompositeTask * >** (p. 1657), **decaf::util::LinkedList< URI >** (p. 1657), **decaf::util::LinkedList< Pointer< MessageDispatch > >** (p. 1657), **decaf::util::LinkedList< Pointer< DestinationInfo > >** (p. 1657), **decaf::util::LinkedList< PrimitiveValueNode >** (p. 1657), **decaf::util::LinkedList< Pointer< Command > >** (p. 1657), **decaf::util::LinkedList< Pointer< BackupTransport > >** (p. 1657), **decaf::util::LinkedList< cms::MessageProducer * >** (p. 1657), **decaf::util::LinkedList< cms::Destination * >** (p. 1657), **decaf::util::LinkedList< cms::Session * >** (p. 1657), **decaf::util::LinkedList< cms::Connection * >** (p. 1657), **decaf::util::AbstractSequentialList< E >** (p. 151), **decaf::util::AbstractSequentialList< Pointer< Transport > >** (p. 151), **decaf::util::AbstractSequentialList< cms::MessageConsumer * >** (p. 151), **decaf::util::AbstractSequentialList< CompositeTask * >** (p. 151), **decaf::util::AbstractSequentialList< URI >** (p. 151), **decaf::util::AbstractSequentialList< Pointer< MessageDispatch > >** (p. 151), **decaf::util::AbstractSequentialList< Pointer< DestinationInfo > >** (p. 151), **decaf::util::AbstractSequentialList< PrimitiveValueNode >** (p. 151), **decaf::util::AbstractSequentialList< Pointer< Command > >** (p. 151), **decaf::util::AbstractSequentialList< Pointer< BackupTransport > >** (p. 151), **decaf::util::AbstractSequentialList< cms::MessageProducer * >** (p. 151), **decaf::util::AbstractSequentialList< cms::Destination * >** (p. 151), **decaf::util::AbstractSequentialList< cms::Session * >** (p. 151), and **decaf::util::AbstractSequentialList< cms::Connection * >** (p. 151).

The documentation for this class was generated from the following file:

- `src/main/decaf/util/List.h`

6.313 `decaf::util::ListIterator< E >` Class Template Reference

An iterator for lists that allows the programmer to traverse the list in either direction, modify the list during iteration, and obtain the iterator's current position in the list.

```
#include <src/main/decaf/util/ListIterator.h>
```

Inheritance diagram for `decaf::util::ListIterator< E >`:

Public Member Functions

- virtual `~ListIterator ()`
- virtual void **add** (const E &e)=0
Inserts the specified element into the list (optional operation).
- virtual void **set** (const E &e)=0
Replaces the last element returned by next or previous with the specified element (optional operation).
- virtual bool **hasPrevious** () const =0
Returns true if this list iterator has more elements when traversing the list in the reverse direction.
- virtual E **previous** ()=0
Returns the previous element in the list.
- virtual int **nextIndex** () const =0
Returns the index of the element that would be returned by a subsequent call to next.
- virtual int **previousIndex** () const =0
Returns the index of the element that would be returned by a subsequent call to previous.

6.313.1 Detailed Description

```
template<typename E>class decaf::util::ListIterator< E >
```

An iterator for lists that allows the programmer to traverse the list in either direction, modify the list during iteration, and obtain the iterator's current position in the list.

Note that the **remove()** (p. 1560) and **set(Object)** methods are not defined in terms of the cursor position; they are defined to operate on the last element returned by a call to **next()** (p. 1560) or **previous()** (p. 1673).

6.313.2 Constructor & Destructor Documentation

6.313.2.1 `template<typename E> virtual decaf::util::ListIterator< E >::~~ListIterator () [inline, virtual]`

6.313.3 Member Function Documentation

6.313.3.1 `template<typename E> virtual void decaf::util::ListIterator< E >::add (const E & e) [pure virtual]`

Inserts the specified element into the list (optional operation).

The element is inserted immediately before the next element that would be returned by next, if any, and after the next element that would be returned by previous, if any. (If the list contains no elements, the new element becomes the sole element on the list.) The new element is inserted before the implicit cursor: a subsequent call to next would be unaffected, and a subsequent call to previous would return the new element. (This call increases by one the value that would be returned by a call to nextIndex or previousIndex.)

Parameters

<code>e</code>	The element to insert into the List (p. 1658).
----------------	---

Exceptions

<i>Unsupported-OperationException</i>	if the add method is not supported by this list iterator.
<i>IllegalArgument-Exception</i>	if some aspect of this element prevents it from being added to this list.

6.313.3.2 `template<typename E> virtual bool decaf::util::ListIterator< E >::hasPrevious () const [pure virtual]`

Returns true if this list iterator has more elements when traversing the list in the reverse direction.

(In other words, returns true if previous would return an element rather than throwing an exception.)

Returns

true if the list iterator has more elements when traversing the list in the reverse direction.

Implemented in **decaf::util::concurrent::CopyOnWriteArrayList< E >::ArrayList-Iterator** (p. 460).

6.313.3.3 `template<typename E> virtual int decaf::util::ListIterator< E>::nextIndex () const [pure virtual]`

Returns the index of the element that would be returned by a subsequent call to next.
(Returns list size if the list iterator is at the end of the list.)

Returns

the index of the element that would be returned by a subsequent call to next, or list size if list iterator is at end of list.

Implemented in **decaf::util::concurrent::CopyOnWriteArrayList< E>::ArrayListIterator** (p. 460).

6.313.3.4 `template<typename E> virtual E decaf::util::ListIterator< E>::previous () [pure virtual]`

Returns the previous element in the list.

This method may be called repeatedly to iterate through the list backwards, or inter-mixed with calls to next to go back and forth. (Note that alternating calls to next and previous will return the same element repeatedly.)

Returns

the previous element in the list.

Exceptions

NoSuchElementException (p. 1984)	if the iteration has no previous element.
--	---

Implemented in **decaf::util::concurrent::CopyOnWriteArrayList< E>::ArrayListIterator** (p. 461).

6.313.3.5 `template<typename E> virtual int decaf::util::ListIterator< E>::previousIndex () const [pure virtual]`

Returns the index of the element that would be returned by a subsequent call to previous.

(Returns -1 if the list iterator is at the beginning of the list.)

Returns

the index of the element that would be returned by a subsequent call to previous, or -1 if list iterator is at beginning of list.

Implemented in **decaf::util::concurrent::CopyOnWriteArrayList< E >::ArrayListIterator** (p. 461).

6.313.3.6 `template<typename E> virtual void decaf::util::ListIterator< E >::set (const E & e) [pure virtual]`

Replaces the last element returned by next or previous with the specified element (optional operation).

This call can be made only if neither **ListIterator.remove** (p. 1560) nor **ListIterator.add** (p. 1672) have been called after the last call to next or previous.

Parameters

<i>e</i>	The element with which to replace the last element returned by next or previous.
----------	--

Exceptions

<i>UnsupportedOperationException</i>	if the add method is not supported by this list iterator.
<i>IllegalArgumentException</i>	if some aspect of this element prevents it from being added to this list.
<i>IllegalStateException</i>	if neither next nor previous have been called, or remove or add have been called after the last call to next or previous.

The documentation for this class was generated from the following file:

- `src/main/decaf/util/ListIterator.h`

6.314 activemq::commands::LocalTransactionId Class Reference

```
#include <src/main/activemq/commands/LocalTransactionId.h>
```

Inheritance diagram for `activemq::commands::LocalTransactionId`:

Public Types

- typedef **decaf::lang::PointerComparator < LocalTransactionId > COMPARTOR**

Public Member Functions

- **LocalTransactionId ()**

- **LocalTransactionId** (const **LocalTransactionId** &other)
- virtual **~LocalTransactionId** ()
- virtual unsigned char **getDataStructureType** () const
*Get the **DataStructure** (p. 1133) Type as defined in CommandTypes.h.*
- virtual **LocalTransactionId** * **cloneDataStructure** () const
Clone this object and return a new instance that the caller now owns, this will be an exact copy of this one.
- virtual void **copyDataStructure** (const **DataStructure** *src)
Copy the contents of the passed object into this objects members, overwriting any existing data.
- virtual std::string **toString** () const
*Returns a string containing the information for this **DataStructure** (p. 1133) such as its type and value of its elements.*
- virtual bool **equals** (const **DataStructure** *value) const
*Compares the **DataStructure** (p. 1133) passed in to this one, and returns if they are equivalent.*
- virtual bool **isLocalTransactionId** () const
- virtual long long **getValue** () const
- virtual void **setValue** (long long value)
- virtual const **Pointer** < **ConnectionId** > & **getConnectionId** () const
- virtual **Pointer**< **ConnectionId** > & **getConnectionId** ()
- virtual void **setConnectionId** (const **Pointer**< **ConnectionId** > &connectionId)
- virtual int **compareTo** (const **LocalTransactionId** &value) const
- virtual bool **equals** (const **LocalTransactionId** &value) const
- virtual bool **operator==** (const **LocalTransactionId** &value) const
- virtual bool **operator<** (const **LocalTransactionId** &value) const
- **LocalTransactionId** & **operator=** (const **LocalTransactionId** &other)

Static Public Attributes

- static const unsigned char **ID_LOCALTRANSACTIONID** = 111

Protected Attributes

- long long value
- **Pointer**< **ConnectionId** > connectionId

6.314.1 Member Typedef Documentation

- 6.314.1.1 **typedef** decaf::lang::PointerComparator<LocalTransactionId>
activemq::commands::LocalTransactionId::COMPARATOR

Reimplemented from **activemq::commands::TransactionId** (p. 2767).

6.314.2 Constructor & Destructor Documentation

6.314.2.1 **activemq::commands::LocalTransactionId::LocalTransactionId** ()

6.314.2.2 **activemq::commands::LocalTransactionId::LocalTransactionId** (const **LocalTransactionId** & *other*)

6.314.2.3 **virtual activemq::commands::LocalTransactionId::~~LocalTransactionId** () [virtual]

6.314.3 Member Function Documentation

6.314.3.1 **virtual LocalTransactionId* activemq::commands::LocalTransactionId::cloneDataStructure** () const [virtual]

Clone this object and return a new instance that the caller now owns, this will be an exact copy of this one.

Returns

new copy of this object.

Reimplemented from **activemq::commands::TransactionId** (p. 2768).

6.314.3.2 **virtual int activemq::commands::LocalTransactionId::compareTo** (const **LocalTransactionId** & *value*) const [virtual]

6.314.3.3 **virtual void activemq::commands::LocalTransactionId::copyDataStructure** (const **DataStructure** * *src*) [virtual]

Copy the contents of the passed object into this objects members, overwriting any existing data.

Returns

src - Source Object

Reimplemented from **activemq::commands::TransactionId** (p. 2768).

6.314.3.4 **virtual bool activemq::commands::LocalTransactionId::equals** (const **DataStructure** * *value*) const [virtual]

Compares the **DataStructure** (p. 1133) passed in to this one, and returns if they are equivalent.

Equivalent here means that they are of the same type, and that each element of the objects are the same.

Returns

true if **DataStructure** (p. 1133)'s are Equal.

Reimplemented from **activemq::commands::TransactionId** (p. 2768).

6.314.3.5 `virtual bool activemq::commands::LocalTransactionId::equals (const LocalTransactionId & value) const` [virtual]

6.314.3.6 `virtual const Pointer<ConnectionId>& activemq::commands::LocalTransactionId::getConnectionId () const` [virtual]

6.314.3.7 `virtual Pointer<ConnectionId>& activemq::commands::LocalTransactionId::getConnectionId ()` [virtual]

6.314.3.8 `virtual unsigned char activemq::commands::LocalTransactionId::getDataStructureType () const` [virtual]

Get the **DataStructure** (p. 1133) Type as defined in CommandTypes.h.

Returns

The type of the data structure

Reimplemented from **activemq::commands::TransactionId** (p. 2769).

6.314.3.9 `virtual long long activemq::commands::LocalTransactionId::getValue () const` [virtual]

6.314.3.10 `virtual bool activemq::commands::LocalTransactionId::isLocalTransactionId () const` [inline, virtual]

Reimplemented from **activemq::commands::TransactionId** (p. 2769).

6.314.3.11 `virtual bool activemq::commands::LocalTransactionId::operator< (const LocalTransactionId & value) const` [virtual]

6.314.3.12 `LocalTransactionId& activemq::commands::LocalTransactionId::operator= (const LocalTransactionId & other)`

6.314.3.13 `virtual bool activemq::commands::LocalTransactionId::operator== (const LocalTransactionId & value) const` [virtual]

6.314.3.14 `virtual void activemq::commands::LocalTransactionId::setConnectionId (const Pointer< ConnectionId > & connectionId)` [virtual]

6.315 activemq::wireformat::openwire::marshal::generated::LocalTransactionId-Marshaller Class

Reference 1683

```
6.314.3.15 virtual void activemq::commands::LocalTransactionId::setValue ( long
           long value ) [virtual]

6.314.3.16 virtual std::string activemq::commands::LocalTransactionId::toString ( )
           const [virtual]
```

Returns a string containing the information for this **DataSet** (p. 1133) such as its type and value of its elements.

Returns

formatted string useful for debugging.

Reimplemented from **activemq::commands::TransactionId** (p. 2770).

6.314.4 Field Documentation

```
6.314.4.1 Pointer<ConnectionId> activemq::commands::LocalTransactionId-
           ::connectionId [protected]

6.314.4.2 const unsigned char activemq::commands::Local-
           TransactionId::ID_LOCALTRANSACTIONID = 111
           [static]

6.314.4.3 long long activemq::commands::LocalTransactionId::value
           [protected]
```

The documentation for this class was generated from the following file:

- src/main/activemq/commands/**LocalTransactionId.h**

6.315 activemq::wireformat::openwire::marshal::generated::LocalTransactionIdMarshaller Class Reference

Marshaling code for Open Wire Format for **LocalTransactionIdMarshaller** (p. 1678).

```
#include <src/main/activemq/wireformat/openwire/marshal/generated/-
LocalTransactionIdMarshaller.h>
```

Inheritance diagram for **activemq::wireformat::openwire::marshal::generated::LocalTransactionIdMarshaller**:

Public Member Functions

- **LocalTransactionIdMarshaller** ()

- virtual `~LocalTransactionIdMarshaller ()`
- virtual `commands::DataStructure * createObject () const`
Creates a new instance of the class that this class is a marshaling director for.
- virtual unsigned char `getDataStructureType () const`
Gets the DataStructureType that this class marshals/unmarshals.
- virtual void `tightUnmarshal (OpenWireFormat *wireFormat, commands::DataStructure *dataStructure, decaf::io::DataInputStream *dataIn, utils::BooleanStream *bs)`
Tight Un-marhsal to the given stream.
- virtual int `tightMarshal1 (OpenWireFormat *wireFormat, commands::DataStructure *dataStructure, utils::BooleanStream *bs)`
Tight Marhsal to the given stream.
- virtual void `tightMarshal2 (OpenWireFormat *wireFormat, commands::DataStructure *dataStructure, decaf::io::DataOutputStream *dataOut, utils::BooleanStream *bs)`
Tight Marhsal to the given stream.
- virtual void `looseUnmarshal (OpenWireFormat *wireFormat, commands::DataStructure *dataStructure, decaf::io::DataInputStream *dataIn)`
Loose Un-marhsal to the given stream.
- virtual void `looseMarshal (OpenWireFormat *wireFormat, commands::DataStructure *dataStructure, decaf::io::DataOutputStream *dataOut)`
Tight Marhsal to the given stream.

6.315.1 Detailed Description

Marshaling code for Open Wire Format for **LocalTransactionIdMarshaller** (p. 1678).

NOTE!: This file is auto generated - do not modify! if you need to make a change, please see the Java Classes in the activemq-openwire-generator module

6.315.2 Constructor & Destructor Documentation

6.315.2.1 `activemq::wireformat::openwire::marshal::generated::LocalTransactionIdMarshaller::LocalTransactionIdMarshaller ()`
`[inline]`

6.315.2.2 `virtual activemq::wireformat::openwire::marshal::generated::LocalTransactionIdMarshaller::~~LocalTransactionIdMarshaller ()`
`[inline, virtual]`

6.315.3 Member Function Documentation

6.315.3.1 `virtual commands::DataStructure* activemq::wireformat::openwire::marshal::generated::LocalTransactionIdMarshaller::createObject () const` `[virtual]`

Creates a new instance of the class that this class is a marshaling director for.

6.315 activemq::wireformat::openwire::marshal::generated::LocalTransactionId-Marshaller Class

Reference

1685

Returns

newly allocated Command

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1120).

6.315.3.2 `virtual unsigned char activemq::wireformat::openwire::marshal::generated::LocalTransactionIdMarshaller::getDataType () const`
[virtual]

Gets the DataType that this class marshals/unmarshals.

Returns

byte Id of this classes DataType

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1122).

6.315.3.3 `virtual void activemq::wireformat::openwire::marshal::generated::LocalTransactionIdMarshaller::looseMarshal (OpenWireFormat * format, commands::DataStructure * command, decaf::io::DataOutputStream * ds)` [virtual]

Tight Marshal to the given stream.

Parameters

<i>format</i>	- The OpenWireFormat properties
<i>command</i>	- the object to Marshal
<i>ds</i>	- DataOutputStream to marshal to

Exceptions

<i>IOException</i>	if an error occurs.
--------------------	---------------------

Reimplemented from **activemq::wireformat::openwire::marshal::generated::LocalTransactionIdMarshaller** (p. 2771).

6.315.3.4 `virtual void activemq::wireformat::openwire::marshal::generated::LocalTransactionIdMarshaller::looseUnmarshal (OpenWireFormat * format, commands::DataStructure * command, decaf::io::DataInputStream * dis)`
[virtual]

Loose Un-marshal to the given stream.

Parameters

<i>format</i>	- The OpenwireFormat properties
<i>command</i>	- the object to Un-Marshal
<i>dis</i>	- the DataInputStream to Un-Marshal from

Exceptions

<i>IOException</i>	if an error occurs.
--------------------	---------------------

Reimplemented from **activemq::wireformat::openwire::marshal::generated::Local-TransactionIdMarshaller** (p. 2772).

6.315.3.5 `virtual int activemq::wireformat::openwire::marshal::generated::Local-TransactionIdMarshaller::tightMarshal1 (OpenWireFormat * format, commands::DataStructure * command, utils::BooleanStream * bs)`
[virtual]

Tight Marshal to the given stream.

Parameters

<i>format</i>	The OpenwireFormat properties
<i>command</i>	The object to Marshal
<i>bs</i>	The boolean stream to marshal to.

Exceptions

<i>IOException</i>	if an error occurs.
--------------------	---------------------

Reimplemented from **activemq::wireformat::openwire::marshal::generated::Local-TransactionIdMarshaller** (p. 2772).

6.315.3.6 `virtual void activemq::wireformat::openwire::marshal::generated::Local-TransactionIdMarshaller::tightMarshal2 (OpenWireFormat * format, commands::DataStructure * command, decaf::io::DataOutputStream * ds, utils::BooleanStream * bs)` [virtual]

Tight Marshal to the given stream.

Parameters

<i>format</i>	- The OpenwireFormat properties
<i>command</i>	- the object to Marshal
<i>ds</i>	- the DataOutputStream to Marshal to
<i>bs</i>	- boolean stream to marshal to.

Exceptions

<i>IOException</i>	if an error occurs.
--------------------	---------------------

Reimplemented from **activemq::wireformat::openwire::marshal::generated::TransactionIdMarshaller** (p. 2773).

6.315.3.7 virtual void **activemq::wireformat::openwire::marshal::generated::LocalTransactionIdMarshaller::tightUnmarshal** (**OpenWireFormat** * *format*, **commands::DataStructure** * *command*, **decaf::io::DataInputStream** * *dis*, **utils::BooleanStream** * *bs*) [virtual]

Tight Un-marhsal to the given stream.

Parameters

<i>format</i>	- The OpenWireFormat properties
<i>command</i>	- the object to Un-Marshal
<i>dis</i>	- the DataInputStream to Un-Marshal from
<i>bs</i>	- boolean stream to unmarshal from.

Exceptions

<i>IOException</i>	if an error occurs.
--------------------	---------------------

Reimplemented from **activemq::wireformat::openwire::marshal::generated::TransactionIdMarshaller** (p. 2773).

The documentation for this class was generated from the following file:

- src/main/activemq/wireformat/openwire/marshal/generated/**LocalTransactionIdMarshaller.h**

6.316 decaf::util::concurrent::Lock Class Reference

A wrapper class around a given synchronization mechanism that provides automatic release upon destruction.

```
#include <src/main/decaf/util/concurrent/Lock.h>
```

Public Member Functions

- **Lock** (**Synchronizable** *object, const bool initiallyLocked=true)
Constructor - initializes the object member and locks the object if desired.
- virtual ~**Lock** ()
Destructor - Unlocks the object if it is locked.
- void **lock** ()

Locks the object.

- void **unlock** ()

Unlocks the object if it is already locked, otherwise a call to this method has no effect.

- bool **isLocked** () const

Indicates whether or not the object is locked.

6.316.1 Detailed Description

A wrapper class around a given synchronization mechanism that provides automatic release upon destruction.

Since

1.0

6.316.2 Constructor & Destructor Documentation

6.316.2.1 `decaf::util::concurrent::Lock (Synchronizable * object, const bool initiallyLocked = true)`

Constructor - initializes the object member and locks the object if desired.

Parameters

<i>object</i>	The sync object to control
<i>initially-Locked</i>	If true, the object will automatically be locked.

6.316.2.2 `virtual decaf::util::concurrent::Lock::~~Lock () [virtual]`

Destructor - Unlocks the object if it is locked.

6.316.3 Member Function Documentation

6.316.3.1 `bool decaf::util::concurrent::Lock::isLocked () const [inline]`

Indicates whether or not the object is locked.

Returns

true if the object is locked, otherwise false.

6.316.3.2 `void decaf::util::concurrent::Lock::lock ()`

Locks the object.

6.316.3.3 void decaf::util::concurrent::Lock::unlock ()

Unlocks the object if it is already locked, otherwise a call to this method has no effect.

The documentation for this class was generated from the following file:

- src/main/decaf/util/concurrent/**Lock.h**

6.317 decaf::util::concurrent::locks::Lock Class Reference

Lock (p. 1684) implementations provide more extensive locking operations than can be obtained using synchronized statements.

```
#include <src/main/decaf/util/concurrent/locks/Lock.h>
```

Inheritance diagram for decaf::util::concurrent::locks::Lock:

Public Member Functions

- virtual \sim **Lock** ()
- virtual void **lock** ()=0
Acquires the lock.
- virtual void **lockInterruptibly** ()=0
Acquires the lock unless the current thread is interrupted.
- virtual bool **tryLock** ()=0
Acquires the lock only if it is free at the time of invocation.
- virtual bool **tryLock** (long long time, const **TimeUnit** &unit)=0
Acquires the lock if it is free within the given waiting time and the current thread has not been interrupted.
- virtual void **unlock** ()=0
Releases the lock.
- virtual **Condition** * **newCondition** ()=0
*Returns a new **Condition** (p. 921) instance that is bound to this **Lock** (p. 1684) instance.*

6.317.1 Detailed Description

Lock (p. 1684) implementations provide more extensive locking operations than can be obtained using synchronized statements.

They allow more flexible structuring, may have quite different properties, and may support multiple associated **Condition** (p. 921) objects.

A lock is a tool for controlling access to a shared resource by multiple threads. - Commonly, a lock provides exclusive access to a shared resource: only one thread

at a time can acquire the lock and all access to the shared resource requires that the lock be acquired first. However, some locks may allow concurrent access to a shared resource, such as the read lock of a **ReadWriteLock** (p. 2246).

While the scoping mechanism for synchronized statements makes it much easier to program with monitor locks, and helps avoid many common programming errors involving locks, there are occasions where you need to work with locks in a more flexible way. For example, some algorithms for traversing concurrently accessed data structures require the use of "hand-over-hand" or "chain locking": you acquire the lock of node A, then node B, then release A and acquire C, then release B and acquire D and so on. Implementations of the **Lock** (p. 1684) interface enable the use of such techniques by allowing a lock to be acquired and released in different scopes, and allowing multiple locks to be acquired and released in any order.

With this increased flexibility comes additional responsibility. The absence of block-structured locking removes the automatic release of locks that occurs with synchronized statements. In most cases, the following idiom should be used:

```
Lock (p. 1684) l = ...; l.lock(); try { // access the resource protected by this lock } catch(...) { l.unlock(); }
```

When locking and unlocking occur in different scopes, care must be taken to ensure that all code that is executed while the lock is held is protected by try-catch ensure that the lock is released when necessary.

Lock (p. 1684) implementations provide additional functionality over the use of synchronized methods and statements by providing a non-blocking attempt to acquire a lock (**tryLock()** (p. 1687)), an attempt to acquire the lock that can be interrupted (**lockInterruptibly()** (p. 1686)), and an attempt to acquire the lock that can timeout (**tryLock(long, TimeUnit)**).

Note that **Lock** (p. 1684) instances are just normal objects and can themselves be used as the target in a synchronized statement.

The three forms of lock acquisition (interruptible, non-interruptible, and timed) may differ in their performance characteristics, ordering guarantees, or other implementation qualities. Further, the ability to interrupt the ongoing acquisition of a lock may not be available in a given **Lock** (p. 1684) class. Consequently, an implementation is not required to define exactly the same guarantees or semantics for all three forms of lock acquisition, nor is it required to support interruption of an ongoing lock acquisition. An implementation is required to clearly document the semantics and guarantees provided by each of the locking methods. It must also obey the interruption semantics as defined in this interface, to the extent that interruption of lock acquisition is supported: which is either totally, or only on method entry.

As interruption generally implies cancellation, and checks for interruption are often infrequent, an implementation can favor responding to an interrupt over normal method return. This is true even if it can be shown that the interrupt occurred after another action may have unblocked the thread. An implementation should document this behavior.

Since

1.0

6.317.2 Constructor & Destructor Documentation

6.317.2.1 virtual **decaf::util::concurrent::locks::Lock::~~Lock** () [inline, virtual]

6.317.3 Member Function Documentation

6.317.3.1 virtual void **decaf::util::concurrent::locks::Lock::lock** () [pure virtual]

Acquires the lock.

If the lock is not available then the current thread becomes disabled for thread scheduling purposes and lies dormant until the lock has been acquired.

Implementation Considerations

A **Lock** (p. 1684) implementation may be able to detect erroneous use of the lock, such as an invocation that would cause deadlock, and may throw an exception in such circumstances. The circumstances and the exception type must be documented by that **Lock** (p. 1684) implementation.

Exceptions

<i>RuntimeException</i>	if an error occurs while acquiring the lock.
-------------------------	--

Implemented in **decaf::util::concurrent::locks::ReentrantLock** (p. 2259).

6.317.3.2 virtual void **decaf::util::concurrent::locks::Lock::lockInterruptibly** () [pure virtual]

Acquires the lock unless the current thread is interrupted.

Acquires the lock if it is available and returns immediately.

If the lock is not available then the current thread becomes disabled for thread scheduling purposes and lies dormant until one of two things happens:

* The lock is acquired by the current thread; or * Some other thread interrupts the current thread, and interruption of lock acquisition is supported.

If the current thread:

* has its interrupted status set on entry to this method; or * is interrupted while acquiring the lock, and interruption of lock acquisition is supported,

then InterruptedException is thrown and the current thread's interrupted status is cleared.

Implementation Considerations

The ability to interrupt a lock acquisition in some implementations may not be possible, and if possible may be an expensive operation. The programmer should be aware that this may be the case. An implementation should document when this is the case.

An implementation can favor responding to an interrupt over normal method return.

A **Lock** (p. 1684) implementation may be able to detect erroneous use of the lock, such as an invocation that would cause deadlock, and may throw an exception in such circumstances. The circumstances and the exception type must be documented by that **Lock** (p. 1684) implementation.

Exceptions

<i>RuntimeException</i>	if an error occurs while acquiring the lock.
<i>InterruptedException</i>	if the current thread is interrupted while acquiring the lock (and interruption of lock acquisition is supported).

Implemented in **decaf::util::concurrent::locks::ReentrantLock** (p. 2260).

6.317.3.3 virtual Condition* decaf::util::concurrent::locks::Lock::newCondition () [pure virtual]

Returns a new **Condition** (p. 921) instance that is bound to this **Lock** (p. 1684) instance.

Before waiting on the condition the lock must be held by the current thread. A call to **Condition.await()** (p. 923) will atomically release the lock before waiting and re-acquire the lock before the wait returns.

Implementation Considerations

The exact operation of the **Condition** (p. 921) instance depends on the **Lock** (p. 1684) implementation and must be documented by that implementation.

Returns

A new **Condition** (p. 921) instance for this **Lock** (p. 1684) instance the caller must delete the returned **Condition** (p. 921) object when done with it.

Exceptions

<i>RuntimeException</i>	if an error occurs while creating the Condition (p. 921).
<i>UnsupportedOperationException</i>	if this Lock (p. 1684) implementation does not support conditions

Implemented in **decaf::util::concurrent::locks::ReentrantLock** (p. 2260).

6.317.3.4 virtual bool decaf::util::concurrent::locks::Lock::tryLock () [pure virtual]

Acquires the lock only if it is free at the time of invocation.

Acquires the lock if it is available and returns immediately with the value true. If the lock is not available then this method will return immediately with the value false.

A typical usage idiom for this method would be:

Lock (p. 1684) `lock = ...; if (lock.tryLock()) { try { // manipulate protected state } catch(...) { lock.unlock(); } } else { // perform alternative actions }`

This usage ensures that the lock is unlocked if it was acquired, and doesn't try to unlock if the lock was not acquired.

Returns

true if the lock was acquired and false otherwise

Exceptions

<i>RuntimeException</i>	if an error occurs while acquiring the lock.
-------------------------	--

Implemented in **decaf::util::concurrent::locks::ReentrantLock** (p. 2261).

6.317.3.5 `virtual bool decaf::util::concurrent::locks::Lock::tryLock (long long time, const TimeUnit & unit) [pure virtual]`

Acquires the lock if it is free within the given waiting time and the current thread has not been interrupted.

If the lock is available this method returns immediately with the value true. If the lock is not available then the current thread becomes disabled for thread scheduling purposes and lies dormant until one of three things happens:

* The lock is acquired by the current thread; or * Some other thread interrupts the current thread, and interruption of lock acquisition is supported; or * The specified waiting time elapses

If the lock is acquired then the value true is returned.

If the current thread:

* has its interrupted status set on entry to this method; or * is interrupted while acquiring the lock, and interruption of lock acquisition is supported,

then InterruptedException is thrown and the current thread's interrupted status is cleared.

If the specified waiting time elapses then the value false is returned. If the time is less than or equal to zero, the method will not wait at all.

Implementation Considerations

The ability to interrupt a lock acquisition in some implementations may not be possible, and if possible may be an expensive operation. The programmer should be aware that this may be the case. An implementation should document when this is the case.

An implementation can favor responding to an interrupt over normal method return, or reporting a timeout.

A **Lock** (p. 1684) implementation may be able to detect erroneous use of the lock, such as an invocation that would cause deadlock, and may throw an (unchecked) exception in such circumstances. The circumstances and the exception type must be documented

by that **Lock** (p. 1684) implementation.

Parameters

<i>time</i>	the maximum time to wait for the lock
<i>unit</i>	the time unit of the time argument

Returns

true if the lock was acquired and false if the waiting time elapsed before the lock was acquired

Exceptions

<i>RuntimeException</i>	if an error occurs while acquiring the lock.
<i>InterruptedException</i>	if the current thread is interrupted while acquiring the lock (and interruption of lock acquisition is supported)

Implemented in **decaf::util::concurrent::locks::ReentrantLock** (p. 2262).

6.317.3.6 `virtual void decaf::util::concurrent::locks::Lock::unlock ()` [pure virtual]

Releases the lock.

Implementation Considerations

A **Lock** (p. 1684) implementation will usually impose restrictions on which thread can release a lock (typically only the holder of the lock can release it) and may throw an exception if the restriction is violated. Any restrictions and the exception type must be documented by that **Lock** (p. 1684) implementation.

Exceptions

<i>RuntimeException</i>	if an error occurs while acquiring the lock.
<i>IllegalMonitorStateException</i>	if the current thread is not the owner of the lock.

Implemented in **decaf::util::concurrent::locks::ReentrantLock** (p. 2263).

The documentation for this class was generated from the following file:

- src/main/decaf/util/concurrent/locks/**Lock.h**

6.318 decaf::util::concurrent::locks::LockSupport Class Reference

Basic thread blocking primitives for creating locks and other synchronization classes.

```
#include <src/main/decaf/util/concurrent/locks/LockSupport.h>
```

Public Member Functions

- `~LockSupport ()`

Static Public Member Functions

- static void **unpark** (**decaf::lang::Thread** *thread) throw ()
Makes available the permit for the given thread, if it was not already available.
- static void **park** () throw ()
Disables the current thread for thread scheduling purposes unless the permit is available.
- static void **parkNanos** (long long nanos) throw ()
Disables the current thread for thread scheduling purposes, for up to the specified waiting time, unless the permit is available.
- static void **parkUntil** (long long deadline) throw ()
Disables the current thread for thread scheduling purposes, until the specified deadline, unless the permit is available.

6.318.1 Detailed Description

Basic thread blocking primitives for creating locks and other synchronization classes.

This class associates, with each thread that uses it, a permit (in the sense of the **Semaphore** (p. 2331) class). A call to **park** will return immediately if the permit is available, consuming it in the process; otherwise it may block. A call to **unpark** makes the permit available, if it was not already available. (Unlike with Semaphores though, permits do not accumulate. There is at most one.)

Methods **park** and **unpark** provide efficient means of blocking and unblocking threads. Races between one thread invoking **park** and another thread trying to **unpark** it will preserve liveness, due to the permit. Additionally, **park** will return if the caller's thread was interrupted, and timeout versions are supported. The **park** method may also return at any other time, for "no reason", so in general must be invoked within a loop that rechecks conditions upon return. In this sense **park** serves as an optimization of a "busy wait" that does not waste as much time spinning, but must be paired with an **unpark** to be effective.

These methods are designed to be used as tools for creating higher-level synchronization utilities, and are not in themselves useful for most concurrency control applications. The **park** method is designed for use only in constructions of the form:

```
while (!canProceed()) { ... LockSupport.park(this); }
```

where neither **canProceed** nor any other actions prior to the call to **park** entail locking or blocking. Because only one permit is associated with each thread, any intermediary uses of **park** could interfere with its intended effects.

Sample Usage. Here is a sketch of a first-in-first-out non-reentrant lock class:

```
class FIFOMutex { private:
AtomicBoolean locked; ConcurrentLinkedQueue<Thread*> waiters;
public:
void lock() {
bool wasInterrupted = false; Thread* current = Thread::currentThread(); waiters.add(
current );
// Block while not first in queue or cannot acquire lock while( waiters.peek() != current ||
!locked.compareAndSet( false, true ) ) {
LockSupport.park(this); if( Thread::interrupted() ) // ignore interrupts while waiting was-
Interrupted = true; }
waiters.remove(); if( wasInterrupted ) // reassert interrupt status on exit current.-
interrupt(); }
void unlock() { locked.set( false ); LockSupport.unpark (p. 1693)( waiters.peek() ); } }
```

Since

1.0

6.318.2 Constructor & Destructor Documentation

6.318.2.1 `decaf::util::concurrent::locks::LockSupport::~~LockSupport ()`

6.318.3 Member Function Documentation

6.318.3.1 `static void decaf::util::concurrent::locks::LockSupport::park () throw ()`
`[static]`

Disables the current thread for thread scheduling purposes unless the permit is available.

If the permit is available then it is consumed and the call returns immediately; otherwise the current thread becomes disabled for thread scheduling purposes and lies dormant until one of three things happens:

- * Some other thread invokes unpark with the current thread as the target; or
- * Some other thread interrupts the current thread; or
- * The call spuriously (that is, for no reason) returns.

This method does not report which of these caused the method to return. Callers should re-check the conditions which caused the thread to park in the first place. Callers may also determine, for example, the interrupt status of the thread upon return.

6.318.3.2 `static void decaf::util::concurrent::locks::LockSupport::parkNanos (long long nanos) throw () [static]`

Disables the current thread for thread scheduling purposes, for up to the specified waiting time, unless the permit is available.

If the permit is available then it is consumed and the call returns immediately; otherwise the current thread becomes disabled for thread scheduling purposes and lies dormant until one of four things happens:

- * Some other thread invokes `unpark` with the current thread as the target; or
- * Some other thread interrupts the current thread; or
- * The specified waiting time elapses; or
- * The call spuriously (that is, for no reason) returns.

This method does not report which of these caused the method to return. Callers should re-check the conditions which caused the thread to park in the first place. Callers may also determine, for example, the interrupt status of the thread, or the elapsed time upon return.

Parameters

<i>nanos</i>	the maximum number of nanoseconds to wait
--------------	---

6.318.3.3 `static void decaf::util::concurrent::locks::LockSupport::parkUntil (long long deadline) throw () [static]`

Disables the current thread for thread scheduling purposes, until the specified deadline, unless the permit is available.

If the permit is available then it is consumed and the call returns immediately; otherwise the current thread becomes disabled for thread scheduling purposes and lies dormant until one of four things happens:

- * Some other thread invokes `unpark` with the current thread as the target; or
- * Some other thread interrupts the current thread; or
- * The specified deadline passes; or
- * The call spuriously (that is, for no reason) returns.

This method does not report which of these caused the method to return. Callers should re-check the conditions which caused the thread to park in the first place. Callers may also determine, for example, the interrupt status of the thread, or the current time upon return.

Parameters

<i>deadline</i>	the absolute time, in milliseconds from the Epoch, to wait until
-----------------	--

6.318.3.4 `static void decaf::util::concurrent::locks::LockSupport::unpark (decaf::lang::Thread * thread) throw () [static]`

Makes available the permit for the given thread, if it was not already available.

If the thread was blocked on park then it will unblock. Otherwise, its next call to park is guaranteed not to block. This operation is not guaranteed to have any effect at all if the given thread has not been started.

Parameters

<i>thread</i>	the thread to unport, or NULL in which case the method has no effect.
---------------	---

The documentation for this class was generated from the following file:

- src/main/decaf/util/concurrent/locks/**LockSupport.h**

6.319 decaf::util::logging::Logger Class Reference

A **Logger** (p. 1693) object is used to log messages for a specific system or application component.

```
#include <src/main/decaf/util/logging/Logger.h>
```

Public Member Functions

- virtual **~Logger** ()
- const std::string & **getName** () const
*Gets the name of this **Logger** (p. 1693).*
- **Logger** * **getParent** () const
*Gets the parent of this **Logger** (p. 1693) which will be the nearest existing **Logger** (p. 1693) in this Loggers namespace.*
- void **setParent** (**Logger** *parent)
*Set (p. 2397) the parent for this **Logger** (p. 1693).*
- void **addHandler** (**Handler** *handler)
*Add a log **Handler** (p. 1401) to receive logging messages.*
- void **removeHandler** (**Handler** *handler)
*Removes the specified **Handler** (p. 1401) from this logger, ownership of the **Handler** (p. 1401) pointer is returned to the caller.*
- const std::list< **Handler** * > & **getHandlers** () const
Gets a vector containing all the handlers that this class has been assigned to use.
- void **setFilter** (**Filter** *filter)
*Set (p. 2397) a filter to control output on this **Logger** (p. 1693).*
- const **Filter** * **getFilter** () const
*Gets the **Filter** (p. 1333) object that this class is using.*
- **Level** **getLevel** () const
*Get the log **Level** (p. 1616) that has been specified for this **Logger** (p. 1693).*
- void **setLevel** (const **Level** &level)
Set (p. 2397) the log level specifying which message levels will be logged by this logger.

- bool **getUseParentHandlers** () const
Discover whether or not this logger is sending its output to its parent logger.
- void **setUseParentHandlers** (bool value)
*Specify whether or not this logger should send its output to its parent **Logger** (p. 1693).*
- virtual void **entering** (const std::string &blockName, const std::string &file, const int line)
Logs an Block Enter message.
- virtual void **exiting** (const std::string &blockName, const std::string &file, const int line)
Logs an Block Exit message.
- virtual void **severe** (const std::string &file, const int line, const std::string function-Name, const std::string &message)
*Log a SEVERE **Level** (p. 1616) Log.*
- virtual void **warning** (const std::string &file, const int line, const std::string functionName, const std::string &message)
*Log a WARN **Level** (p. 1616) Log.*
- virtual void **info** (const std::string &file, const int line, const std::string function-Name, const std::string &message)
*Log a INFO **Level** (p. 1616) Log.*
- virtual void **debug** (const std::string &file, const int line, const std::string function-Name, const std::string &message)
*Log a DEBUG **Level** (p. 1616) Log.*
- virtual void **config** (const std::string &file, const int line, const std::string function-Name, const std::string &message)
*Log a CONFIG **Level** (p. 1616) Log.*
- virtual void **fine** (const std::string &file, const int line, const std::string function-Name, const std::string &message)
*Log a FINE **Level** (p. 1616) Log.*
- virtual void **finer** (const std::string &file, const int line, const std::string function-Name, const std::string &message)
*Log a FINER **Level** (p. 1616) Log.*
- virtual void **finest** (const std::string &file, const int line, const std::string function-Name, const std::string &message)
*Log a FINEST **Level** (p. 1616) Log.*
- virtual void **throwing** (const std::string &file, const int line, const std::string functionName, const decaf::lang::Throwable &thrown)
Log throwing an exception.
- virtual bool **isLoggable** (const **Level** &level) const
Check if a message of the given level would actually be logged by this logger.
- virtual void **log** (**LogRecord** &record)
*Log a **LogRecord** (p. 1719).*
- virtual void **log** (const **Level** &level, const std::string &message)
Log a message, with no arguments.

- virtual void **log** (const **Level** &levels, const std::string &file, const int line, const std::string &message,...)

Log a message, with the list of params that is formatted into the message string.

- virtual void **log** (const **Level** &level, const std::string &file, const int line, const std::string &message, **lang::Exception** &ex)

Log a message, with associated Throwable information.

Static Public Member Functions

- static **Logger** * **getAnonymousLogger** ()

Creates an anonymous logger.

- static **Logger** * **getLogger** (const std::string &name)

Find or create a logger for a named subsystem.

Protected Member Functions

- **Logger** (const std::string &name)

*Creates a new instance of the **Logger** (p. 1693) with the given name.*

6.319.1 Detailed Description

A **Logger** (p. 1693) object is used to log messages for a specific system or application component.

Loggers are normally named, using a hierarchical dot-separated namespace. **Logger** (p. 1693) names can be arbitrary strings, but they should normally be based on the namespace or class name of the logged component, such as **decaf.net** (p. 92) or org.-apache.decaf. In addition it is possible to create "anonymous" Loggers that are not stored in the **Logger** (p. 1693) namespace.

Logger (p. 1693) objects may be obtained by calls on one of the getLogger factory methods. These will either create a new **Logger** (p. 1693) or return a suitable existing **Logger** (p. 1693).

Logging messages will be forwarded to registered **Handler** (p. 1401) objects, which can forward the messages to a variety of destinations, including consoles, files, OS logs, etc.

Each **Logger** (p. 1693) keeps track of a "parent" **Logger** (p. 1693), which is its nearest existing ancestor in the **Logger** (p. 1693) namespace.

Each **Logger** (p. 1693) has a "Level" associated with it. This reflects a minimum **Level** (p. 1616) that this logger cares about. If a **Logger** (p. 1693)'s level is set to **Level::INHERIT** (p. 1620), then its effective level is inherited from its parent, which may in turn obtain it recursively from its parent, and so on up the tree.

The log level can be configured based on the properties from the logging configuration file, as described in the description of the **LogManager** (p. 1712) class. However it may also be dynamically changed by calls on the **Logger.setLevel** (p. 1704) method.

If a logger's level is changed the change may also affect child loggers, since any child logger that has 'inherit' as its level will inherit its effective level from its parent.

On each logging call the **Logger** (p. 1693) initially performs a cheap check of the request level (e.g. SEVERE or FINE) against the effective log level of the logger. If the request level is lower than the log level, the logging call returns immediately.

After passing this initial (cheap) test, the **Logger** (p. 1693) will allocate a **LogRecord** (p. 1719) to describe the logging message. It will then call a **Filter** (p. 1333) (if present) to do a more detailed check on whether the record should be published. If that passes it will then publish the **LogRecord** (p. 1719) to its output Handlers. By default, loggers also publish to their parent's Handlers, recursively up the tree.

Formatting is the responsibility of the output **Handler** (p. 1401), which will typically call a **Formatter** (p. 1387).

Note that formatting need not occur synchronously. It may be delayed until a **LogRecord** (p. 1719) is actually written to an external sink.

All methods on **Logger** (p. 1693) are thread safe.

Since

1.0

6.319.2 Constructor & Destructor Documentation

6.319.2.1 decaf::util::logging::Logger::Logger (const std::string & *name*) [protected]

Creates a new instance of the **Logger** (p. 1693) with the given name.

The logger will be initially configured with a null **Level** (p. 1616) and with useParent-Handlers true.

Parameters

<i>name</i>	A name for the logger. This should be a dot-separated name and should normally be based on the package name or class name of the subsystem, such as decaf.net (p. 92) or org.apache.decaf. It may be empty for anonymous Loggers.
-------------	--

6.319.2.2 virtual decaf::util::logging::Logger::~~Logger () [virtual]

6.319.3 Member Function Documentation

6.319.3.1 void decaf::util::logging::Logger::addHandler (Handler * *handler*)

Add a log **Handler** (p. 1401) to receive logging messages.

By default, Loggers also send their output to their parent logger. Typically the root - **Logger** (p. 1693) is configured with a set of Handlers that essentially act as default handlers for all loggers.

The ownership of the given **Handler** (p. 1401) is passed to the **Logger** (p. 1693) and the **Handler** (p. 1401) will be deleted when this **Logger** (p. 1693) is destroyed unless the caller first calls `removeHandler` with the same pointer value as was originally given.

Parameters

<i>handler</i>	A Logging Handler (p. 1401)
----------------	------------------------------------

Exceptions

<i>NullPointerException</i>	if the Handler (p. 1401) given is NULL.
-----------------------------	--

6.319.3.2 `virtual void decaf::util::logging::Logger::config (const std::string & file,
const int line, const std::string functionName, const std::string & message)
[virtual]`

Log a CONFIG **Level** (p. 1616) Log.

If the logger is currently enabled for the CONFIG message level then the given message is forwarded to all the registered output **Handler** (p. 1401) objects.

Parameters

<i>file</i>	The file name where the log was generated.
<i>line</i>	The line number where the log was generated.
<i>function-Name</i>	The name of the function that logged this.
<i>message</i>	The message to log at this Level (p. 1616).

6.319.3.3 `virtual void decaf::util::logging::Logger::debug (const std::string & file,
const int line, const std::string functionName, const std::string & message)
[virtual]`

Log a DEBUG **Level** (p. 1616) Log.

If the logger is currently enabled for the DEBUG message level then the given message is forwarded to all the registered output **Handler** (p. 1401) objects.

Parameters

<i>file</i>	The file name where the log was generated.
<i>line</i>	The line number where the log was generated.
<i>function-Name</i>	The name of the function that logged this.
<i>message</i>	The message to log at this Level (p. 1616).

6.319.3.4 virtual void **decaf::util::logging::Logger::entering** (const std::string & *blockName*, const std::string & *file*, const int *line*) [virtual]

Logs an Block Enter message.

This is a convenience method that is used to tag a block enter, a log record with the given information is logged at the **Level::FINER** (p. 1620) log level.

Parameters

<i>blockName</i>	The source block name, (usually <code>ClassName::MethodName</code> , or <code>-MethodName</code>).
<i>file</i>	The source file name where this method was called from.
<i>line</i>	The source line number where this method was called from.

6.319.3.5 virtual void **decaf::util::logging::Logger::exiting** (const std::string & *blockName*, const std::string & *file*, const int *line*) [virtual]

Logs an Block Exit message.

This is a convenience method that is used to tag a block enter, a log record with the given information is logged at the **Level::FINER** (p. 1620) log level.

Parameters

<i>blockName</i>	The source block name, (usually <code>ClassName::MethodName</code> , or <code>-MethodName</code>).
<i>file</i>	The source file name where this method was called from.
<i>line</i>	The source line number where this method was called from.

6.319.3.6 virtual void **decaf::util::logging::Logger::fine** (const std::string & *file*, const int *line*, const std::string *functionName*, const std::string & *message*) [virtual]

Log a FINE **Level** (p. 1616) Log.

If the logger is currently enabled for the FINE message level then the given message is forwarded to all the registered output **Handler** (p. 1401) objects.

Parameters

<i>file</i>	The file name where the log was generated.
<i>line</i>	The line number where the log was generated.
<i>function-Name</i>	The name of the function that logged this.
<i>message</i>	The message to log at this Level (p. 1616).

6.319.3.7 `virtual void decaf::util::logging::Logger::finer (const std::string & file,
const int line, const std::string functionName, const std::string & message)
[virtual]`

Log a FINER **Level** (p. 1616) Log.

If the logger is currently enabled for the FINER message level then the given message is forwarded to all the registered output **Handler** (p. 1401) objects.

Parameters

<i>file</i>	The file name where the log was generated.
<i>line</i>	The line number where the log was generated.
<i>function-Name</i>	The name of the function that logged this.
<i>message</i>	The message to log at this Level (p. 1616).

6.319.3.8 `virtual void decaf::util::logging::Logger::finest (const std::string & file,
const int line, const std::string functionName, const std::string & message)
[virtual]`

Log a FINEST **Level** (p. 1616) Log.

If the logger is currently enabled for the FINEST message level then the given message is forwarded to all the registered output **Handler** (p. 1401) objects.

Parameters

<i>file</i>	The file name where the log was generated.
<i>line</i>	The line number where the log was generated.
<i>function-Name</i>	The name of the function that logged this.
<i>message</i>	The message to log at this Level (p. 1616).

6.319.3.9 `static Logger* decaf::util::logging::Logger::getAnonymousLogger ()
[static]`

Creates an anonymous logger.

The newly created **Logger** (p. 1693) is not registered in the **LogManager** (p. 1712) namespace. There will be no access checks on updates to the logger. Even although the new logger is anonymous, it is configured to have the root logger ("") as its parent. This means that by default it inherits its effective level and handlers from the root logger.

The caller is responsible for destroying the returned logger.

Returns

Newly created anonymous logger

6.319.3.10 `const Filter* decaf::util::logging::Logger::getFilter () const`
[inline]

Gets the **Filter** (p. 1333) object that this class is using.

Returns

the **Filter** (p. 1333) in use, (can be NULL).

6.319.3.11 `const std::list<Handler*>& decaf::util::logging::Logger::getHandlers ()`
const

Gets a vector containing all the handlers that this class has been assigned to use.

Returns

a list of handlers that are used by this logger

6.319.3.12 `Level decaf::util::logging::Logger::getLevel () const` [inline]

Get the log **Level** (p. 1616) that has been specified for this **Logger** (p. 1693).

The result may be the INHERIT level, which means that this logger's effective level will be inherited from its parent.

Returns

the level that is currently set

6.319.3.13 `static Logger* decaf::util::logging::Logger::getLogger (const std::string &`
`name)` [static]

Find or create a logger for a named subsystem.

If a logger has already been created with the given name it is returned. Otherwise a new logger is created.

If a new logger is created its log level will be configured based on the **LogManager** (p. 1712) and it will be configured to also send logging output to its parent loggers Handlers. It will be registered in the **LogManager** (p. 1712) global namespace.

Parameters

<i>name</i>	- A name for the logger. This should be a dot-separated name and should normally be based on the package name or class name of the subsystem, such as cms or activemq.core.ActiveMQConnection (p. 187)
-------------	---

Returns

a suitable logger.

6.319.3.14 `const std::string& decaf::util::logging::Logger::getName () const`
`[inline]`

Gets the name of this **Logger** (p. 1693).

Returns

logger name

6.319.3.15 `Logger* decaf::util::logging::Logger::getParent () const` `[inline]`

Gets the parent of this **Logger** (p. 1693) which will be the nearest existing **Logger** (p. 1693) in this Loggers namespace.

If this is the Root **Logger** (p. 1693) than this method returns NULL.

Returns

Pointer to this Loggers nearest parent **Logger** (p. 1693).

6.319.3.16 `bool decaf::util::logging::Logger::getUseParentHandlers () const`
`[inline]`

Discover whether or not this logger is sending its output to its parent logger.

Returns

true if using Parent Handlers

6.319.3.17 `virtual void decaf::util::logging::Logger::info (const std::string & file,`
`const int line, const std::string functionName, const std::string & message)`
`[virtual]`

Log a INFO **Level** (p. 1616) Log.

If the logger is currently enabled for the INFO message level then the given message is forwarded to all the registered output **Handler** (p. 1401) objects.

Parameters

<i>file</i>	The file name where the log was generated.
<i>line</i>	The line number where the log was generated.
<i>function-Name</i>	The name of the function that logged this.
<i>message</i>	The message to log at this Level (p. 1616).

6.319.3.18 virtual bool decaf::util::logging::Logger::isLoggable (const Level & *level*)
const [virtual]

Check if a message of the given level would actually be logged by this logger.

This check is based on the Loggers effective level, which may be inherited from its parent.

Parameters

<i>level</i>	- a message logging level
--------------	---------------------------

Returns

true if the given message level is currently being logged.

6.319.3.19 virtual void decaf::util::logging::Logger::log (LogRecord & *record*)
[virtual]

Log a **LogRecord** (p. 1719).

All the other logging methods in this class call through this method to actually perform any logging. Subclasses can override this single method to capture all log activity.

Parameters

<i>record</i>	- the LogRecord (p. 1719) to be published
---------------	--

6.319.3.20 virtual void decaf::util::logging::Logger::log (const Level & *level*, const std::string & *message*) [virtual]

Log a message, with no arguments.

If the logger is currently enabled for the given message level then the given message is forwarded to all the registered output **Handler** (p. 1401) objects

Parameters

<i>level</i>	the Level (p. 1616) to log at
<i>message</i>	the message to log

6.319.3.21 virtual void decaf::util::logging::Logger::log (const Level & *levels*, const std::string & *file*, const int *line*, const std::string & *message*, ...) [virtual]

Log a message, with the list of params that is formatted into the message string.

If the logger is currently enabled for the given message level then the given message is forwarded to all the registered output **Handler** (p. 1401) objects

Parameters

<i>level</i>	the Level (p. 1616) to log at
<i>file</i>	the message to log
<i>line</i>	the line in the file
...	variable length argument to format the message string.

6.319.3.22 `virtual void decaf::util::logging::Logger::log (const Level & level, const std::string & file, const int line, const std::string & message, lang::Exception & ex) [virtual]`

Log a message, with associated Throwable information.

If the logger is currently enabled for the given message level then the given arguments are stored in a **LogRecord** (p. 1719) which is forwarded to all registered output handlers. Note that the thrown argument is stored in the **LogRecord** (p. 1719) thrown property, rather than the **LogRecord** (p. 1719) parameters property. Thus is it processed specially by output Formatters and is not treated as a formatting parameter to the **LogRecord** (p. 1719) message property.

Parameters

<i>level</i>	the Level (p. 1616) to log at.
<i>file</i>	File that the message was logged in.
<i>line</i>	the line number where the message was logged at.
<i>message</i>	the message to log.
<i>ex</i>	the Exception to log

6.319.3.23 `void decaf::util::logging::Logger::removeHandler (Handler * handler)`

Removes the specified **Handler** (p. 1401) from this logger, ownership of the **Handler** (p. 1401) pointer is returned to the caller.

Returns silently if the given **Handler** (p. 1401) is not found.

Parameters

<i>handler</i>	The Handler (p. 1401) to remove
----------------	--

6.319.3.24 `void decaf::util::logging::Logger::setFilter (Filter * filter)`

Set (p. 2397) a filter to control output on this **Logger** (p. 1693).

After passing the initial "level" check, the **Logger** (p. 1693) will call this **Filter** (p. 1333) to check if a log record should really be published.

The caller releases ownership of this filter to this logger

Parameters

<i>filter</i>	The Filter (p. 1333) to use, (can be NULL).
---------------	--

6.319.3.25 void decaf::util::logging::Logger::setLevel (const Level & *level*)
[inline]

Set (p. 2397) the log level specifying which message levels will be logged by this logger.

Message levels lower than this value will be discarded. The level value **Level::OFF** (p. 1620) can be used to turn off logging.

If the new level is the INHERIT **Level** (p. 1616), it means that this node should inherit its level from its nearest ancestor with a specific (non-INHERIT) level value.

Parameters

<i>level</i>	The new Level (p. 1616) value to use when logging.
--------------	---

6.319.3.26 void decaf::util::logging::Logger::setParent (Logger * *parent*)
[inline]

Set (p. 2397) the parent for this **Logger** (p. 1693).

This method is used by the **LogManager** (p. 1712) to update a **Logger** (p. 1693) when the namespace changes.

It should not be called from application code.

6.319.3.27 void decaf::util::logging::Logger::setUseParentHandlers (bool *value*)
[inline]

Specify whether or not this logger should send its output to it's parent **Logger** (p. 1693).

This means that any LogRecords will also be written to the parent's Handlers, and potentially to its parent, recursively up the namespace.

Parameters

<i>value</i>	True is output is to be written to the parent
--------------	---

6.319.3.28 virtual void decaf::util::logging::Logger::severe (const std::string & *file*,
const int *line*, const std::string *functionName*, const std::string & *message*)
[virtual]

Log a SEVERE **Level** (p. 1616) Log.

If the logger is currently enabled for the SEVERE message level then the given message is forwarded to all the registered output **Handler** (p. 1401) objects.

Parameters

<i>file</i>	The file name where the log was generated.
<i>line</i>	The line number where the log was generated.
<i>function-Name</i>	The name of the function that logged this.
<i>message</i>	The message to log at this Level (p. 1616).

6.319.3.29 `virtual void decaf::util::logging::Logger::throwing (const std::string & file, const int line, const std::string functionName, const decaf::lang::Throwable & thrown) [virtual]`

Log throwing an exception.

This is a convenience method to log that a method is terminating by throwing an exception. The logging is done using the FINER level.

If the logger is currently enabled for the given message level then the given arguments are stored in a **LogRecord** (p. 1719) which is forwarded to all registered output handlers. The **LogRecord** (p. 1719)'s message is set to "THROW".

Parameters

<i>file</i>	The file name where the log was generated.
<i>line</i>	The line number where the log was generated.
<i>function-Name</i>	The name of the function that logged this.
<i>thrown</i>	The Throwable that will be thrown, will be cloned.

6.319.3.30 `virtual void decaf::util::logging::Logger::warning (const std::string & file, const int line, const std::string functionName, const std::string & message) [virtual]`

Log a WARN **Level** (p. 1616) Log.

If the logger is currently enabled for the WARN message level then the given message is forwarded to all the registered output **Handler** (p. 1401) objects.

Parameters

<i>file</i>	The file name where the log was generated.
<i>line</i>	The line number where the log was generated.
<i>function-Name</i>	The name of the function that logged this.
<i>message</i>	The message to log at this Level (p. 1616).

The documentation for this class was generated from the following file:

- src/main/decaf/util/logging/Logger.h

6.320 decaf::util::logging::LoggerHierarchy Class Reference

```
#include <src/main/decaf/util/logging/LoggerHierarchy.h>
```

Public Member Functions

- **LoggerHierarchy** ()
- virtual **~LoggerHierarchy** ()

6.320.1 Constructor & Destructor Documentation

6.320.1.1 **decaf::util::logging::LoggerHierarchy::LoggerHierarchy** ()

6.320.1.2 virtual **decaf::util::logging::LoggerHierarchy::~~LoggerHierarchy** ()
[virtual]

The documentation for this class was generated from the following file:

- src/main/decaf/util/logging/**LoggerHierarchy.h**

6.321 activemq::io::LoggingInputStream Class Reference

```
#include <src/main/activemq/io/LoggingInputStream.h>
```

Inheritance diagram for activemq::io::LoggingInputStream:

Public Member Functions

- **LoggingInputStream** (decaf::io::InputStream *inputStream, bool own=false)

Creates a DataInputStream that uses the specified underlying InputStream.

- virtual **~LoggingInputStream** ()

Protected Member Functions

- virtual int **doReadByte** ()
- virtual int **doReadArrayBounded** (unsigned char *buffer, int size, int offset, int length)

6.321.1 Constructor & Destructor Documentation

6.321.1.1 `activemq::io::LoggingInputStream::LoggingInputStream (
 decaf::io::InputStream * inputStream, bool own = false)`

Creates a DataInputStream that uses the specified underlying InputStream.

Parameters

<i>inputStream</i>	the InputStream instance to wrap.
<i>own</i>	indicates if this class owns the wrapped string defaults to false.

6.321.1.2 `virtual activemq::io::LoggingInputStream::~~LoggingInputStream ()
 [virtual]`

6.321.2 Member Function Documentation

6.321.2.1 `virtual int activemq::io::LoggingInputStream::doReadArrayBounded (
 unsigned char * buffer, int size, int offset, int length) [protected,
 virtual]`

Reimplemented from `decaf::io::FilterInputStream` (p. 1338).

6.321.2.2 `virtual int activemq::io::LoggingInputStream::doReadByte ()
 [protected, virtual]`

Reimplemented from `decaf::io::FilterInputStream` (p. 1338).

The documentation for this class was generated from the following file:

- `src/main/activemq/io/LoggingInputStream.h`

6.322 activemq::io::LoggingOutputStream Class Reference

OutputStream filter that just logs the data being written.

```
#include <src/main/activemq/io/LoggingOutputStream.h>
```

Inheritance diagram for `activemq::io::LoggingOutputStream`:

Public Member Functions

- `LoggingOutputStream` (OutputStream *next, bool **own**=false)
Constructor.
- `virtual ~LoggingOutputStream ()`

Protected Member Functions

- virtual void **doWriteByte** (unsigned char c)
- virtual void **doWriteArrayBounded** (const unsigned char *buffer, int size, int offset, int length)

6.322.1 Detailed Description

OutputStream filter that just logs the data being written.

6.322.2 Constructor & Destructor Documentation

6.322.2.1 `activemq::io::LoggingOutputStream::LoggingOutputStream (OutputStream * next, bool own = false)`

Constructor.

Parameters

<i>next</i>	The OutputStream to wrap an write logs to.
<i>own</i>	If true, this object will control the lifetime of the output stream that it encapsulates.

6.322.2.2 `virtual activemq::io::LoggingOutputStream::~~LoggingOutputStream ()` [virtual]

6.322.3 Member Function Documentation

6.322.3.1 `virtual void activemq::io::LoggingOutputStream::doWriteArrayBounded (const unsigned char * buffer, int size, int offset, int length)` [protected, virtual]

Reimplemented from `decaf::io::FilterOutputStream` (p. 1343).

6.322.3.2 `virtual void activemq::io::LoggingOutputStream::doWriteByte (unsigned char c)` [protected, virtual]

Reimplemented from `decaf::io::FilterOutputStream` (p. 1343).

The documentation for this class was generated from the following file:

- `src/main/activemq/io/LoggingOutputStream.h`

6.323 activemq::transport::logging::LoggingTransport Class - Reference

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

```
#include <src/main/activemq/transport/logging/Logging-Transport.h>
```

Inheritance diagram for activemq::transport::logging::LoggingTransport:

Public Member Functions

- **LoggingTransport** (const **Pointer**< **Transport** > &next)

Constructor.

- virtual **~LoggingTransport** ()
- virtual void **onCommand** (const **Pointer**< **Command** > &command)

Event handler for the receipt of a command.

- virtual void **oneway** (const **Pointer**< **Command** > &command)

Sends a one-way command.

- virtual **Pointer**< **Response** > **request** (const **Pointer**< **Command** > &command)

Sends the given command to the broker and then waits for the response.

Parameters

command	<i>the command to be sent.</i>
---------	--------------------------------

Returns

the response from the broker.

Exceptions

IOException	<i>if an exception occurs during the read of the command.</i>
UnsupportedOperationException	<i>if this method is not implemented by this transport.</i>

- virtual **Pointer**< **Response** > **request** (const **Pointer**< **Command** > &command, unsigned int timeout)

Sends the given command to the broker and then waits for the response.

Parameters

command	<i>The command to be sent.</i>
timeout	<i>The time to wait for this response.</i>

Returns

the response from the broker.

Exceptions

IOException	<i>if an exception occurs during the read of the command.</i>
UnsupportedOperationException	<i>if this method is not implemented by this transport.</i>

6.323.1 Detailed Description

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

6.323.2 Constructor & Destructor Documentation

6.323.2.1 `activemq::transport::logging::LoggingTransport::LoggingTransport (const Pointer< Transport > & next)`

Constructor.

Parameters

<i>next</i>	- the next Transport (p. 2790) in the chain
-------------	--

6.323.2.2 `virtual activemq::transport::logging::LoggingTransport::~LoggingTransport () [inline, virtual]`

6.323.3 Member Function Documentation

6.323.3.1 `virtual void activemq::transport::logging::LoggingTransport::onCommand (const Pointer< Command > & command) [virtual]`

Event handler for the receipt of a command.

Parameters

<i>command</i>	- the received command object.
----------------	--------------------------------

Reimplemented from **activemq::transport::TransportFilter** (p. 2805).

6.323.3.2 `virtual void activemq::transport::logging::LoggingTransport::oneway (const Pointer< Command > & command) [virtual]`

Sends a one-way command.

Does not wait for any response from the broker.

Parameters

<i>command</i>	The command to be sent.
----------------	-------------------------

Exceptions

<i>IOException</i>	if an exception occurs during writing of the command.
<i>UnsupportedOperationException</i>	if this method is not implemented by this transport.

Reimplemented from **activemq::transport::TransportFilter** (p. 2806).

6.323.3.3 **virtual Pointer<Response> activemq::transport::logging::Logging-Transport::request (const Pointer< Command > & command)**
[virtual]

Sends the given command to the broker and then waits for the response.

Parameters

<i>command</i>	the command to be sent.
----------------	-------------------------

Returns

the response from the broker.

Exceptions

<i>IOException</i>	if an exception occurs during the read of the command.
<i>Unsupported-Operation-Exception</i>	if this method is not implemented by this transport.

Not supported by this class - throws an exception.

Reimplemented from **activemq::transport::TransportFilter** (p. 2807).

6.323.3.4 **virtual Pointer<Response> activemq::transport::logging::Logging-Transport::request (const Pointer< Command > & command, unsigned int timeout)** [virtual]

Sends the given command to the broker and then waits for the response.

Parameters

<i>command</i>	The command to be sent.
<i>timeout</i>	The time to wait for this response.

Returns

the response from the broker.

Exceptions

<i>IOException</i>	if an exception occurs during the read of the command.
<i>Unsupported-Operation-Exception</i>	if this method is not implemented by this transport.

Not supported by this class - throws an exception.

Reimplemented from **activemq::transport::TransportFilter** (p. 2807).

The documentation for this class was generated from the following file:

- src/main/activemq/transport/logging/**LoggingTransport.h**

6.324 decaf::util::logging::LogManager Class Reference

There is a single global **LogManager** (p. 1712) object that is used to maintain a set of shared state about Loggers and log services.

```
#include <src/main/decaf/util/logging/LogManager.h>
```

Public Member Functions

- virtual **~LogManager** ()
- bool **addLogger** (**Logger** *logger)
Add a named logger.
- **Logger** * **getLogger** (const std::string &name)
*Retrieves or creates a new **Logger** (p. 1693) using the name specified a new logger inherits the configuration of the logger's parent if there is no configuration data for the logger.*
- int **getLoggerNames** (const std::vector< std::string > &names)
*Gets a list of known **Logger** (p. 1693) Names from this Manager, new loggers added while this method is in progress are not guaranteed to be in the list.*
- void **setProperties** (const **util::Properties** &properties)
*Sets the **Properties** (p. 2200) this **LogManager** (p. 1712) should use to configure its loggers.*
- const **util::Properties** & **getProperties** () const
*Gets a reference to the Logging **Properties** (p. 2200) used by this logger.*
- std::string **getProperty** (const std::string &name)
*Gets the value of a named property of this **LogManager** (p. 1712).*
- void **addPropertyChangeListener** (PropertyChangeListener *listener)
*Adds a change listener for **LogManager** (p. 1712) **Properties** (p. 2200), adding the same instance of a change event listener does nothing.*
- void **removePropertyChangeListener** (PropertyChangeListener *listener)
*Removes a properties change listener from the **LogManager** (p. 1712), if the listener is not found of the param is NULL this method returns silently.*
- void **readConfiguration** ()
Reinitialize the logging properties and reread the logging configuration.
- void **readConfiguration** (**decaf::io::InputStream** *stream)
*Reinitialize the logging properties and reread the logging configuration from the given stream, which should be in **decaf.util.Properties** (p. 2200) format.*
- void **reset** ()
Reset the logging configuration.

Static Public Member Functions

- static **LogManager** & **getLogManager** ()
*Get the global **LogManager** (p. 1712) instance.*

Protected Member Functions

- **LogManager** ()
Constructor, hidden to protect against direct instantiation.
- **LogManager** (const **LogManager** &manager)
Copy Constructor.
- void **operator=** (const **LogManager** &manager)
Assignment operator.

Friends

- class **decaf::lang::Runtime**

6.324.1 Detailed Description

There is a single global **LogManager** (p. 1712) object that is used to maintain a set of shared state about Loggers and log services.

This **LogManager** (p. 1712) object:

- * Manages a hierarchical namespace of **Logger** (p. 1693) objects. All named Loggers are stored in this namespace.
- * Manages a set of logging control properties. These are simple key-value pairs that can be used by Handlers and other logging objects to configure themselves.

The global **LogManager** (p. 1712) object can be retrieved using **LogManager::getLogManager()** (p. 1716). The **LogManager** (p. 1712) object is created during class initialization and cannot subsequently be changed.

TODO By default, the **LogManager** (p. 1712) reads its initial configuration from a properties file "lib/logging.properties" in the JRE directory. If you edit that property file you can change the default logging configuration for all uses of that JRE.

In addition, the **LogManager** (p. 1712) uses two optional system properties that allow more control over reading the initial configuration:

- * "decaf.logger.config.class" * "decaf.logger.config.file"

These two properties may be set via the Preferences API, or as command line property definitions to the "java" command, or as system property definitions passed to JNI_CreateJavaVM.

If the "java.util.logging.config.class" property is set, then the property value is treated as a class name. The given class will be loaded, an object will be instantiated, and that object's constructor is responsible for reading in the initial configuration. (That object may

use other system properties to control its configuration.) The alternate configuration class can use `readConfiguration(InputStream)` to define properties in the **LogManager** (p. 1712).

If "decaf.util.logging.config.class" property is not set, then the "decaf.util.logging.config.-file" system property can be used to specify a properties file (in **decaf.util.Properties** (p. 2200) format). The initial logging configuration will be read from this file.

If neither of these properties is defined then, as described above, the **LogManager** (p. 1712) will read its initial configuration from a properties file "lib/logging.properties" in the working directory.

The properties for loggers and Handlers will have names starting with the dot-separated name for the handler or logger. ***TODO***

The global logging properties may include:

- * A property "handlers". This defines a whitespace separated list of class names for handler classes to load and register as handlers on the root **Logger** (p. 1693) (the - **Logger** (p. 1693) named ""). Each class name must be for a **Handler** (p. 1401) class which has a default constructor. Note that these Handlers may be created lazily, when they are first used.
- * A property "<logger>.handlers". This defines a whitespace or comma separated list of class names for handlers classes to load and register as handlers to the specified logger. Each class name must be for a **Handler** (p. 1401) class which has a default constructor. Note that these Handlers may be created lazily, when they are first used.
- * A property "<logger>.useParentHandlers". This defines a boolean value. By default every logger calls its parent in addition to handling the logging message itself, this often result in messages being handled by the root logger as well. When setting this property to false a **Handler** (p. 1401) needs to be configured for this logger otherwise no logging messages are delivered.
- * A property "config". - This property is intended to allow arbitrary configuration code to be run. The property defines a whitespace separated list of class names. A new instance will be created for each named class. The default constructor of each class may execute arbitrary code to update the logging configuration, such as setting logger levels, adding handlers, adding filters, etc.

Loggers are organized into a naming hierarchy based on their dot separated names. Thus "a.b.c" is a child of "a.b", but "a.b1" and a.b2" are peers.

All properties whose names end with ".level" are assumed to define log levels for - Loggers. Thus "foo.level" defines a log level for the logger called "foo" and (recursively) for any of its children in the naming hierarchy. Log Levels are applied in the order they are defined in the properties file. Thus level settings for child nodes in the tree should come after settings for their parents. The property name ".level" can be used to set the level for the root of the tree.

All methods on the **LogManager** (p. 1712) object are multi-thread safe.

Since

1.0

6.324.2 Constructor & Destructor Documentation

6.324.2.1 `virtual decaf::util::logging::LogManager::~~LogManager ()`
[virtual]

6.324.2.2 `decaf::util::logging::LogManager::LogManager ()` [protected]

Constructor, hidden to protect against direct instantiation.

6.324.2.3 `decaf::util::logging::LogManager::LogManager (const LogManager & manager)` [protected]

Copy Constructor.

Parameters

<i>manager</i>	the Manager to copy
----------------	---------------------

6.324.3 Member Function Documentation

6.324.3.1 `bool decaf::util::logging::LogManager::addLogger (Logger * logger)`

Add a named logger.

This does nothing and returns false if a logger with the same name is already registered.

The **Logger** (p. 1693) factory methods call this method to register each newly created **Logger** (p. 1693).

Parameters

<i>logger</i>	The new Logger (p. 1693) instance to add to this LogManager (p. 1712).
---------------	--

Exceptions

<i>NullPointerException</i>	if logger is NULL.
<i>IllegalArgumentException</i>	if the logger has no name.

6.324.3.2 `void decaf::util::logging::LogManager::addPropertyChangeListener (PropertyChangeListener * listener)`

Adds a change listener for **LogManager** (p. 1712) **Properties** (p. 2200), adding the same instance of a change event listener does nothing.

Parameters

<i>listener</i>	The PropertyChangeListener to add (can be NULL).
-----------------	--

6.324.3.3 `Logger* decaf::util::logging::LogManager::getLogger (const std::string & name)`

Retrieves or creates a new **Logger** (p. 1693) using the name specified a new logger inherits the configuration of the logger's parent if there is no configuration data for the logger.

Parameters

<i>name</i>	The name of the Logger (p. 1693).
-------------	--

6.324.3.4 `int decaf::util::logging::LogManager::getLoggerNames (const std::vector< std::string > & names)`

Gets a list of known **Logger** (p. 1693) Names from this Manager, new loggers added while this method is in progress are not guaranteed to be in the list.

Parameters

<i>names</i>	STL Vector to hold string logger names.
--------------	---

Returns

names count of how many loggers were inserted.

6.324.3.5 `static LogManager& decaf::util::logging::LogManager::getLogManager () [static]`

Get the global **LogManager** (p. 1712) instance.

Returns

A reference to the global **LogManager** (p. 1712) instance.

6.324.3.6 `const util::Properties& decaf::util::logging::LogManager::getProperties () const [inline]`

Gets a reference to the Logging **Properties** (p. 2200) used by this logger.

Returns

The **Logger** (p. 1693) **Properties** (p. 2200) Pointer

6.324.3.7 `std::string decaf::util::logging::LogManager::getProperty (const std::string & name)`

Gets the value of a named property of this **LogManager** (p. 1712).

Parameters

<i>name</i>	The name of the Property to retrieve.
-------------	---------------------------------------

Returns

the value of the property

6.324.3.8 `void decaf::util::logging::LogManager::operator= (const LogManager & manager)`
[protected]

Assignment operator.

Parameters

<i>manager</i>	the manager to assign from
----------------	----------------------------

6.324.3.9 `void decaf::util::logging::LogManager::readConfiguration ()`

Reinitialize the logging properties and reread the logging configuration.

The same rules are used for locating the configuration properties as are used at startup. So normally the logging properties will be re-read from the same file that was used at startup.

Any log level definitions in the new configuration file will be applied using **Logger.setLevel()** (p. 1704), if the target **Logger** (p. 1693) exists.

A PropertyChangeEvent will be fired after the properties are read.

Exceptions

<i>IOException</i>	if an I/O error occurs.
--------------------	-------------------------

6.324.3.10 `void decaf::util::logging::LogManager::readConfiguration (decaf::io::InputStream * stream)`

Reinitialize the logging properties and reread the logging configuration from the given stream, which should be in **decaf.util.Properties** (p. 2200) format.

A PropertyChangeEvent will be fired after the properties are read.

Any log level definitions in the new configuration file will be applied using **Logger.set-**

Level() (p. 1704), if the target **Logger** (p. 1693) exists.

Parameters

<i>stream</i>	The InputStream to read the Properties (p. 2200) from.
---------------	---

Exceptions

<i>NullPointerException</i>	if stream is NULL.
<i>IOException</i>	if an I/O error occurs.

6.324.3.11 void decaf::util::logging::LogManager::removePropertyChangeListener (PropertyChangeListener * listener)

Removes a properties change listener from the **LogManager** (p. 1712), if the listener is not found of the param is NULL this method returns silently.

Parameters

<i>listener</i>	The PropertyChangeListener to remove from the listeners set.
-----------------	--

6.324.3.12 void decaf::util::logging::LogManager::reset ()

Reset the logging configuration.

For all named loggers, the reset operation removes and closes all Handlers and (except for the root logger) sets the level to INHERIT. The root logger's level is set to **Level::INFO** (p. 1620).

6.324.3.13 void decaf::util::logging::LogManager::setProperties (const util::Properties & properties)

Sets the **Properties** (p. 2200) this **LogManager** (p. 1712) should use to configure its loggers.

Once set a properties change event is fired.

Parameters

<i>properties</i>	Pointer to read the configuration from
-------------------	--

6.324.4 Friends And Related Function Documentation

6.324.4.1 friend class decaf::lang::Runtime [friend]

The documentation for this class was generated from the following file:

- src/main/decaf/util/logging/LogManager.h

6.325 decaf::util::logging::LogRecord Class Reference

LogRecord (p. 1719) objects are used to pass logging requests between the logging framework and individual log Handlers.

```
#include <src/main/decaf/util/logging/LogRecord.h>
```

Public Member Functions

- **LogRecord** ()
- virtual **~LogRecord** ()
- **Level** **getLevel** () const
*Get **Level** (p. 1616) of this log record.*
- void **setLevel** (**Level** value)
*Set (p. 2397) the **Level** (p. 1616) of this Log Record.*
- const std::string & **getLoggerName** () const
*Gets the Source **Logger** (p. 1693)'s Name.*
- void **setLoggerName** (const std::string &loggerName)
*Sets the Source **Logger** (p. 1693)'s Name.*
- const std::string & **getSourceFile** () const
Gets the Source Log File name.
- void **setSourceFile** (const std::string &sourceFile)
Sets the Source Log File Name.
- unsigned int **getSourceLine** () const
Gets the Source Log line number.
- void **setSourceLine** (unsigned int sourceLine)
Sets the Source Log line number.
- const std::string & **getMessage** () const
Gets the Message to be Logged.
- void **setMessage** (const std::string &message)
Sets the Message to be Logged.
- const std::string & **getSourceFunction** () const
Gets the name of the function where this log was logged.
- void **setSourceFunction** (const std::string &functionName)
Sets the name of the function where this log was logged.
- long long **getTimestamp** () const
Gets the time in mills that this message was logged.
- void **setTimestamp** (long long timeStamp)
Sets the time in mills that this message was logged.
- long long **getTreadId** () const
Gets the Thread Id where this Log was created.

- void **setTreadId** (long long threadId)
Sets the Thread Id where this Log was created.
- **decaf::lang::Throwable * getThrown** () const
*Gets any Throwable associated with this **LogRecord** (p. 1719).*
- void **setThrown** (**decaf::lang::Throwable *thrown**)
*Sets the Throwable associated with this **LogRecord** (p. 1719), the pointer becomes the property of this instance of the **LogRecord** (p. 1719) and will be deleted when the record is destroyed.*

6.325.1 Detailed Description

LogRecord (p. 1719) objects are used to pass logging requests between the logging framework and individual log Handlers.

When a **LogRecord** (p. 1719) is passed into the logging framework it logically belongs to the framework and should no longer be used or updated by the client application.

Since

1.0

6.325.2 Constructor & Destructor Documentation

6.325.2.1 **decaf::util::logging::LogRecord::LogRecord** ()

6.325.2.2 **virtual decaf::util::logging::LogRecord::~~LogRecord** () `[virtual]`

6.325.3 Member Function Documentation

6.325.3.1 **Level decaf::util::logging::LogRecord::getLevel** () const `[inline]`

Get **Level** (p. 1616) of this log record.

Returns

Level (p. 1616) enumeration value.

6.325.3.2 **const std::string& decaf::util::logging::LogRecord::getLoggerName** ()
const `[inline]`

Gets the Source **Logger** (p. 1693)'s Name.

Returns

the source loggers name

6.325.3.3 `const std::string& decaf::util::logging::LogRecord::getMessage () const`
`[inline]`

Gets the Message to be Logged.

Returns

the source logger's message

6.325.3.4 `const std::string& decaf::util::logging::LogRecord::getSourceFile () const`
`[inline]`

Gets the Source Log File name.

Returns

the source loggers name

6.325.3.5 `const std::string& decaf::util::logging::LogRecord::getSourceFunction ()`
`const [inline]`

Gets the name of the function where this log was logged.

Returns

the source logger's message

6.325.3.6 `unsigned int decaf::util::logging::LogRecord::getSourceLine () const`
`[inline]`

Gets the Source Log line number.

Returns

the source loggers line number

6.325.3.7 `decaf::lang::Throwable* decaf::util::logging::LogRecord::getThrown () const`
`[inline]`

Gets any Throwable associated with this **LogRecord** (p. 1719).

Returns

point to a Throwable instance or Null.

6.325.3.8 `long long decaf::util::logging::LogRecord::getTimestamp () const`
[inline]

Gets the time in mills that this message was logged.

Returns

UTC time in milliseconds

6.325.3.9 `long long decaf::util::logging::LogRecord::getTreadId () const`
[inline]

Gets the Thread Id where this Log was created.

Returns

the source loggers line number

6.325.3.10 `void decaf::util::logging::LogRecord::setLevel (Level value)`
[inline]

Set (p. 2397) the **Level** (p. 1616) of this Log Record.

Parameters

<i>value</i>	Level (p. 1616) Enumeration Value
--------------	--

6.325.3.11 `void decaf::util::logging::LogRecord::setLoggerName (const std::string & loggerName)` [inline]

Sets the Source **Logger** (p. 1693)'s Name.

Parameters

<i>loggerName</i>	the source loggers name
-------------------	-------------------------

6.325.3.12 `void decaf::util::logging::LogRecord::setMessage (const std::string & message)` [inline]

Sets the Message to be Logged.

Parameters

<i>message</i>	the source loggers message
----------------	----------------------------

6.325.3.13 `void decaf::util::logging::LogRecord::setSourceFile (const std::string & sourceFile) [inline]`

Sets the Source Log File Name.

Parameters

<i>sourceFile</i>	the source loggers name
-------------------	-------------------------

6.325.3.14 `void decaf::util::logging::LogRecord::setSourceFunction (const std::string & functionName) [inline]`

Sets the name of the function where this log was logged.

Parameters

<i>function-Name</i>	the source of the log
----------------------	-----------------------

6.325.3.15 `void decaf::util::logging::LogRecord::setSourceLine (unsigned int sourceLine) [inline]`

Sets the Source Log line number.

Parameters

<i>sourceLine</i>	the source logger's line number
-------------------	---------------------------------

6.325.3.16 `void decaf::util::logging::LogRecord::setThrown (decaf::lang::Throwable * thrown) [inline]`

Sets the Throwable associated with this **LogRecord** (p. 1719), the pointer becomes the property of this instance of the **LogRecord** (p. 1719) and will be deleted when the record is destroyed.

Parameters

<i>thrown</i>	A pointer to a Throwable that will be associated with this record.
---------------	--

6.325.3.17 `void decaf::util::logging::LogRecord::setTimestamp (long long timeStamp) [inline]`

Sets the time in mills that this message was logged.

Parameters

<i>timeStamp</i>	UTC Time in Milliseconds.
------------------	---------------------------

6.325.3.18 void decaf::util::logging::LogRecord::setTreadId (long long *threadId*)
[inline]

Sets the Thread Id where this Log was created.

Parameters

<i>threadId</i>	the source logger's line number
-----------------	---------------------------------

The documentation for this class was generated from the following file:

- src/main/decaf/util/logging/**LogRecord.h**

6.326 decaf::util::logging::LogWriter Class Reference

```
#include <src/main/decaf/util/logging/LogWriter.h>
```

Public Member Functions

- **LogWriter** ()
- virtual ~**LogWriter** ()
- virtual void **log** (const std::string &file, const int line, const std::string &prefix, const std::string &message)
Writes a message to the output destination.
- virtual void **log** (const std::string &message)
Writes a message to the output destination.

Static Public Member Functions

- static **LogWriter** & **getInstance** ()
Get the singleton instance.
- static void **returnInstance** ()
Returns a Checked out instance of this Writer.
- static void **destroy** ()
*Forcefully Delete the Instance of this **LogWriter** (p. 1724) even if there are outstanding references.*

6.326.1 Constructor & Destructor Documentation

6.326.1.1 `decaf::util::logging::LogWriter::LogWriter ()`

6.326.1.2 `virtual decaf::util::logging::LogWriter::~~LogWriter ()` `[virtual]`

6.326.2 Member Function Documentation

6.326.2.1 `static void decaf::util::logging::LogWriter::destroy ()` `[static]`

Forcefully Delete the Instance of this **LogWriter** (p. 1724) even if there are outstanding references.

6.326.2.2 `static LogWriter& decaf::util::logging::LogWriter::getInstance ()`
`[static]`

Get the singleton instance.

6.326.2.3 `virtual void decaf::util::logging::LogWriter::log (const std::string & file, const int line, const std::string & prefix, const std::string & message)` `[virtual]`

Writes a message to the output destination.

Parameters

<i>file</i>	
<i>line</i>	
<i>prefix</i>	
<i>message</i>	

6.326.2.4 `virtual void decaf::util::logging::LogWriter::log (const std::string & message)`
`[virtual]`

Writes a message to the output destination.

Parameters

<i>message</i>	
----------------	--

6.326.2.5 `static void decaf::util::logging::LogWriter::returnInstance ()`
`[static]`

Returns a Checked out instance of this Writer.

The documentation for this class was generated from the following file:

- src/main/decaf/util/logging/LogWriter.h

6.327 decaf::lang::Long Class Reference

```
#include <src/main/decaf/lang/Long.h>
```

Inheritance diagram for decaf::lang::Long:

Public Member Functions

- **Long** (long long value)
- **Long** (const std::string &value)
*Constructs a new **Long** (p. 1726) and attempts to convert the given string to an long long value, assigning it to the new object is successful or throwing a NumberFormatException if the string is not a properly formatted long long.*
- virtual ~**Long** ()
- virtual int **compareTo** (const **Long** &l) const
*Compares this **Long** (p. 1726) instance with another.*
- bool **equals** (const **Long** &l) const
- virtual bool **operator==** (const **Long** &l) const
Compares equality between this object and the one passed.
- virtual bool **operator<** (const **Long** &l) const
Compares this object to another and returns true if this object is considered to be less than the one passed.
- virtual int **compareTo** (const long long &l) const
*Compares this **Long** (p. 1726) instance with another.*
- bool **equals** (const long long &l) const
- virtual bool **operator==** (const long long &l) const
Compares equality between this object and the one passed.
- virtual bool **operator<** (const long long &l) const
Compares this object to another and returns true if this object is considered to be less than the one passed.
- std::string **toString** () const
- virtual double **doubleValue** () const
Answers the double value which the receiver represents.
- virtual float **floatValue** () const
Answers the float value which the receiver represents.
- virtual unsigned char **byteValue** () const
Answers the byte value which the receiver represents.
- virtual short **shortValue** () const
Answers the short value which the receiver represents.
- virtual int **intValue** () const

Answers the int value which the receiver represents.

- virtual long long **longValue** () const

Answers the long value which the receiver represents.

Static Public Member Functions

- static int **bitCount** (long long value)
Returns the number of one-bits in the two's complement binary representation of the specified int value.
- static **Long decode** (const std::string &value)
*Decodes a **String** (p. 2620) into a **Long** (p. 1726).*
- static long long **highestOneBit** (long long value)
Returns an long long value with at most a single one-bit, in the position of the highest-order ("leftmost") one-bit in the specified int value.
- static long long **lowestOneBit** (long long value)
Returns an long long value with at most a single one-bit, in the position of the lowest-order ("rightmost") one-bit in the specified int value.
- static int **numberOfLeadingZeros** (long long value)
Returns the number of zero bits preceding the highest-order ("leftmost") one-bit in the two's complement binary representation of the specified long long value.
- static int **numberOfTrailingZeros** (long long value)
Returns the number of zero bits following the lowest-order ("rightmost") one-bit in the two's complement binary representation of the specified long long value.
- static long long **parseLong** (const std::string &value)
Parses the string argument as a signed decimal long.
- static long long **parseLong** (const std::string &value, int radix)
*Returns a **Long** (p. 1726) object holding the value extracted from the specified string when parsed with the radix given by the second argument.*
- static long long **reverseBytes** (long long value)
Returns the value obtained by reversing the order of the bytes in the two's complement representation of the specified long long value.
- static long long **reverse** (long long value)
Returns the value obtained by reversing the order of the bits in the two's complement binary representation of the specified long long value.
- static long long **rotateLeft** (long long value, int distance)
Returns the value obtained by rotating the two's complement binary representation of the specified value left by the specified number of bits.
- static long long **rotateRight** (long long value, int distance)
Returns the value obtained by rotating the two's complement binary representation of the specified value right by the specified number of bits.
- static int **signum** (long long value)
Returns the signum function of the specified value.
- static std::string **toString** (long long value)
*Converts the long to a **String** (p. 2620) representation.*
- static std::string **toString** (long long value, int radix)

- static std::string **toHexString** (long long value)
Returns a string representation of the integer argument as an unsigned integer in base 16.
- static std::string **toOctalString** (long long value)
Returns a string representation of the long long argument as an unsigned long long in base 8.
- static std::string **toBinaryString** (long long value)
Returns a string representation of the long long argument as an unsigned long long in base 2.
- static **Long valueOf** (long long value)
*Returns a **Long** (p. 1726) instance representing the specified int value.*
- static **Long valueOf** (const std::string &value)
*Returns a **Long** (p. 1726) object holding the value given by the specified std::string.*
- static **Long valueOf** (const std::string &value, int radix)
*Returns a **Long** (p. 1726) object holding the value extracted from the specified std::string when parsed with the radix given by the second argument.*

Static Public Attributes

- static const int **SIZE** = 64
The size in bits of the primitive long long type.
- static const long long **MAX_VALUE** = (long long)0x7FFFFFFFFFFFFFFFLL
The maximum value that the primitive type can hold.
- static const long long **MIN_VALUE** = (long long)0x8000000000000000LL
The minimum value that the primitive type can hold.

6.327.1 Constructor & Destructor Documentation

6.327.1.1 decaf::lang::Long::Long (long long value)

Parameters

<i>value</i>	- the primitive long long to wrap
--------------	-----------------------------------

6.327.1.2 decaf::lang::Long::Long (const std::string & value)

Constructs a new **Long** (p. 1726) and attempts to convert the given string to an long long value, assigning it to the new object is successful or throwing a `NumberFormatException` if the string is not a properly formatted long long.

Parameters

<i>value</i>	The string to convert to a primitive type to wrap.
--------------	--

Exceptions

<i>NumberFormatException</i>	if the string is not a a valid 64bit long.
------------------------------	--

6.327.1.3 `virtual decaf::lang::Long::~~Long () [inline, virtual]`

6.327.2 Member Function Documentation

6.327.2.1 `static int decaf::lang::Long::bitCount (long long value) [static]`

Returns the number of one-bits in the two's complement binary representation of the specified int value.

This function is sometimes referred to as the population count.

Parameters

<i>value</i>	- the long long to count
--------------	--------------------------

Returns

the number of one-bits in the two's complement binary representation of the specified long long value.

6.327.2.2 `virtual unsigned char decaf::lang::Long::byteValue () const [inline, virtual]`

Answers the byte value which the receiver represents.

Returns

int the value of the receiver.

Reimplemented from **decaf::lang::Number** (p. 1993).

6.327.2.3 `virtual int decaf::lang::Long::compareTo (const Long & l) const [virtual]`

Compares this **Long** (p. 1726) instance with another.

Parameters

<i>l</i>	- the Long (p. 1726) instance to be compared
----------	---

Returns

zero if this object represents the same integer value as the argument; a positive value if this object represents a value greater than the passed in value, and -1 if this object represents a value less than the passed in value.

Implements **decaf::lang::Comparable**< **Long** > (p. 886).

6.327.2.4 `virtual int decaf::lang::Long::compareTo (const long long & /) const`
[virtual]

Compares this **Long** (p. 1726) instance with another.

Parameters

/	- the Integer (p. 1500) instance to be compared
---	--

Returns

zero if this object represents the same integer value as the argument; a positive value if this object represents a value greater than the passed in value, and -1 if this object represents a value less than the passed in value.

Implements **decaf::lang::Comparable**< **long long** > (p. 886).

6.327.2.5 `static Long decaf::lang::Long::decode (const std::string & value)`
[static]

Decodes a **String** (p. 2620) into a **Long** (p. 1726).

Accepts decimal, hexadecimal, and octal numbers given by the following grammar:

The sequence of characters following an (optional) negative sign and/or radix specifier ("0x", "0X", "#", or leading zero) is parsed as by the Integer.parseInt method with the indicated radix (10, 16, or 8). This sequence of characters must represent a positive value or a NumberFormatException will be thrown. The result is negated if first character of the specified **String** (p. 2620) is the minus sign. No whitespace characters are permitted in the string.

Parameters

<i>value</i>	- The string to decode
--------------	------------------------

Returns

a **Long** (p. 1726) object containing the decoded value

Exceptions

<i>NumberFormatException</i>	if the string is not formatted correctly.
------------------------------	---

6.327.2.6 `virtual double decaf::lang::Long::doubleValue () const [inline, virtual]`

Answers the double value which the receiver represents.

Returns

double the value of the receiver.

Implements **decaf::lang::Number** (p. 1993).

6.327.2.7 `bool decaf::lang::Long::equals (const Long & /) const [inline, virtual]`

Parameters

/	- the Long (p. 1726) object to compare against.
---	--

Returns

true if the two **Integer** (p. 1500) Objects have the same value.

Implements **decaf::lang::Comparable< Long >** (p. 887).

6.327.2.8 `bool decaf::lang::Long::equals (const long long & /) const [inline, virtual]`

Parameters

/	- the Long (p. 1726) object to compare against.
---	--

Returns

true if the two **Integer** (p. 1500) Objects have the same value.

Implements **decaf::lang::Comparable< long long >** (p. 887).

6.327.2.9 `virtual float decaf::lang::Long::floatValue () const [inline, virtual]`

Answers the float value which the receiver represents.

Returns

float the value of the receiver.

Implements **decaf::lang::Number** (p. 1993).

6.327.2.10 `static long long decaf::lang::Long::highestOneBit (long long value)`
[static]

Returns an long long value with at most a single one-bit, in the position of the highest-order ("leftmost") one-bit in the specified int value.

Returns zero if the specified value has no one-bits in its two's complement binary representation, that is, if it is equal to zero.

Parameters

<i>value</i>	- the long long to be inspected
--------------	---------------------------------

Returns

an long long value with a single one-bit, in the position of the highest-order one-bit in the specified value, or zero if the specified value is itself equal to zero.

6.327.2.11 `virtual int decaf::lang::Long::intValue () const` [inline, virtual]

Answers the int value which the receiver represents.

Returns

int the value of the receiver.

Implements **decaf::lang::Number** (p. 1993).

6.327.2.12 `virtual long long decaf::lang::Long::longValue () const` [inline, virtual]

Answers the long value which the receiver represents.

Returns

long the value of the receiver.

Implements **decaf::lang::Number** (p. 1994).

6.327.2.13 `static long long decaf::lang::Long::lowestOneBit (long long value)`
[static]

Returns an long long value with at most a single one-bit, in the position of the lowest-order ("rightmost") one-bit in the specified int value.

Returns zero if the specified value has no one-bits in its two's complement binary representation, that is, if it is equal to zero.

Parameters

<i>value</i>	- the long long to be inspected
--------------	---------------------------------

Returns

an long long value with a single one-bit, in the position of the lowest-order one-bit in the specified value, or zero if the specified value is itself equal to zero.

6.327.2.14 `static int decaf::lang::Long::numberOfLeadingZeros (long long value)`
[static]

Returns the number of zero bits preceding the highest-order ("leftmost") one-bit in the two's complement binary representation of the specified long long value.

Returns 64 if the specified value has no one-bits in its two's complement representation, in other words if it is equal to zero.

Note that this method is closely related to the logarithm base 2. For all positive int values *x*:

* floor(log2(*x*)) = 63 - numberOfLeadingZeros(*x*) * ceil(log2(*x*)) = 64 - numberOfLeadingZeros(*x* - 1)

Parameters

<i>value</i>	- the long long to be inspected
--------------	---------------------------------

Returns

the number of zero bits preceding the highest-order ("leftmost") one-bit in the two's complement binary representation of the specified long long value, or 64 if the value is equal to zero.

6.327.2.15 `static int decaf::lang::Long::numberOfTrailingZeros (long long value)`
[static]

Returns the number of zero bits following the lowest-order ("rightmost") one-bit in the two's complement binary representation of the specified long long value.

Returns 64 if the specified value has no one-bits in its two's complement representation, in other words if it is equal to zero.

Parameters

<i>value</i>	- the int to be inspected
--------------	---------------------------

Returns

the number of zero bits following the lowest-order ("rightmost") one-bit in the two's complement binary representation of the specified long long value, or 64 if the value is equal to zero.

6.327.2.16 `virtual bool decaf::lang::Long::operator< (const Long & /) const [inline, virtual]`

Compares this object to another and returns true if this object is considered to be less than the one passed.

This

Parameters

/	- the value to be compared to this one.
---	---

Returns

true if this object is equal to the one passed.

Implements **decaf::lang::Comparable< Long >** (p. 887).

6.327.2.17 `virtual bool decaf::lang::Long::operator< (const long long & /) const [inline, virtual]`

Compares this object to another and returns true if this object is considered to be less than the one passed.

This

Parameters

/	- the value to be compared to this one.
---	---

Returns

true if this object is equal to the one passed.

Implements **decaf::lang::Comparable< long long >** (p. 887).

6.327.2.18 `virtual bool decaf::lang::Long::operator== (const Long & /) const [inline, virtual]`

Compares equality between this object and the one passed.

Parameters

	<i>/</i> - the value to be compared to this one.
--	--

Returns

true if this object is equal to the one passed.

Implements **decaf::lang::Comparable**< **Long** > (p. 888).

6.327.2.19 `virtual bool decaf::lang::Long::operator==(const long long & /) const`
`[inline, virtual]`

Compares equality between this object and the one passed.

Parameters

	<i>/</i> - the value to be compared to this one.
--	--

Returns

true if this object is equal to the one passed.

Implements **decaf::lang::Comparable**< **long long** > (p. 888).

6.327.2.20 `static long long decaf::lang::Long::parseLong (const std::string & value)`
`[static]`

Parses the string argument as a signed decimal long.

The characters in the string must all be decimal digits, except that the first character may be an ASCII minus sign '-' to indicate a negative value. The resulting long value is returned, exactly as if the argument and the radix 10 were given as arguments to the `parseLong(java.lang.String, int)` method.

Note that the characters LL or ULL are not permitted to appear at the end of this string as would normally be permitted in a C++ program.

Parameters

<i>value</i>	- String (p. 2620) to parse
--------------	------------------------------------

Returns

long long value

Exceptions

<i>NumberFormatException</i>	on invalid string value
------------------------------	-------------------------

6.327.2.21 `static long long decaf::lang::Long::parseLong (const std::string & value, int radix) [static]`

Returns a **Long** (p. 1726) object holding the value extracted from the specified string when parsed with the radix given by the second argument.

The first argument is interpreted as representing a signed long in the radix specified by the second argument, exactly as if the arguments were given to the `parseLong(std::string, int)` method. The result is a **Long** (p. 1726) object that represents the long long value specified by the string.

Parameters

<i>value</i>	- String (p. 2620) to parse
<i>radix</i>	- the base encoding of the string

Returns

long long value

Exceptions

<i>NumberFormatException</i>	on invalid string value
------------------------------	-------------------------

6.327.2.22 `static long long decaf::lang::Long::reverse (long long value) [static]`

Returns the value obtained by reversing the order of the bits in the two's complement binary representation of the specified long long value.

Parameters

<i>value</i>	- the value whose bits are to be reversed
--------------	---

Returns

the reversed bits long long.

6.327.2.23 `static long long decaf::lang::Long::reverseBytes (long long value) [static]`

Returns the value obtained by reversing the order of the bytes in the two's complement representation of the specified long long value.

Parameters

<i>value</i>	- the long long whose bytes we are to reverse
--------------	---

Returns

the reversed long long.

6.327.2.24 `static long long decaf::lang::Long::rotateLeft (long long value, int distance)`
`[static]`

Returns the value obtained by rotating the two's complement binary representation of the specified value left by the specified number of bits.

(Bits shifted out of the left hand, or high-order, side reenter on the right, or low-order.)

Note that left rotation with a negative distance is equivalent to right rotation: `rotateLeft(val, -distance) == rotateRight(val, distance)`. Note also that rotation by any multiple of 32 is a no-op, so all but the last five bits of the rotation distance can be ignored, even if the distance is negative: `rotateLeft(val, distance) == rotateLeft(val, distance & 0x1F)`.

Parameters

<i>value</i>	- the long long to be inspected
<i>distance</i>	- the number of bits to rotate

Returns

the value obtained by rotating the two's complement binary representation of the specified value left by the specified number of bits.

6.327.2.25 `static long long decaf::lang::Long::rotateRight (long long value, int distance)`
`[static]`

Returns the value obtained by rotating the two's complement binary representation of the specified value right by the specified number of bits.

(Bits shifted out of the right hand, or low-order, side reenter on the left, or high-order.)

Note that right rotation with a negative distance is equivalent to left rotation: `rotateRight(val, -distance) == rotateLeft(val, distance)`. Note also that rotation by any multiple of 32 is a no-op, so all but the last five bits of the rotation distance can be ignored, even if the distance is negative: `rotateRight(val, distance) == rotateRight(val, distance & 0x1F)`.

Parameters

<i>value</i>	- the long long to be inspected
<i>distance</i>	- the number of bits to rotate

Returns

the value obtained by rotating the two's complement binary representation of the specified value right by the specified number of bits.

6.327.2.26 `virtual short decaf::lang::Long::shortValue () const [inline, virtual]`

Answers the short value which the receiver represents.

Returns

int the value of the receiver.

Reimplemented from **decaf::lang::Number** (p. 1994).

6.327.2.27 `static int decaf::lang::Long::signum (long long value) [static]`

Returns the signum function of the specified value.

(The return value is -1 if the specified value is negative; 0 if the specified value is zero; and 1 if the specified value is positive.)

Parameters

<i>value</i>	- the long long to be inspected
--------------	---------------------------------

Returns

the signum function of the specified long long value.

6.327.2.28 `static std::string decaf::lang::Long::toBinaryString (long long value) [static]`

Returns a string representation of the long long argument as an unsigned long long in base 2.

The unsigned long long value is the argument plus 2^{32} if the argument is negative; otherwise it is equal to the argument. This value is converted to a string of ASCII digits in binary (base 2) with no extra leading 0s. If the unsigned magnitude is zero, it is represented by a single zero character '0'; otherwise, the first character of the representation of the unsigned magnitude will not be the zero character. The characters '0' and '1' are used as binary digits.

Parameters

<i>value</i>	- the long long to be translated to a binary string
--------------	---

Returns

the unsigned long long value as a binary string

6.327.2.29 `static std::string decaf::lang::Long::toHexString (long long value)`
[static]

Returns a string representation of the integer argument as an unsigned integer in base 16.

The unsigned integer value is the argument plus 2^{32} if the argument is negative; otherwise, it is equal to the argument. This value is converted to a string of ASCII digits in hexadecimal (base 16) with no extra leading 0s. If the unsigned magnitude is zero, it is represented by a single zero character '0'; otherwise, the first character of the representation of the unsigned magnitude will not be the zero character. The following characters are used as hexadecimal digits:

0123456789abcdef

If uppercase letters are desired, the `toUpperCase()` method may be called on the result:

Parameters

<i>value</i>	- the long long to be translated to an Octal string
--------------	---

Returns

the unsigned long long value as a Octal string

6.327.2.30 `static std::string decaf::lang::Long::toOctalString (long long value)`
[static]

Returns a string representation of the long long argument as an unsigned long long in base 8.

The unsigned long long value is the argument plus 2^{32} if the argument is negative; otherwise, it is equal to the argument. This value is converted to a string of ASCII digits in octal (base 8) with no extra leading 0s.

If the unsigned magnitude is zero, it is represented by a single zero character '0'; otherwise, the first character of the representation of the unsigned magnitude will not be the zero character. The following characters are used as octal digits:

01234567

Parameters

<i>value</i>	- the long long to be translated to an Octal string
--------------	---

Returns

the unsigned long long value as a Octal string

6.327.2.31 `std::string decaf::lang::Long::toString () const`

Returns

this **Long** (p. 1726) Object as a **String** (p. 2620) Representation

6.327.2.32 `static std::string decaf::lang::Long::toString (long long value) [static]`

Converts the long to a **String** (p. 2620) representation.

Parameters

<i>value</i>	The long to convert to a std::string.
--------------	---------------------------------------

Returns

string representation

6.327.2.33 `static std::string decaf::lang::Long::toString (long long value, int radix) [static]`

6.327.2.34 `static Long decaf::lang::Long::valueOf (long long value) [inline, static]`

Returns a **Long** (p. 1726) instance representing the specified int value.

Parameters

<i>value</i>	- the long long to wrap
--------------	-------------------------

Returns

the new **Integer** (p. 1500) object wrapping value.

6.327.2.35 `static Long decaf::lang::Long::valueOf (const std::string & value) [static]`

Returns a **Long** (p. 1726) object holding the value given by the specified std::string.

The argument is interpreted as representing a signed decimal long long, exactly as if the argument were given to the `parseLong(std::string)` method. The result is a **Integer** (p. 1500) object that represents the long long value specified by the string.

Parameters

<i>value</i>	- std::string to parse as base 10
--------------	-----------------------------------

Returns

new **Long** (p. 1726) Object wrapping the primitive

Exceptions

<i>NumberFormatException</i>	if the string is not a decimal long long.
------------------------------	---

6.327.2.36 `static Long decaf::lang::Long::valueOf (const std::string & value, int radix)`
`[static]`

Returns a **Long** (p. 1726) object holding the value extracted from the specified `std::string` when parsed with the radix given by the second argument.

The first argument is interpreted as representing a signed long long in the radix specified by the second argument, exactly as if the argument were given to the `parseLong(std::string, int)` method. The result is a **Long** (p. 1726) object that represents the long long value specified by the string.

Parameters

<i>value</i>	- <code>std::string</code> to parse as base (<code>radix</code>)
<i>radix</i>	- base of the string to parse.

Returns

new **Long** (p. 1726) Object wrapping the primitive

Exceptions

<i>NumberFormatException</i>	if the string is not a valid long long.
------------------------------	---

6.327.3 Field Documentation

6.327.3.1 `const long long decaf::lang::Long::MAX_VALUE = (long long)0xFFFFFFFFFFFFFFFFLL` `[static]`

The maximum value that the primitive type can hold.

6.327.3.2 `const long long decaf::lang::Long::MIN_VALUE = (long long)0x8000000000000000LL` `[static]`

The minimum value that the primitive type can hold.

6.327.3.3 `const int decaf::lang::Long::SIZE = 64` [static]

The size in bits of the primitive long long type.

The documentation for this class was generated from the following file:

- src/main/decaf/lang/**Long.h**

6.328 decaf::internal::nio::LongArrayBuffer Class Reference

```
#include <src/main/decaf/internal/nio/LongArrayBuffer.h>
```

Inheritance diagram for decaf::internal::nio::LongArrayBuffer:

Public Member Functions

- **LongArrayBuffer** (int size, bool readOnly=false)
*Creates a **IntArrayBuffer** (p. 1477) object that has its backing array allocated internally and is then owned and deleted when this object is deleted.*
- **LongArrayBuffer** (long long *array, int size, int offset, int length, bool readOnly=false)
*Creates a **LongArrayBuffer** (p. 1742) object that wraps the given array.*
- **LongArrayBuffer** (const decaf::lang::Pointer< **ByteArrayAdapter** > &array, int offset, int length, bool readOnly=false)
Creates a byte buffer that wraps the passed ByteArrayAdapter and start at the given offset.
- **LongArrayBuffer** (const **LongArrayBuffer** &other)
*Create a **LongArrayBuffer** (p. 1742) that mirrors this one, meaning it shares a reference to this buffers ByteArrayAdapter and when changes are made to that data it is reflected in both.*
- virtual ~**LongArrayBuffer** ()
- virtual long long * array ()
*Returns the long long array that backs this buffer (optional operation).
Modifications to this buffer's content will cause the returned array's content to be modified, and vice versa.
Invoke the hasArray method before invoking this method in order to ensure that this buffer has an accessible backing array.*
Returns
*the array that backs this **Buffer** (p. 582).*

Exceptions

ReadOnlyBuffer-Exception (p. 2244)	<i>if this Buffer (p. 582) is read only.</i>
UnsupportedOperation-Exception	<i>if the underlying store has no array.</i>

- virtual int **arrayOffset** ()

Returns the offset within this buffer's backing array of the first element of the buffer (optional operation).

Invoke the `hasArray` method before invoking this method in order to ensure that this buffer has an accessible backing array.

Returns

The offset long longo the backing array where index zero starts.

Exceptions

ReadOnlyBuffer-Exception (p. 2244)	if this Buffer (p. 582) is read only.
UnsupportedOperation-Exception	if the underlying store has no array.

- virtual **LongBuffer * asReadOnlyBuffer** () const

Creates a new, read-only long long buffer that shares this buffer's content.

The content of the new buffer will be that of this buffer. Changes to this buffer's content will be visible in the new buffer; the new buffer itself, however, will be read-only and will not allow the shared content to be modified. The two buffers' position, limit, and mark values will be independent.

If this buffer is itself read-only then this method behaves in exactly the same way as the `duplicate` method.

The new buffer's capacity, limit, position, and mark values will be identical to those of this buffer.

Returns

The new, read-only long long buffer which the caller then owns.

- virtual **LongBuffer & compact** ()

Compacts this buffer.

The bytes between the buffer's current position and its limit, if any, are copied to the beginning of the buffer. That is, the byte at index $p = \text{position}()$ (p. 587) is copied to index zero, the byte at index $p + 1$ is copied to index one, and so forth until the byte at index $\text{limit}()$ (p. 586) - 1 is copied to index $n = \text{limit}()$ (p. 586) - 1 - p . The buffer's position is then set to $n+1$ and its limit is set to its capacity. The mark, if defined, is discarded.

The buffer's position is set to the number of bytes copied, rather than to zero, so that an invocation of this method can be followed immediately by an invocation of another relative put method.

Returns

a reference to this **LongBuffer** (p. 1752).

Exceptions

ReadOnlyBuffer-Exception (p. 2244)	if this buffer is read-only.
--	------------------------------

- virtual **LongBuffer * duplicate** ()

Creates a new long long buffer that shares this buffer's content.

The content of the new buffer will be that of this buffer. Changes to this buffer's content will be visible in the new buffer, and vice versa; the two buffers' position, limit, and mark values will be independent.

The new buffer's capacity, limit, position, and mark values will be identical to those of this buffer. The new buffer will be read-only if, and only if, this buffer is read-only.

Returns

*a new long long **Buffer** (p. 582) which the caller owns.*

- virtual long long **get** ()

Relative get method.

Reads the value at this buffer's current position, and then increments the position.

Returns

the long long at the current position.

Exceptions

BufferUnderflow-Exception (p. 611)	<i>if there no more data to return.</i>
--	---

- virtual long long **get** (int index) const

Absolute get method.

Reads the value at the given index.

Parameters

index	<i>The index in the Buffer (p. 582) where the long long is to be read.</i>
-------	---

Returns

the long long that is located at the given index.

Exceptions

IndexOutOfBoundsException	<i>if index is not smaller than the buffer's limit, or index is negative.</i>
---------------------------	---

- virtual bool **hasArray** () const

Tells whether or not this buffer is backed by an accessible long long array.

If this method returns true then the array and arrayOffset methods may safely be invoked. Subclasses should override this method if they do not have a backing array as this class always returns true.

Returns

true if, and only if, this buffer is backed by an array and is not read-only.

- virtual bool **isReadOnly** () const

Tells whether or not this buffer is read-only.

Returns

true if, and only if, this buffer is read-only.

- virtual **LongBuffer & put** (long long value)

Writes the given long longs long onto this buffer at the current position, and then increments the position.

Parameters

value	<i>The long longs value to be written.</i>
-------	--

Returns

a reference to this buffer.

Exceptions

BufferOverflow-Exception (p. 609)	<i>if this buffer's current position is not smaller than its limit</i>
ReadOnlyBuffer-Exception (p. 2244)	<i>if this buffer is read-only</i>

- virtual **LongBuffer** & **put** (int index, long long value)

Writes the given long longs long longo this buffer at the given index.

Parameters

index	<i>The position in the Buffer (p. 582) to write the data</i>
value	<i>The long longs to write.</i>

Returns

a reference to this buffer.

Exceptions

IndexOutOfBoundsException	<i>if index greater than the buffer's limit minus the size of the type being written.</i>
ReadOnlyBuffer-Exception (p. 2244)	<i>if this buffer is read-only</i>

- virtual **LongBuffer** * **slice** () const

*Creates a new **LongBuffer** (p. 1752) whose content is a shared subsequence of this buffer's content.*

The content of the new buffer will start at this buffer's current position. Changes to this buffer's content will be visible in the new buffer, and vice versa; the two buffers' position, limit, and mark values will be independent.

The new buffer's position will be zero, its capacity and its limit will be the number of bytes remaining in this buffer, and its mark will be undefined. The new buffer will be read-only if, and only if, this buffer is read-only.

Returns

*the newly create **LongBuffer** (p. 1752) which the caller owns.*

Protected Member Functions

- virtual void **setReadOnly** (bool value)

*Sets this **LongArrayBuffer** (p. 1742) as Read-Only.*

6.328.1 Constructor & Destructor Documentation

6.328.1.1 **decaf::internal::nio::LongArrayBuffer::LongArrayBuffer** (int size, bool readOnly = false)

Creates a **IntArrayBuffer** (p. 1477) object that has its backing array allocated internally and is then owned and deleted when this object is deleted.

The array is initially created with all elements initialized to zero.

Parameters

<i>size</i>	The size of the array, this is the limit we read and write to.
<i>readOnly</i>	Boolean indicating if this buffer should be read-only, default as false.

Exceptions

<i>IllegalArgument-Exception</i>	if the capacity value is negative.
----------------------------------	------------------------------------

6.328.1.2 decaf::internal::nio::LongArrayBuffer::LongArrayBuffer (long long * array, int size, int offset, int length, bool readOnly = false)

Creates a **LongArrayBuffer** (p. 1742) object that wraps the given array.

If the own flag is set then it will delete this array when this object is deleted.

Parameters

<i>array</i>	The actual array to wrap.
<i>size</i>	The size of the given array.
<i>offset</i>	The position that is this buffers start position.
<i>length</i>	The limit of how many bytes into the array this Buffer can write.
<i>readOnly</i>	Boolean indicating if this buffer should be read-only, default as false.

Exceptions

<i>NullPointerException</i>	if buffer is NULL
<i>IndexOutOfBoundsException</i>	if offset is greater than array capacity.

6.328.1.3 decaf::internal::nio::LongArrayBuffer::LongArrayBuffer (const decaf::lang::Pointer< ByteArrayAdapter > & array, int offset, int length, bool readOnly = false)

Creates a byte buffer that wraps the passed ByteArrayAdapter and start at the given offset.

The capacity and limit of the new **LongArrayBuffer** (p. 1742) will be that of the remaining capacity of the passed buffer.

Parameters

<i>array</i>	The ByteArrayAdapter to wrap.
<i>offset</i>	The position that is this buffers start position.
<i>length</i>	The limit of how many bytes into the array this Buffer can write.
<i>readOnly</i>	Boolean indicating if this buffer should be read-only, default as false.

Exceptions

<i>NullPointerException</i>	if array is NULL
<i>IndexOutOfBoundsException</i>	if offset + length is greater than array size.

6.328.1.4 **decaf::internal::nio::LongArrayBuffer::LongArrayBuffer** (**const LongArrayBuffer & other**)

Create a **LongArrayBuffer** (p. 1742) that mirrors this one, meaning it shares a reference to this buffers ByteArrayAdapter and when changes are made to that data it is reflected in both.

Parameters

<i>other</i>	The LongArrayBuffer (p. 1742) this one is to mirror.
--------------	---

6.328.1.5 **virtual decaf::internal::nio::LongArrayBuffer::~~LongArrayBuffer** () [virtual]

6.328.2 Member Function Documentation

6.328.2.1 **virtual long long* decaf::internal::nio::LongArrayBuffer::array** () [virtual]

Returns the long long array that backs this buffer (optional operation).

Modifications to this buffer's content will cause the returned array's content to be modified, and vice versa.

Invoke the `hasArray` method before invoking this method in order to ensure that this buffer has an accessible backing array.

Returns

the array that backs this **Buffer** (p. 582).

Exceptions

<i>ReadOnlyBufferException</i> (p. 2244)	if this Buffer (p. 582) is read only.
<i>UnsupportedOperationException</i>	if the underlying store has no array.

Implements **decaf::nio::LongBuffer** (p. 1755).

6.328.2.2 virtual int **decaf::internal::nio::LongArrayBuffer::arrayOffset** ()
[virtual]

Returns the offset within this buffer's backing array of the first element of the buffer (optional operation).

Invoke the `hasArray` method before invoking this method in order to ensure that this buffer has an accessible backing array.

Returns

The offset long longo the backing array where index zero starts.

Exceptions

<i>ReadOnlyBufferException</i> (p. 2244)	if this Buffer (p. 582) is read only.
<i>UnsupportedOperationException</i>	if the underlying store has no array.

Implements **decaf::nio::LongBuffer** (p. 1756).

6.328.2.3 virtual **LongBuffer*** **decaf::internal::nio::LongArrayBuffer::asReadOnlyBuffer** () const [virtual]

Creates a new, read-only long long buffer that shares this buffer's content.

The content of the new buffer will be that of this buffer. Changes to this buffer's content will be visible in the new buffer; the new buffer itself, however, will be read-only and will not allow the shared content to be modified. The two buffers' position, limit, and mark values will be independent.

If this buffer is itself read-only then this method behaves in exactly the same way as the `duplicate` method.

The new buffer's capacity, limit, position, and mark values will be identical to those of this buffer.

Returns

The new, read-only long long buffer which the caller then owns.

Implements **decaf::nio::LongBuffer** (p. 1756).

6.328.2.4 virtual **LongBuffer&** **decaf::internal::nio::LongArrayBuffer::compact** ()
[virtual]

Compacts this buffer.

The bytes between the buffer's current position and its limit, if any, are copied to the beginning of the buffer. That is, the byte at index $p = \text{position}()$ (p. 587) is copied to index zero, the byte at index $p + 1$ is copied to index one, and so forth until the byte at index $\text{limit}()$ (p. 586) - 1 is copied to index $n = \text{limit}()$ (p. 586) - 1 - p . The buffer's position is then set to $n+1$ and its limit is set to its capacity. The mark, if defined, is discarded.

The buffer's position is set to the number of bytes copied, rather than to zero, so that an invocation of this method can be followed immediately by an invocation of another relative put method.

Returns

a reference to this **LongBuffer** (p. 1752).

Exceptions

<i>ReadOnlyBuffer-Exception</i> (p. 2244)	if this buffer is read-only.
---	------------------------------

Implements **decaf::nio::LongBuffer** (p. 1757).

6.328.2.5 `virtual LongBuffer* decaf::internal::nio::LongArrayBuffer::duplicate ()`
[virtual]

Creates a new long long buffer that shares this buffer's content.

The content of the new buffer will be that of this buffer. Changes to this buffer's content will be visible in the new buffer, and vice versa; the two buffers' position, limit, and mark values will be independent.

The new buffer's capacity, limit, position, and mark values will be identical to those of this buffer. The new buffer will be read-only if, and only if, this buffer is read-only.

Returns

a new long long **Buffer** (p. 582) which the caller owns.

Implements **decaf::nio::LongBuffer** (p. 1757).

6.328.2.6 `virtual long long decaf::internal::nio::LongArrayBuffer::get ()`
[virtual]

Relative get method.

Reads the value at this buffer's current position, and then increments the position.

Returns

the long long at the current position.

Exceptions

<i>BufferUnderflow-Exception</i> (p. 611)	if there no more data to return.
---	----------------------------------

Implements **decaf::nio::LongBuffer** (p. 1758).

6.328.2.7 `virtual long long decaf::internal::nio::LongArrayBuffer::get (int index) const`
[virtual]

Absolute get method.

Reads the value at the given index.

Parameters

<i>index</i>	The index in the Buffer (p. 582) where the long long is to be read.
--------------	--

Returns

the long long that is located at the given index.

Exceptions

<i>IndexOutOfBounds-Exception</i>	if index is not smaller than the buffer's limit, or index is negative.
--	--

Implements **decaf::nio::LongBuffer** (p. 1758).

6.328.2.8 `virtual bool decaf::internal::nio::LongArrayBuffer::hasArray () const`
[inline, virtual]

Tells whether or not this buffer is backed by an accessible long long array.

If this method returns true then the array and arrayOffset methods may safely be invoked. Subclasses should override this method if they do not have a backing array as this class always returns true.

Returns

true if, and only if, this buffer is backed by an array and is not read-only.

Implements **decaf::nio::LongBuffer** (p. 1760).

6.328.2.9 `virtual bool decaf::internal::nio::LongArrayBuffer::isReadOnly () const`
[inline, virtual]

Tells whether or not this buffer is read-only.

Returns

true if, and only if, this buffer is read-only.

Implements **decaf::nio::Buffer** (p. 586).

6.328.2.10 `virtual LongBuffer& decaf::internal::nio::LongArrayBuffer::put (long long value) [virtual]`

Writes the given long longs long longo this buffer at the current position, and then increments the position.

Parameters

<i>value</i>	The long longs value to be written.
--------------	-------------------------------------

Returns

a reference to this buffer.

Exceptions

<i>BufferOverflow-Exception</i> (p. 609)	if this buffer's current position is not smaller than its limit
<i>ReadOnlyBuffer-Exception</i> (p. 2244)	if this buffer is read-only

Implements **decaf::nio::LongBuffer** (p. 1762).

6.328.2.11 `virtual LongBuffer& decaf::internal::nio::LongArrayBuffer::put (int index, long long value) [virtual]`

Writes the given long longs long longo this buffer at the given index.

Parameters

<i>index</i>	The position in the Buffer (p. 582) to write the data
<i>value</i>	The long longs to write.

Returns

a reference to this buffer.

Exceptions

<i>IndexOutOfBoundsException</i>	if index greater than the buffer's limit minus the size of the type being written.
ReadOnlyBufferException (p. 2244)	if this buffer is read-only

Implements **decaf::nio::LongBuffer** (p. 1763).

6.328.2.12 virtual void **decaf::internal::nio::LongArrayBuffer::setReadOnly** (bool *value*) [inline, protected, virtual]

Sets this **LongArrayBuffer** (p. 1742) as Read-Only.

Parameters

<i>value</i>	Boolean value, true if this buffer is to be read-only, false otherwise.
--------------	---

6.328.2.13 virtual **LongBuffer*** **decaf::internal::nio::LongArrayBuffer::slice** ()
const [virtual]

Creates a new **LongBuffer** (p. 1752) whose content is a shared subsequence of this buffer's content.

The content of the new buffer will start at this buffer's current position. Changes to this buffer's content will be visible in the new buffer, and vice versa; the two buffers' position, limit, and mark values will be independent.

The new buffer's position will be zero, its capacity and its limit will be the number of bytes remaining in this buffer, and its mark will be undefined. The new buffer will be read-only if, and only if, this buffer is read-only.

Returns

the newly create **LongBuffer** (p. 1752) which the caller owns.

Implements **decaf::nio::LongBuffer** (p. 1763).

The documentation for this class was generated from the following file:

- src/main/decaf/internal/nio/**LongArrayBuffer.h**

6.329 decaf::nio::LongBuffer Class Reference

This class defines four categories of operations upon long long buffers:

```
#include <src/main/decaf/nio/LongBuffer.h>
```

Inheritance diagram for `decaf::nio::LongBuffer`:

Public Member Functions

- virtual `~LongBuffer ()`
- virtual `std::string toString () const`
- virtual `long long * array ()=0`
Returns the long long array that backs this buffer (optional operation).
- virtual `int arrayOffset ()=0`
Returns the offset within this buffer's backing array of the first element of the buffer (optional operation).
- virtual `LongBuffer * asReadOnlyBuffer () const =0`
Creates a new, read-only long long buffer that shares this buffer's content.
- virtual `LongBuffer & compact ()=0`
Compacts this buffer.
- virtual `LongBuffer * duplicate ()=0`
Creates a new long long buffer that shares this buffer's content.
- virtual `long long get ()=0`
Relative get method.
- virtual `long long get (int index) const =0`
Absolute get method.
- `LongBuffer & get (std::vector< long long > buffer)`
Relative bulk get method.
- `LongBuffer & get (long long *buffer, int size, int offset, int length)`
Relative bulk get method.
- virtual `bool hasArray () const =0`
Tells whether or not this buffer is backed by an accessible long long array.
- `LongBuffer & put (LongBuffer &src)`
This method transfers the long longs remaining in the given source buffer long longo this buffer.
- `LongBuffer & put (const long long *buffer, int size, int offset, int length)`
This method transfers long longs long longo this buffer from the given source array.
- `LongBuffer & put (std::vector< long long > &buffer)`
This method transfers the entire content of the given source long longs array long longo this buffer.
- virtual `LongBuffer & put (long long value)=0`
Writes the given long longs long longo this buffer at the current position, and then increments the position.
- virtual `LongBuffer & put (int index, long long value)=0`
Writes the given long longs long longo this buffer at the given index.
- virtual `LongBuffer * slice () const =0`

*Creates a new **LongBuffer** (p. 1752) whose content is a shared subsequence of this buffer's content.*

- virtual int **compareTo** (const **LongBuffer** &value) const
- virtual bool **equals** (const **LongBuffer** &value) const
- virtual bool **operator==** (const **LongBuffer** &value) const
- virtual bool **operator<** (const **LongBuffer** &value) const

Static Public Member Functions

- static **LongBuffer** * **allocate** (int capacity)
Allocates a new Double buffer.
- static **LongBuffer** * **wrap** (long long *array, int size, int offset, int length)
*Wraps the passed buffer with a new **LongBuffer** (p. 1752).*
- static **LongBuffer** * **wrap** (std::vector< long long > &buffer)
*Wraps the passed STL long long Vector in a **LongBuffer** (p. 1752).*

Protected Member Functions

- **LongBuffer** (int capacity)
*Creates a **LongBuffer** (p. 1752) object that has its backing array allocated long long-
nally and is then owned and deleted when this object is deleted.*

6.329.1 Detailed Description

This class defines four categories of operations upon long long buffers:

o Absolute and relative get and put methods that read and write single long longs;
o Relative bulk get methods that transfer contiguous sequences of long longs from this
buffer long longo an array; and
o Relative bulk put methods that transfer contiguous
sequences of long longs from a long long array or some other long long buffer long
longo this buffer
o Methods for compacting, duplicating, and slicing a long long buffer.

Double buffers can be created either by allocation, which allocates space for the buffer's
content, by wrapping an existing long long array long longo a buffer, or by creating a
view of an existing byte buffer

Methods in this class that do not otherwise have a value to return are specified to return
the buffer upon which they are invoked. This allows method invocations to be chained.

6.329.2 Constructor & Destructor Documentation

6.329.2.1 `decaf::nio::LongBuffer::LongBuffer (int capacity)` `[protected]`

Creates a **LongBuffer** (p. 1752) object that has its backing array allocated long longer-nally and is then owned and deleted when this object is deleted.

The array is initially created with all elements initialized to zero.

Parameters

<i>capacity</i>	The size and limit of the Buffer (p. 582) in long longs.
-----------------	---

Exceptions

<i>IllegalArgument-Exception</i>	if capacity is negative.
----------------------------------	--------------------------

6.329.2.2 `virtual decaf::nio::LongBuffer::~~LongBuffer ()` `[inline, virtual]`

6.329.3 Member Function Documentation

6.329.3.1 `static LongBuffer* decaf::nio::LongBuffer::allocate (int capacity)` `[static]`

Allocates a new Double buffer.

The new buffer's position will be zero, its limit will be its capacity, and its mark will be undefined. It will have a backing array, and its array offset will be zero.

Parameters

<i>capacity</i>	The size of the Double buffer in long longs.
-----------------	--

Returns

the **LongBuffer** (p. 1752) that was allocated, caller owns.

6.329.3.2 `virtual long long* decaf::nio::LongBuffer::array ()` `[pure virtual]`

Returns the long long array that backs this buffer (optional operation).

Modifications to this buffer's content will cause the returned array's content to be modified, and vice versa.

Invoke the `hasArray` method before invoking this method in order to ensure that this buffer has an accessible backing array.

Returns

the array that backs this **Buffer** (p. 582).

Exceptions

<i>ReadOnlyBuffer-Exception</i> (p. 2244)	if this Buffer (p. 582) is read only.
<i>Unsupported-OperationException</i>	if the underlying store has no array.

Implemented in **decaf::internal::nio::LongArrayBuffer** (p. 1747).

6.329.3.3 virtual int decaf::nio::LongBuffer::arrayOffset () [pure virtual]

Returns the offset within this buffer's backing array of the first element of the buffer (optional operation).

Invoke the `hasArray` method before invoking this method in order to ensure that this buffer has an accessible backing array.

Returns

The offset long longo the backing array where index zero starts.

Exceptions

<i>ReadOnlyBuffer-Exception</i> (p. 2244)	if this Buffer (p. 582) is read only.
<i>Unsupported-OperationException</i>	if the underlying store has no array.

Implemented in **decaf::internal::nio::LongArrayBuffer** (p. 1748).

6.329.3.4 virtual LongBuffer* decaf::nio::LongBuffer::asReadOnlyBuffer () const [pure virtual]

Creates a new, read-only long long buffer that shares this buffer's content.

The content of the new buffer will be that of this buffer. Changes to this buffer's content will be visible in the new buffer; the new buffer itself, however, will be read-only and will not allow the shared content to be modified. The two buffers' position, limit, and mark values will be independent.

If this buffer is itself read-only then this method behaves in exactly the same way as the `duplicate` method.

The new buffer's capacity, limit, position, and mark values will be identical to those of

this buffer.

Returns

The new, read-only long long buffer which the caller then owns.

Implemented in **decaf::internal::nio::LongArrayBuffer** (p. 1748).

6.329.3.5 `virtual LongBuffer& decaf::nio::LongBuffer::compact () [pure virtual]`

Compacts this buffer.

The bytes between the buffer's current position and its limit, if any, are copied to the beginning of the buffer. That is, the byte at index $p = \text{position}()$ (p. 587) is copied to index zero, the byte at index $p + 1$ is copied to index one, and so forth until the byte at index $\text{limit}()$ (p. 586) - 1 is copied to index $n = \text{limit}()$ (p. 586) - 1 - p . The buffer's position is then set to $n + 1$ and its limit is set to its capacity. The mark, if defined, is discarded.

The buffer's position is set to the number of bytes copied, rather than to zero, so that an invocation of this method can be followed immediately by an invocation of another relative put method.

Returns

a reference to this **LongBuffer** (p. 1752).

Exceptions

<i>ReadOnlyBuffer-Exception</i> (p. 2244)	if this buffer is read-only.
---	------------------------------

Implemented in **decaf::internal::nio::LongArrayBuffer** (p. 1748).

6.329.3.6 `virtual int decaf::nio::LongBuffer::compareTo (const LongBuffer & value) const [virtual]`

6.329.3.7 `virtual LongBuffer* decaf::nio::LongBuffer::duplicate () [pure virtual]`

Creates a new long long buffer that shares this buffer's content.

The content of the new buffer will be that of this buffer. Changes to this buffer's content will be visible in the new buffer, and vice versa; the two buffers' position, limit, and mark values will be independent.

The new buffer's capacity, limit, position, and mark values will be identical to those of this buffer. The new buffer will be read-only if, and only if, this buffer is read-only.

Returns

a new long long **Buffer** (p. 582) which the caller owns.

Implemented in **decaf::internal::nio::LongArrayBuffer** (p. 1749).

6.329.3.8 virtual bool decaf::nio::LongBuffer::equals (const LongBuffer & value)
const [virtual]

6.329.3.9 virtual long long decaf::nio::LongBuffer::get () [pure virtual]

Relative get method.

Reads the value at this buffer's current position, and then increments the position.

Returns

the long long at the current position.

Exceptions

BufferUnderflow-Exception (p. 611)	if there no more data to return.
--	----------------------------------

Implemented in **decaf::internal::nio::LongArrayBuffer** (p. 1749).

6.329.3.10 virtual long long decaf::nio::LongBuffer::get (int index) const [pure virtual]

Absolute get method.

Reads the value at the given index.

Parameters

<i>index</i>	The index in the Buffer (p. 582) where the long long is to be read.
--------------	--

Returns

the long long that is located at the given index.

Exceptions

IndexOutOfBounds-Exception	if index is not smaller than the buffer's limit, or index is negative.
-----------------------------------	--

Implemented in **decaf::internal::nio::LongArrayBuffer** (p. 1750).

6.329.3.11 LongBuffer& decaf::nio::LongBuffer::get (std::vector< long long > *buffer*)

Relative bulk get method.

This method transfers values from this buffer long longo the given destination vector. An invocation of this method of the form `src.get(a)` behaves in exactly the same way as the invocation. The vector must be sized to the amount of data that is to be read, that is to say, the caller should call `buffer.resize(N)` before calling this get method.

Returns

a reference to this **Buffer** (p. 582).

Exceptions

BufferUnderflow-Exception (p. 611)	if there are fewer than length long longs remaining in this buffer.
--	---

6.329.3.12 LongBuffer& decaf::nio::LongBuffer::get (long long * *buffer*, int *size*, int *offset*, int *length*)

Relative bulk get method.

This method transfers long longs from this buffer long longo the given destination array. If there are fewer long longs remaining in the buffer than are required to satisfy the request, that is, if `length > remaining()` (p. 588), then no bytes are transferred and a **BufferUnderflowException** (p. 611) is thrown.

Otherwise, this method copies length long longs from this buffer long longo the given array, starting at the current position of this buffer and at the given offset in the array. The position of this buffer is then incremented by length.

Parameters

<i>buffer</i>	The pointer to an allocated long long buffer to fill.
<i>size</i>	The size of the passed in buffer.
<i>offset</i>	The position in the buffer to start filling.
<i>length</i>	The amount of data to put in the passed buffer.

Returns

a reference to this **Buffer** (p. 582).

Exceptions

BufferUnderflow-Exception (p. 611)	if there are fewer than length long longs remaining in this buffer
--	--

<i>NullPolong</i>	longerException if the passed buffer is null.
<i>IndexOutOfBounds-Exception</i>	if the preconditions of size, offset, or length are not met.

6.329.3.13 `virtual bool decaf::nio::LongBuffer::hasArray () const` [pure virtual]

Tells whether or not this buffer is backed by an accessible long long array.

If this method returns true then the array and arrayOffset methods may safely be invoked. Subclasses should override this method if they do not have a backing array as this class always returns true.

Returns

true if, and only if, this buffer is backed by an array and is not read-only.

Implemented in **decaf::internal::nio::LongArrayBuffer** (p. 1750).

6.329.3.14 `virtual bool decaf::nio::LongBuffer::operator< (const LongBuffer & value) const` [virtual]

6.329.3.15 `virtual bool decaf::nio::LongBuffer::operator== (const LongBuffer & value) const` [virtual]

6.329.3.16 `LongBuffer& decaf::nio::LongBuffer::put (LongBuffer & src)`

This method transfers the long longs remaining in the given source buffer long longo this buffer.

If there are more long longs remaining in the source buffer than in this buffer, that is, if `src.remaining() > remaining()` (p. 588), then no long longs are transferred and a **BufferOverflowException** (p. 609) is thrown.

Otherwise, this method copies `n = src.remaining()` long longs from the given buffer long longo this buffer, starting at each buffer's current position. The positions of both buffers are then incremented by `n`.

Parameters

<i>src</i>	The buffer to take long longs from an place in this one.
------------	--

Returns

a reference to this buffer.

Exceptions

BufferOverflow-Exception (p. 609)	if there is insufficient space in this buffer for the remaining long longs in the source buffer
<i>IllegalArgument-Exception</i>	if the source buffer is this buffer
ReadOnlyBuffer-Exception (p. 2244)	if this buffer is read-only

6.329.3.17 LongBuffer& decaf::nio::LongBuffer::put (const long long * *buffer*, int *size*, int *offset*, int *length*)

This method transfers long longs long longo this buffer from the given source array.

If there are more long longs to be copied from the array than remain in this buffer, that is, if `length > remaining()` (p. 588), then no long longs are transferred and a **BufferOverflowException** (p. 609) is thrown.

Otherwise, this method copies `length` bytes from the given array long longo this buffer, starting at the given `offset` in the array and at the current position of this buffer. The position of this buffer is then incremented by `length`.

Parameters

<i>buffer</i>	The array from which long longs are to be read.
<i>size</i>	The size of the buffer passed.
<i>offset</i>	The offset within the array of the first char to be read.
<i>length</i>	The number of long longs to be read from the given array.

Returns

a reference to this buffer.

Exceptions

BufferOverflow-Exception (p. 609)	if there is insufficient space in this buffer
ReadOnlyBuffer-Exception (p. 2244)	if this buffer is read-only
<i>NullPolong</i>	longerException if the passed buffer is null.
<i>IndexOutOfBounds-Exception</i>	if the preconditions of <code>size</code> , <code>offset</code> , or <code>length</code> are not met.

6.329.3.18 LongBuffer& decaf::nio::LongBuffer::put (std::vector< long long > & *buffer*)

This method transfers the entire content of the given source long longs array long longo this buffer.

This is the same as calling put(&buffer[0], 0, buffer.size()).

Parameters

<i>buffer</i>	The buffer whose contents are copied to this LongBuffer (p. 1752).
---------------	---

Returns

a reference to this buffer.

Exceptions

BufferOverflow-Exception (p. 609)	if there is insufficient space in this buffer.
ReadOnlyBuffer-Exception (p. 2244)	if this buffer is read-only.

6.329.3.19 virtual LongBuffer& decaf::nio::LongBuffer::put (long long *value*) [pure virtual]

Writes the given long longs long longo this buffer at the current position, and then increments the position.

Parameters

<i>value</i>	The long longs value to be written.
--------------	-------------------------------------

Returns

a reference to this buffer.

Exceptions

BufferOverflow-Exception (p. 609)	if this buffer's current position is not smaller than its limit
ReadOnlyBuffer-Exception (p. 2244)	if this buffer is read-only

Implemented in **decaf::internal::nio::LongArrayBuffer** (p. 1751).

6.329.3.20 `virtual LongBuffer& decaf::nio::LongBuffer::put (int index, long long value) [pure virtual]`

Writes the given long longs long longo this buffer at the given index.

Parameters

<i>index</i>	The position in the Buffer (p. 582) to write the data
<i>value</i>	The long longs to write.

Returns

a reference to this buffer.

Exceptions

<i>IndexOutOfBoundsException</i>	if index greater than the buffer's limit minus the size of the type being written.
ReadOnlyBuffer-Exception (p. 2244)	if this buffer is read-only

Implemented in **decaf::internal::nio::LongArrayBuffer** (p. 1751).

6.329.3.21 `virtual LongBuffer* decaf::nio::LongBuffer::slice () const [pure virtual]`

Creates a new **LongBuffer** (p. 1752) whose content is a shared subsequence of this buffer's content.

The content of the new buffer will start at this buffer's current position. Changes to this buffer's content will be visible in the new buffer, and vice versa; the two buffers' position, limit, and mark values will be independent.

The new buffer's position will be zero, its capacity and its limit will be the number of bytes remaining in this buffer, and its mark will be undefined. The new buffer will be read-only if, and only if, this buffer is read-only.

Returns

the newly create **LongBuffer** (p. 1752) which the caller owns.

Implemented in **decaf::internal::nio::LongArrayBuffer** (p. 1752).

6.329.3.22 `virtual std::string decaf::nio::LongBuffer::toString () const [virtual]`

Returns

a std::string describing this object

6.329.3.23 static **LongBuffer*** **decaf::nio::LongBuffer::wrap** (long long * *array*, int *size*, int *offset*, int *length*) [static]

Wraps the passed buffer with a new **LongBuffer** (p. 1752).

The new buffer will be backed by the given long long array; that is, modifications to the buffer will cause the array to be modified and vice versa. The new buffer's capacity will be array.length, its position will be offset, its limit will be offset + length, and its mark will be undefined. Its backing array will be the given array, and its array offset will be zero.

Parameters

<i>array</i>	The array that will back the new buffer.
<i>size</i>	The size of the passed in array.
<i>offset</i>	The offset of the subarray to be used.
<i>length</i>	The length of the subarray to be used.

Returns

a new **LongBuffer** (p. 1752) that is backed by buffer, caller owns.

Exceptions

<i>NullPointerException</i>	if the array pointer is NULL.
<i>IndexOutOfBoundsException</i>	if the preconditions of size, offset, or length are not met.

6.329.3.24 static **LongBuffer*** **decaf::nio::LongBuffer::wrap** (std::vector< long long > & *buffer*) [static]

Wraps the passed STL long long Vector in a **LongBuffer** (p. 1752).

The new buffer will be backed by the given long long array; modifications to the buffer will cause the array to be modified and vice versa. The new buffer's capacity and limit will be buffer.size(), its position will be zero, and its mark will be undefined. Its backing array will be the given array, and its array offset will be zero.

Parameters

<i>buffer</i>	The vector that will back the new buffer, the vector must have been sized to the desired size already by calling vector.resize(N).
---------------	--

Returns

a new **LongBuffer** (p. 1752) that is backed by buffer, caller owns.

The documentation for this class was generated from the following file:

- src/main/decaf/nio/**LongBuffer.h**

6.330 activemq::util::LongSequenceGenerator Class Reference

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

```
#include <src/main/activemq/util/LongSequenceGenerator.h>
```

Public Member Functions

- **LongSequenceGenerator** ()
- virtual **~LongSequenceGenerator** ()
- long long **getNextSequenceld** ()
- long long **getLastSequenceld** ()

6.330.1 Detailed Description

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

This class is thread safe so the ids can be requested in different threads safely.

6.330.2 Constructor & Destructor Documentation

6.330.2.1 **activemq::util::LongSequenceGenerator::LongSequenceGenerator** ()

6.330.2.2 **virtual activemq::util::LongSequenceGenerator::~~LongSequenceGenerator** () [inline, virtual]

6.330.3 Member Function Documentation

6.330.3.1 **long long activemq::util::LongSequenceGenerator::getLastSequenceld** ()

Returns

the last id that was generated.

6.330.3.2 long long activemq::util::LongSequenceGenerator::getNextSequenceld ()

Returns

the next id in the sequence.

The documentation for this class was generated from the following file:

- src/main/activemq/util/**LongSequenceGenerator.h**

6.331 decaf::net::MalformedURLException Class Reference

```
#include <src/main/decaf/net/MalformedURLException.h>
```

Inheritance diagram for decaf::net::MalformedURLException:

Public Member Functions

- **MalformedURLException** () throw ()
Default Constructor.
- **MalformedURLException** (const **Exception** &ex) throw ()
Conversion Constructor from some other Exception.
- **MalformedURLException** (const **MalformedURLException** &ex) throw ()
Copy Constructor.
- **MalformedURLException** (const char *file, const int lineNumber, const std::exception *cause, const char *msg,...) throw ()
Constructor - Initializes the file name and line number where this message occurred.
- **MalformedURLException** (const std::exception *cause) throw ()
Constructor.
- **MalformedURLException** (const char *file, const int lineNumber, const char *msg,...) throw ()
Constructor - Initializes the file name and line number where this message occurred.
- virtual **MalformedURLException** * **clone** () const
Clones this exception.
- virtual ~**MalformedURLException** () throw ()

6.331.1 Constructor & Destructor Documentation

6.331.1.1 decaf::net::MalformedURLException::MalformedURLException ()
throw () [inline]

Default Constructor.

6.331.1.2 **decaf::net::MalformedURLException::MalformedURLException (const Exception & ex) throw ()** [inline]

Conversion Constructor from some other Exception.

Parameters

<i>ex</i>	An exception that should become this type of Exception
-----------	--

6.331.1.3 **decaf::net::MalformedURLException::MalformedURLException (const MalformedURLException & ex) throw ()** [inline]

Copy Constructor.

Parameters

<i>ex</i>	An exception that should become this type of Exception
-----------	--

6.331.1.4 **decaf::net::MalformedURLException::MalformedURLException (const char * file, const int lineNumber, const std::exception * cause, const char * msg, ...) throw ()** [inline]

Constructor - Initializes the file name and line number where this message occurred.

Sets the message to report, using an optional list of arguments to parse into the message

Parameters

<i>file</i>	The file name where exception occurs
<i>lineNumber</i>	The line number where the exception occurred.
<i>cause</i>	The exception that was the cause for this one to be thrown.
<i>msg</i>	The message to report
<i>...</i>	list of primitives that are formatted into the message

6.331.1.5 **decaf::net::MalformedURLException::MalformedURLException (const std::exception * cause) throw ()** [inline]

Constructor.

Parameters

<i>cause</i>	Pointer to the exception that caused this one to be thrown, the object is cloned caller retains ownership.
--------------	--

6.332 decaf::util::Map< K, V, COMPARATOR > Class Template Reference 1773

6.331.1.6 **decaf::net::MalformedURLException::MalformedURLException** (const char * *file*, const int *lineNumber*, const char * *msg*, ...) throw () [inline]

Constructor - Initializes the file name and line number where this message occurred.

Sets the message to report, using an optional list of arguments to parse into the message

Parameters

<i>file</i>	The file name where exception occurs
<i>lineNumber</i>	The line number where the exception occurred.
<i>msg</i>	The message to report
...	list of primitives that are formatted into the message

6.331.1.7 **virtual decaf::net::MalformedURLException::~~MalformedURLException** () throw () [inline, virtual]

6.331.2 Member Function Documentation

6.331.2.1 **virtual MalformedURLException* decaf::net::MalformedURLException::clone** () const [inline, virtual]

Clones this exception.

This is useful for cases where you need to preserve the type of the original exception as well as the message. All subclasses should override.

Returns

a new Exception instance that is a copy of this Exception object.

Reimplemented from **decaf::io::IOException** (p. 1547).

The documentation for this class was generated from the following file:

- src/main/decaf/net/**MalformedURLException.h**

6.332 decaf::util::Map< K, V, COMPARATOR > Class Template - Reference

Map (p. 1768) template that wraps around a std::map to provide a more user-friendly interface and to provide common functions that do not exist in std::map.

```
#include <src/main/decaf/util/Map.h>
```

Inheritance diagram for decaf::util::Map< K, V, COMPARATOR >:

Data Structures

- class **Entry**

Public Member Functions

- **Map** ()
Default constructor - does nothing.
- virtual **~Map** ()
- virtual bool **equals** (const **Map** &source) const =0
Comparison, equality is dependent on the method of determining if the element are equal.
- virtual void **copy** (const **Map** &source)=0
Copies the content of the source map into this map.
- virtual void **clear** ()=0
Removes all keys and values from this map.
- virtual bool **containsKey** (const K &key) const =0
Indicates whether or this map contains a value for the given key.
- virtual bool **containsValue** (const V &value) const =0
Indicates whether or this map contains a value for the given value, i.e.
- virtual bool **isEmpty** () const =0
- virtual int **size** () const =0
- virtual V & **get** (const K &key)=0
*Gets the value mapped to the specified key in the **Map** (p. 1768).*
- virtual const V & **get** (const K &key) const =0
*Gets the value mapped to the specified key in the **Map** (p. 1768).*
- virtual void **put** (const K &key, const V &value)=0
Sets the value for the specified key.
- virtual void **putAll** (const **Map**< K, V, COMPARATOR > &other)=0
*Stores a copy of the Mappings contained in the other **Map** (p. 1768) in this one.*
- virtual V **remove** (const K &key)=0
Removes the value (key/value pair) for the specified key from the map, returns a copy of the value that was mapped to the key.
- virtual std::vector< K > **keySet** () const =0
*Returns a **Set** (p. 2397) view of the mappings contained in this map.*
- virtual std::vector< V > **values** () const =0

6.332.1 Detailed Description

template<typename K, typename V, typename COMPARATOR = std::less<K>> class decaf::util::Map< K, V, COMPARATOR >

Map (p. 1768) template that wraps around a std::map to provide a more user-friendly interface and to provide common functions that do not exist in std::map.

6.332.2 Constructor & Destructor Documentation

6.332.2.1 `template<typename K, typename V, typename COMPARATOR = std::less<K>>
decaf::util::Map< K, V, COMPARATOR >::Map () [inline]`

Default constructor - does nothing.

6.332.2.2 `template<typename K, typename V, typename COMPARATOR = std::less<K>>
virtual decaf::util::Map< K, V, COMPARATOR >::~~Map () [inline,
virtual]`

6.332.3 Member Function Documentation

6.332.3.1 `template<typename K, typename V, typename COMPARATOR = std::less<K>>
virtual void decaf::util::Map< K, V, COMPARATOR >::clear () [pure
virtual]`

Removes all keys and values from this map.

Exceptions

<i>Unsupported- OperationException</i>	if this map is unmodifiable.
--	------------------------------

Implemented in `decaf::util::concurrent::ConcurrentStlMap< K, V, COMPARATOR >` (p.910), `decaf::util::concurrent::ConcurrentStlMap< Pointer< MessageId >, Pointer< Message >, MessageId::COMPARATOR >` (p.910), `decaf::util::concurrent::ConcurrentStlMap< Pointer< ConnectionId >, Pointer< ConnectionState >, ConnectionId::COMPARATOR >` (p.910), `decaf::util::concurrent::ConcurrentStlMap< Pointer< ConsumerId >, Pointer< ConsumerState >, ConsumerId::COMPARATOR >` (p.910), `decaf::util::concurrent::ConcurrentStlMap< Pointer< SessionId >, Pointer< SessionState >, SessionId::COMPARATOR >` (p.910), `decaf::util::concurrent::ConcurrentStlMap< Pointer< LocalTransactionId >, Pointer< TransactionState >, LocalTransactionId::COMPARATOR >` (p.910), `decaf::util::concurrent::ConcurrentStlMap< Pointer< ProducerId >, Pointer< ProducerState >, ProducerId::COMPARATOR >` (p.910), `decaf::util::StlMap< K, V, COMPARATOR >` (p.2557), `decaf::util::StlMap< cms::Session *, SessionResolver * >` (p.2557), `decaf::util::StlMap< std::string, WireFormatFactory * >` (p.2557), `decaf::util::StlMap< decaf::lang::Runnable *, decaf::util::TimerTask * >` (p.2557), `decaf::util::StlMap< std::string, PrimitiveValueNode >` (p.2557), `decaf::util::StlMap< std::string, cms::Queue * >` (p.2557), `decaf::util::StlMap< std::string, CachedConsumer * >` (p.2557), `decaf::util::StlMap< Pointer< commands::ConsumerId >, ActiveMQConsumer *, commands::ConsumerId::COMPARATOR >` (p.2557), `decaf::util::StlMap< std::string, TransportFactory * >` (p.2557), `decaf::util::StlMap< Pointer< ConsumerId >, Pointer< ConsumerInfo >, ConsumerId::COMPARATOR >` (p.2557), `decaf::util::StlMap< int, Pointer< Command > >` (p.2557), `decaf::util::StlMap< std::string, CachedProducer * >` (p.2557), and `decaf::util::StlMap< std::string, cms::Topic * >` (p.2557).

```
6.332.3.2  template<typename K, typename V, typename COMPARATOR = std::less<K>>
            virtual bool decaf::util::Map< K, V, COMPARATOR >::containsKey ( const K &
            key ) const [pure virtual]
```

Indicates whether or this map contains a value for the given key.

Parameters

<i>key</i>	The key to look up.
------------	---------------------

Returns

true if this map contains the value, otherwise false.

Implemented in **decaf::util::concurrent::ConcurrentStlMap< K, V, COMPARATOR >** (p. 910), **decaf::util::StlMap< K, V, COMPARATOR >** (p. 2557), **decaf::util::StlMap< cms::Session *, SessionResolver * >** (p. 2557), **decaf::util::StlMap< std::string, WireFormatFactory * >** (p. 2557), **decaf::util::StlMap< decaf::lang::Runnable *, decaf::util::TimerTask * >** (p. 2557), **decaf::util::StlMap< std::string, PrimitiveValueNode >** (p. 2557), **decaf::util::StlMap< std::string, cms::Queue * >** (p. 2557), **decaf::util::StlMap< std::string, CachedConsumer * >** (p. 2557), **decaf::util::StlMap< Pointer< commands::ConsumerId >, ActiveMQConsumer *, commands::ConsumerId::COMPARATOR >** (p. 2557), **decaf::util::StlMap< std::string, TransportFactory * >** (p. 2557), **decaf::util::StlMap< Pointer< ConsumerId >, Pointer< ConsumerInfo >, ConsumerId::COMPARATOR >** (p. 2557), **decaf::util::StlMap< int, Pointer< Command > >** (p. 2557), **decaf::util::StlMap< std::string, CachedProducer * >** (p. 2557), and **decaf::util::StlMap< std::string, cms::Topic * >** (p. 2557).

```
6.332.3.3  template<typename K, typename V, typename COMPARATOR = std::less<K>>
            virtual bool decaf::util::Map< K, V, COMPARATOR >::containsValue ( const V
            & value ) const [pure virtual]
```

Indicates whether or this map contains a value for the given value, i.e.

they are equal, this is done by operator== so the types must pass equivalence testing in this manner.

Parameters

<i>value</i>	The Value to look up.
--------------	-----------------------

Returns

true if this map contains the value, otherwise false.

Implemented in **decaf::util::concurrent::ConcurrentStlMap< K, V, COMPARATOR >** (p. 910), **decaf::util::StlMap< K, V, COMPARATOR >** (p. 2557), **decaf::util::StlMap< cms::Session *, SessionResolver * >** (p. 2557), **decaf::util::StlMap<**

std::string, WireFormatFactory * > (p. 2557), decaf::util::StlMap< decaf::lang::Runnable *, decaf::util::TimerTask * > (p. 2557), decaf::util::StlMap< std::string, PrimitiveValueNode > (p. 2557), decaf::util::StlMap< std::string, cms::Queue * > (p. 2557), decaf::util::StlMap< std::string, CachedConsumer * > (p. 2557), decaf::util::StlMap< Pointer< commands::ConsumerId >, ActiveMQConsumer *, commands::ConsumerId::COMPARATOR > (p. 2557), decaf::util::StlMap< std::string, TransportFactory * > (p. 2557), decaf::util::StlMap< Pointer< ConsumerId >, Pointer< ConsumerInfo >, ConsumerId::COMPARATOR > (p. 2557), decaf::util::StlMap< int, Pointer< Command > > (p. 2557), decaf::util::StlMap< std::string, CachedProducer * > (p. 2557), and decaf::util::StlMap< std::string, cms::Topic * > (p. 2557).

6.332.3.4 `template<typename K, typename V, typename COMPARATOR = std::less<K>>
virtual void decaf::util::Map< K, V, COMPARATOR >::copy (const Map< K, V,
COMPARATOR > & source) [pure virtual]`

Copies the content of the source map into this map.

Erases all existing data in this map.

Parameters

<i>source</i>	The source object to copy from.
---------------	---------------------------------

Implemented in **decaf::util::concurrent::ConcurrentStlMap< K, V, COMPARATOR >** (p. 911), and **decaf::util::StlMap< K, V, COMPARATOR >** (p. 2558).

6.332.3.5 `template<typename K, typename V, typename COMPARATOR = std::less<K>>
virtual bool decaf::util::Map< K, V, COMPARATOR >::equals (const Map< K,
V, COMPARATOR > & source) const [pure virtual]`

Comparison, equality is dependent on the method of determining if the element are equal.

Parameters

<i>source</i>	- Map (p. 1768) to compare to this one.
---------------	--

Returns

true if the **Map** (p. 1768) passed is equal in value to this one.

Implemented in **decaf::util::concurrent::ConcurrentStlMap< K, V, COMPARATOR >** (p. 911), and **decaf::util::StlMap< K, V, COMPARATOR >** (p. 2558).

```
6.332.3.6  template<typename K, typename V, typename COMPARATOR = std::less<K>>
            virtual V& decaf::util::Map< K, V, COMPARATOR >::get ( const K & key )
            [pure virtual]
```

Gets the value mapped to the specified key in the **Map** (p. 1768).

If there is no element in the map whose key is equivalent to the key provided then a **NoSuchElementException** (p. 1984) is thrown.

Parameters

<i>key</i>	The search key.
------------	-----------------

Returns

A reference to the value for the given key.

Exceptions

NoSuchElementException (p. 1984)	if the key requests doesn't exist in the Map (p. 1768).
--	--

Implemented in **decaf::util::concurrent::ConcurrentStlMap< K, V, COMPARATOR >** (p. 912), **decaf::util::StlMap< K, V, COMPARATOR >** (p. 2559), **decaf::util::StlMap< cms::Session *, SessionResolver * >** (p. 2559), **decaf::util::StlMap< std::string, WireFormatFactory * >** (p. 2559), **decaf::util::StlMap< decaf::lang::Runnable *, decaf::util::TimerTask * >** (p. 2559), **decaf::util::StlMap< std::string, PrimitiveValueNode >** (p. 2559), **decaf::util::StlMap< std::string, cms::Queue * >** (p. 2559), **decaf::util::StlMap< std::string, CachedConsumer * >** (p. 2559), **decaf::util::StlMap< Pointer< commands::ConsumerId >, ActiveMQConsumer *, commands::ConsumerId::COMPARATOR >** (p. 2559), **decaf::util::StlMap< std::string, TransportFactory * >** (p. 2559), **decaf::util::StlMap< Pointer< ConsumerId >, Pointer< ConsumerInfo >, ConsumerId::COMPARATOR >** (p. 2559), **decaf::util::StlMap< int, Pointer< Command > >** (p. 2559), **decaf::util::StlMap< std::string, CachedProducer * >** (p. 2559), and **decaf::util::StlMap< std::string, cms::Topic * >** (p. 2559).

Referenced by **decaf::util::StlMap< std::string, cms::Topic * >::equals()**, **decaf::util::concurrent::ConcurrentStlMap< Pointer< ProducerId >, Pointer< ProducerState >, ProducerId::COMPARATOR >::equals()**, **decaf::util::StlMap< std::string, cms::Topic * >::putAll()**, and **decaf::util::concurrent::ConcurrentStlMap< Pointer< ProducerId >, Pointer< ProducerState >, ProducerId::COMPARATOR >::putAll()**.

6.332 decaf::util::Map< K, V, COMPARATOR > Class Template Reference 1779

```
6.332.3.7  template<typename K, typename V, typename COMPARATOR = std::less<K>>
            virtual const V& decaf::util::Map< K, V, COMPARATOR >::get ( const K & key )
            const [pure virtual]
```

Gets the value mapped to the specified key in the **Map** (p. 1768).

If there is no element in the map whose key is equivalent to the key provided then a **NoSuchElementException** (p. 1984) is thrown.

Parameters

<i>key</i>	The search key.
------------	-----------------

Returns

A {const} reference to the value for the given key.

Exceptions

NoSuchElementException (p. 1984)	if the key requests doesn't exist in the Map (p. 1768).
--	--

Implemented in **decaf::util::concurrent::ConcurrentStlMap< K, V, COMPARATOR >** (p. 912), **decaf::util::StlMap< K, V, COMPARATOR >** (p. 2559), **decaf::util::StlMap< cms::Session *, SessionResolver * >** (p. 2559), **decaf::util::StlMap< std::string, WireFormatFactory * >** (p. 2559), **decaf::util::StlMap< decaf::lang::Runnable *, decaf::util::TimerTask * >** (p. 2559), **decaf::util::StlMap< std::string, PrimitiveValueNode >** (p. 2559), **decaf::util::StlMap< std::string, cms::Queue * >** (p. 2559), **decaf::util::StlMap< std::string, CachedConsumer * >** (p. 2559), **decaf::util::StlMap< Pointer< commands::ConsumerId >, ActiveMQConsumer *, commands::ConsumerId::COMPARATOR >** (p. 2559), **decaf::util::StlMap< std::string, TransportFactory * >** (p. 2559), **decaf::util::StlMap< Pointer< ConsumerId >, Pointer< ConsumerInfo >, ConsumerId::COMPARATOR >** (p. 2559), **decaf::util::StlMap< int, Pointer< Command > >** (p. 2559), **decaf::util::StlMap< std::string, CachedProducer * >** (p. 2559), and **decaf::util::StlMap< std::string, cms::Topic * >** (p. 2559).

```
6.332.3.8  template<typename K, typename V, typename COMPARATOR = std::less<K>>
            virtual bool decaf::util::Map< K, V, COMPARATOR >::isEmpty ( ) const
            [pure virtual]
```

Returns

if the **Map** (p. 1768) contains any element or not, TRUE or FALSE

Implemented in **decaf::util::concurrent::ConcurrentStlMap< K, V, COMPARATOR >** (p. 913), **decaf::util::concurrent::ConcurrentStlMap< Pointer< MessageId >, Pointer< Message >, MessageId::COMPARATOR >** (p. 913), **decaf::util::concurrent::ConcurrentStlMap< Pointer< ConnectionId >, Pointer<**

ConnectionState >, **ConnectionId::COMPARATOR** > (p. 913), **decaf::util::concurrent::ConcurrentStlMap**< **Pointer**< **ConsumerId** >, **Pointer**< **ConsumerState** >, **ConsumerId::COMPARATOR** > (p. 913), **decaf::util::concurrent::ConcurrentStlMap**< **Pointer**< **SessionId** >, **Pointer**< **SessionState** >, **SessionId::COMPARATOR** > (p. 913), **decaf::util::concurrent::ConcurrentStlMap**< **Pointer**< **LocalTransactionId** >, **Pointer**< **TransactionState** >, **LocalTransactionId::COMPARATOR** > (p. 913), **decaf::util::concurrent::ConcurrentStlMap**< **Pointer**< **ProducerId** >, **Pointer**< **ProducerState** >, **ProducerId::COMPARATOR** > (p. 913), **decaf::util::StlMap**< **K**, **V**, **COMPARATOR** > (p. 2560), **decaf::util::StlMap**< **cms::Session** *, **SessionResolver** * > (p. 2560), **decaf::util::StlMap**< **std::string**, **WireFormatFactory** * > (p. 2560), **decaf::util::StlMap**< **decaf::lang::Runnable** *, **decaf::util::TimerTask** * > (p. 2560), **decaf::util::StlMap**< **std::string**, **PrimitiveValueNode** > (p. 2560), **decaf::util::StlMap**< **std::string**, **cms::Queue** * > (p. 2560), **decaf::util::StlMap**< **std::string**, **CachedConsumer** * > (p. 2560), **decaf::util::StlMap**< **Pointer**< **commands::ConsumerId** >, **ActiveMQConsumer** *, **commands::ConsumerId::COMPARATOR** > (p. 2560), **decaf::util::StlMap**< **std::string**, **TransportFactory** * > (p. 2560), **decaf::util::StlMap**< **Pointer**< **ConsumerId** >, **Pointer**< **ConsumerInfo** >, **ConsumerId::COMPARATOR** > (p. 2560), **decaf::util::StlMap**< **int**, **Pointer**< **Command** > > (p. 2560), **decaf::util::StlMap**< **std::string**, **CachedProducer** * > (p. 2560), and **decaf::util::StlMap**< **std::string**, **cms::Topic** * > (p. 2560).

```

6.332.3.9  template<typename K, typename V, typename COMPARATOR = std::less<K>>
            virtual std::vector<K> decaf::util::Map< K, V, COMPARATOR >::keySet ( )
            const [pure virtual]
  
```

Returns a **Set** (p. 2397) view of the mappings contained in this map.

The set is backed by the map, so changes to the map are reflected in the set, and vice-versa. If the map is modified while an iteration over the set is in progress (except through the iterator's own remove operation, or through the setValue operation on a map entry returned by the iterator) the results of the iteration are undefined. The set supports element removal, which removes the corresponding mapping from the map, via the **Iterator.remove** (p. 1560), **Set.remove** (p. 861), **removeAll**, **retainAll** and **clear** operations. It does not support the **add** or **addAll** operations.

Returns

the entire set of keys in this map as a **std::vector**.

Implemented in **decaf::util::concurrent::ConcurrentStlMap**< **K**, **V**, **COMPARATOR** > (p. 913), **decaf::util::concurrent::ConcurrentStlMap**< **Pointer**< **MessageId** >, **Pointer**< **Message** >, **MessageId::COMPARATOR** > (p. 913), **decaf::util::concurrent::ConcurrentStlMap**< **Pointer**< **ConnectionId** >, **Pointer**< **ConnectionState** >, **ConnectionId::COMPARATOR** > (p. 913), **decaf::util::concurrent::ConcurrentStlMap**< **Pointer**< **ConsumerId** >, **Pointer**< **ConsumerState** >, **ConsumerId::COMPARATOR** > (p. 913), **decaf::util::concurrent::ConcurrentStlMap**< **Pointer**< **SessionId** >, **Pointer**< **SessionState** >, **SessionId::COMPARATOR** > (p. 913), **decaf::util::concurrent::ConcurrentStlMap**<

Pointer< LocalTransactionId >, Pointer< TransactionState >, LocalTransactionId::COMPARATOR > (p. 913), decaf::util::concurrent::ConcurrentStlMap< Pointer< ProducerId >, Pointer< ProducerState >, ProducerId::COMPARATOR > (p. 913), decaf::util::StlMap< K, V, COMPARATOR > (p. 2560), decaf::util::StlMap< cms::Session *, SessionResolver * > (p. 2560), decaf::util::StlMap< std::string, WireFormatFactory * > (p. 2560), decaf::util::StlMap< decaf::lang::Runnable *, decaf::util::TimerTask * > (p. 2560), decaf::util::StlMap< std::string, PrimitiveValueNode > (p. 2560), decaf::util::StlMap< std::string, cms::Queue * > (p. 2560), decaf::util::StlMap< std::string, CachedConsumer * > (p. 2560), decaf::util::StlMap< Pointer< commands::ConsumerId >, ActiveMQConsumer *, commands::ConsumerId::COMPARATOR > (p. 2560), decaf::util::StlMap< std::string, TransportFactory * > (p. 2560), decaf::util::StlMap< Pointer< ConsumerId >, Pointer< ConsumerInfo >, ConsumerId::COMPARATOR > (p. 2560), decaf::util::StlMap< int, Pointer< Command > > (p. 2560), decaf::util::StlMap< std::string, CachedProducer * > (p. 2560), and decaf::util::StlMap< std::string, cms::Topic * > (p. 2560).

Referenced by decaf::util::StlMap< std::string, cms::Topic * >::equals(), decaf::util::concurrent::ConcurrentStlMap< Pointer< ProducerId >, Pointer< ProducerState >, ProducerId::COMPARATOR >::equals(), decaf::util::StlMap< std::string, cms::Topic * >::putAll(), and decaf::util::concurrent::ConcurrentStlMap< Pointer< ProducerId >, Pointer< ProducerState >, ProducerId::COMPARATOR >::putAll().

6.332.3.10 `template<typename K, typename V, typename COMPARATOR = std::less<K>>
virtual void decaf::util::Map< K, V, COMPARATOR >::put (const K & key, const
V & value) [pure virtual]`

Sets the value for the specified key.

Parameters

<i>key</i>	The target key.
<i>value</i>	The value to be set.

Exceptions

<i>Unsupported- OperationException</i>	if this map is unmodifiable.
--	------------------------------

Implemented in `decaf::util::concurrent::ConcurrentStlMap< K, V, COMPARATOR >` (p. 914), `decaf::util::StlMap< K, V, COMPARATOR >` (p. 2561), `decaf::util::StlMap< cms::Session *, SessionResolver * >` (p. 2561), `decaf::util::StlMap< std::string, WireFormatFactory * >` (p. 2561), `decaf::util::StlMap< decaf::lang::Runnable *, decaf::util::TimerTask * >` (p. 2561), `decaf::util::StlMap< std::string, PrimitiveValueNode >` (p. 2561), `decaf::util::StlMap< std::string, cms::Queue * >` (p. 2561), `decaf::util::StlMap< std::string, CachedConsumer * >` (p. 2561), `decaf::util::StlMap< Pointer< commands::ConsumerId >, ActiveMQConsumer *, commands::ConsumerId::COMPARATOR >` (p. 2561), `decaf::util::StlMap< std::string, TransportFactory * >` (p. 2561), `decaf::util::StlMap<`

Pointer< **ConsumerId** >, **Pointer**< **ConsumerInfo** >, **ConsumerId::COMPARATOR** > (p. 2561), **decaf::util::StlMap**< **int**, **Pointer**< **Command** > > (p. 2561), **decaf::util::StlMap**< **std::string**, **CachedProducer** * > (p. 2561), and **decaf::util::StlMap**< **std::string**, **cms::Topic** * > (p. 2561).

```
6.332.3.11  template<typename K, typename V, typename COMPARATOR = std::less<K>>
             virtual void decaf::util::Map< K, V, COMPARATOR >::putAll ( const Map< K,
             V, COMPARATOR > & other ) [pure virtual]
```

Stores a copy of the Mappings contained in the other **Map** (p. 1768) in this one.

Parameters

<i>other</i>	A Map (p. 1768) instance whose elements are to all be inserted in this Map (p. 1768).
--------------	---

Exceptions

<i>Unsupported-OperationException</i>	If the implementing class does not support the putAll operation.
---------------------------------------	--

Implemented in **decaf::util::concurrent::ConcurrentStlMap**< **K**, **V**, **COMPARATOR** > (p. 915), **decaf::util::StlMap**< **K**, **V**, **COMPARATOR** > (p. 2562), **decaf::util::StlMap**< **cms::Session** *, **SessionResolver** * > (p. 2562), **decaf::util::StlMap**< **std::string**, **WireFormatFactory** * > (p. 2562), **decaf::util::StlMap**< **decaf::lang::Runnable** *, **decaf::util::TimerTask** * > (p. 2562), **decaf::util::StlMap**< **std::string**, **PrimitiveValueNode** > (p. 2562), **decaf::util::StlMap**< **std::string**, **cms::Queue** * > (p. 2562), **decaf::util::StlMap**< **std::string**, **CachedConsumer** * > (p. 2562), **decaf::util::StlMap**< **Pointer**< **commands::ConsumerId** >, **ActiveMQConsumer** *, **commands::ConsumerId::COMPARATOR** > (p. 2562), **decaf::util::StlMap**< **std::string**, **TransportFactory** * > (p. 2562), **decaf::util::StlMap**< **Pointer**< **ConsumerId** >, **Pointer**< **ConsumerInfo** >, **ConsumerId::COMPARATOR** > (p. 2562), **decaf::util::StlMap**< **int**, **Pointer**< **Command** > > (p. 2562), **decaf::util::StlMap**< **std::string**, **CachedProducer** * > (p. 2562), and **decaf::util::StlMap**< **std::string**, **cms::Topic** * > (p. 2562).

```
6.332.3.12  template<typename K, typename V, typename COMPARATOR = std::less<K>>
             virtual V decaf::util::Map< K, V, COMPARATOR >::remove ( const K & key )
             [pure virtual]
```

Removes the value (key/value pair) for the specified key from the map, returns a copy of the value that was mapped to the key.

Parameters

<i>key</i>	The search key.
------------	-----------------

Returns

a copy of the element that was previously mapped to the given key

Exceptions

NoSuchElementException (p. 1984)	if this key is not in the Map (p. 1768).
<i>UnsupportedOperationException</i>	if this map is unmodifiable.

Implemented in **decaf::util::concurrent::ConcurrentStlMap< K, V, COMPARATOR >** (p. 916), **decaf::util::StlMap< K, V, COMPARATOR >** (p. 2562), **decaf::util::StlMap< cms::Session *, SessionResolver * >** (p. 2562), **decaf::util::StlMap< std::string, WireFormatFactory * >** (p. 2562), **decaf::util::StlMap< decaf::lang::Runnable *, decaf::util::TimerTask * >** (p. 2562), **decaf::util::StlMap< std::string, PrimitiveValueNode >** (p. 2562), **decaf::util::StlMap< std::string, cms::Queue * >** (p. 2562), **decaf::util::StlMap< std::string, CachedConsumer * >** (p. 2562), **decaf::util::StlMap< Pointer< commands::ConsumerId >, ActiveMQConsumer *, commands::ConsumerId::COMPARATOR >** (p. 2562), **decaf::util::StlMap< std::string, TransportFactory * >** (p. 2562), **decaf::util::StlMap< Pointer< ConsumerId >, Pointer< ConsumerInfo >, ConsumerId::COMPARATOR >** (p. 2562), **decaf::util::StlMap< int, Pointer< Command > >** (p. 2562), **decaf::util::StlMap< std::string, CachedProducer * >** (p. 2562), and **decaf::util::StlMap< std::string, cms::Topic * >** (p. 2562).

```
6.332.3.13  template<typename K, typename V, typename COMPARATOR = std::less<K>>
            virtual int decaf::util::Map< K, V, COMPARATOR >::size ( ) const [pure
            virtual]
```

Returns

The number of elements (key/value pairs) in this map.

Implemented in **decaf::util::concurrent::ConcurrentStlMap< K, V, COMPARATOR >** (p. 919), **decaf::util::concurrent::ConcurrentStlMap< Pointer< MessageId >, Pointer< Message >, MessageId::COMPARATOR >** (p. 919), **decaf::util::concurrent::ConcurrentStlMap< Pointer< ConnectionId >, Pointer< ConnectionState >, ConnectionId::COMPARATOR >** (p. 919), **decaf::util::concurrent::ConcurrentStlMap< Pointer< ConsumerId >, Pointer< ConsumerState >, ConsumerId::COMPARATOR >** (p. 919), **decaf::util::concurrent::ConcurrentStlMap< Pointer< SessionId >, Pointer< SessionState >, SessionId::COMPARATOR >** (p. 919), **decaf::util::concurrent::ConcurrentStlMap< Pointer< LocalTransactionId >, Pointer< TransactionState >, LocalTransactionId::COMPARATOR >** (p. 919), **decaf::util::concurrent::ConcurrentStlMap< Pointer< ProducerId >, Pointer< ProducerState >, ProducerId::COMPARATOR >** (p. 919), **decaf::util::StlMap< K, V, COMPARATOR >** (p. 2563), **decaf::util::StlMap< cms::Session *, SessionResolver * >** (p. 2563), **decaf::util::StlMap< std::string, WireFormatFactory * >** (p. 2563), **decaf::util::StlMap< decaf::**

`::lang::Runnable *`, `decaf::util::TimerTask *` > (p. 2563), `decaf::util::StlMap< std::string, PrimitiveValueNode >` (p. 2563), `decaf::util::StlMap< std::string, cms::Queue * >` (p. 2563), `decaf::util::StlMap< std::string, CachedConsumer * >` (p. 2563), `decaf::util::StlMap< Pointer< commands::ConsumerId >, ActiveMQConsumer *, commands::ConsumerId::COMPARATOR >` (p. 2563), `decaf::util::StlMap< std::string, TransportFactory * >` (p. 2563), `decaf::util::StlMap< Pointer< ConsumerId >, Pointer< ConsumerInfo >, ConsumerId::COMPARATOR >` (p. 2563), `decaf::util::StlMap< int, Pointer< Command > >` (p. 2563), `decaf::util::StlMap< std::string, CachedProducer * >` (p. 2563), and `decaf::util::StlMap< std::string, cms::Topic * >` (p. 2563).

```
6.332.3.14  template<typename K, typename V, typename COMPARATOR = std::less<K>>
             virtual std::vector<V> decaf::util::Map< K, V, COMPARATOR >::values ( )
             const [pure virtual]
```

Returns

the entire set of values in this map as a `std::vector`.

Implemented in `decaf::util::concurrent::ConcurrentStlMap< K, V, COMPARATOR >` (p. 919), `decaf::util::concurrent::ConcurrentStlMap< Pointer< MessageId >, Pointer< Message >, MessageId::COMPARATOR >` (p. 919), `decaf::util::concurrent::ConcurrentStlMap< Pointer< ConnectionId >, Pointer< ConnectionState >, ConnectionId::COMPARATOR >` (p. 919), `decaf::util::concurrent::ConcurrentStlMap< Pointer< ConsumerId >, Pointer< ConsumerState >, ConsumerId::COMPARATOR >` (p. 919), `decaf::util::concurrent::ConcurrentStlMap< Pointer< SessionId >, Pointer< SessionState >, SessionId::COMPARATOR >` (p. 919), `decaf::util::concurrent::ConcurrentStlMap< Pointer< LocalTransactionId >, Pointer< TransactionState >, LocalTransactionId::COMPARATOR >` (p. 919), `decaf::util::concurrent::ConcurrentStlMap< Pointer< ProducerId >, Pointer< ProducerState >, ProducerId::COMPARATOR >` (p. 919), `decaf::util::StlMap< K, V, COMPARATOR >` (p. 2563), `decaf::util::StlMap< cms::Session *, SessionResolver * >` (p. 2563), `decaf::util::StlMap< std::string, WireFormatFactory * >` (p. 2563), `decaf::util::StlMap< decaf::lang::Runnable *, decaf::util::TimerTask * >` (p. 2563), `decaf::util::StlMap< std::string, PrimitiveValueNode >` (p. 2563), `decaf::util::StlMap< std::string, cms::Queue * >` (p. 2563), `decaf::util::StlMap< std::string, CachedConsumer * >` (p. 2563), `decaf::util::StlMap< Pointer< commands::ConsumerId >, ActiveMQConsumer *, commands::ConsumerId::COMPARATOR >` (p. 2563), `decaf::util::StlMap< std::string, TransportFactory * >` (p. 2563), `decaf::util::StlMap< Pointer< ConsumerId >, Pointer< ConsumerInfo >, ConsumerId::COMPARATOR >` (p. 2563), `decaf::util::StlMap< int, Pointer< Command > >` (p. 2563), `decaf::util::StlMap< std::string, CachedProducer * >` (p. 2563), and `decaf::util::StlMap< std::string, cms::Topic * >` (p. 2563).

The documentation for this class was generated from the following file:

- `src/main/decaf/util/Map.h`

6.333 cms::MapMessage Class Reference

A **MapMessage** (p. 1779) object is used to send a set of name-value pairs.

```
#include <src/main/cms/MapMessage.h>
```

Inheritance diagram for cms::MapMessage:

Public Member Functions

- virtual **~MapMessage** () throw ()
- virtual bool **isEmpty** () const =0
*Returns true if there are no values stored in the **MapMessage** (p. 1779) body.*
- virtual std::vector< std::string > **getMapNames** () const =0
*Returns an Enumeration of all the names in the **MapMessage** (p. 1779) object.*
- virtual bool **itemExists** (const std::string &name) const =0
*Indicates whether an item exists in this **MapMessage** (p. 1779) object.*
- virtual bool **getBoolean** (const std::string &name) const =0
Returns the Boolean value of the Specified name.
- virtual void **setBoolean** (const std::string &name, bool value)=0
Sets a boolean value with the specified name into the Map.
- virtual unsigned char **getByte** (const std::string &name) const =0
Returns the Byte value of the Specified name.
- virtual void **setByte** (const std::string &name, unsigned char value)=0
Sets a Byte value with the specified name into the Map.
- virtual std::vector< unsigned char > **getBytes** (const std::string &name) const =0
Returns the Bytes value of the Specified name.
- virtual void **setBytes** (const std::string &name, const std::vector< unsigned char > &value)=0
Sets a Bytes value with the specified name into the Map.
- virtual char **getChar** (const std::string &name) const =0
Returns the Char value of the Specified name.
- virtual void **setChar** (const std::string &name, char value)=0
Sets a Char value with the specified name into the Map.
- virtual double **getDouble** (const std::string &name) const =0
Returns the Double value of the Specified name.
- virtual void **setDouble** (const std::string &name, double value)=0
Sets a Double value with the specified name into the Map.
- virtual float **getFloat** (const std::string &name) const =0
Returns the Float value of the Specified name.
- virtual void **setFloat** (const std::string &name, float value)=0
Sets a Float value with the specified name into the Map.

- virtual int **getInt** (const std::string &name) const =0
Returns the Int value of the Specified name.
- virtual void **setInt** (const std::string &name, int value)=0
Sets a Int value with the specified name into the Map.
- virtual long long **getLong** (const std::string &name) const =0
Returns the Long value of the Specified name.
- virtual void **setLong** (const std::string &name, long long value)=0
Sets a Long value with the specified name into the Map.
- virtual short **getShort** (const std::string &name) const =0
Returns the Short value of the Specified name.
- virtual void **setShort** (const std::string &name, short value)=0
Sets a Short value with the specified name into the Map.
- virtual std::string **getString** (const std::string &name) const =0
Returns the String value of the Specified name.
- virtual void **setString** (const std::string &name, const std::string &value)=0
Sets a String value with the specified name into the Map.

6.333.1 Detailed Description

A **MapMessage** (p. 1779) object is used to send a set of name-value pairs.

The names are String objects, and the values are primitive data types in the Java programming language. The names must have a value that is not null, and not an empty string. The entries can be accessed sequentially or randomly by name. The order of the entries is undefined. **MapMessage** (p. 1779) inherits from the **Message** (p. 1839) interface and adds a message body that contains a Map.

When a client receives a **MapMessage** (p. 1779), it is in read-only mode. If a client attempts to write to the message at this point, a **MessageNotWriteableException** (p. 1923) is thrown. To place the **MapMessage** (p. 1779) back into a state where it can be read from and written to, call the `clearBody` method.

MapMessage (p. 1779) objects support the following conversion table. The marked cases must be supported. The unmarked cases must throw a **CMSEException** (p. 826). The String-to-primitive conversions may throw a **MessageFormatException** (p. 1906) if the primitive's `valueOf()` method does not accept it as a valid String representation of the primitive.

A value written as the row type can be read as the column type.

	boolean	byte	short	char	int	long	float	double	String	byte[]
boolean	X									X
byte		X	X		X	X				X
short			X		X	X				X
char				X						X
int					X	X				X
long						X				X
float							X	X		X

double								X	X	
String		X	X	X	X	X	X	X	X	
byte[]										X

Since

1.0

6.333.2 Constructor & Destructor Documentation

6.333.2.1 virtual cms::MapMessage::~MapMessage () throw () [virtual]

6.333.3 Member Function Documentation

6.333.3.1 virtual bool cms::MapMessage::getBoolean (const std::string & name) const
[pure virtual]

Returns the Boolean value of the Specified name.

Parameters

name	Name of the value to fetch from the map
------	---

Exceptions

CMSException (p. 826)	- if the operation fails due to an internal error.
MessageFormatException (p. 1906)	- if this type conversion is invalid.

Implemented in **activemq::commands::ActiveMQMapMessage** (p. 273).

6.333.3.2 virtual unsigned char cms::MapMessage::getByte (const std::string & name)
const [pure virtual]

Returns the Byte value of the Specified name.

Parameters

name	Name of the value to fetch from the map
------	---

Exceptions

CMSException (p. 826)	- if the operation fails due to an internal error.
---------------------------------	--

<i>MessageFormat-Exception</i> (p. 1906)	- if this type conversion is invalid.
--	---------------------------------------

Implemented in **activemq::commands::ActiveMQMapMessage** (p. 273).

6.333.3.3 `virtual std::vector<unsigned char> cms::MapMessage::getBytes (const std::string & name) const [pure virtual]`

Returns the Bytes value of the Specified name.

Parameters

<i>name</i>	Name of the value to fetch from the map
-------------	---

Exceptions

<i>CMSException</i> (p. 826)	- if the operation fails due to an internal error.
<i>MessageFormat-Exception</i> (p. 1906)	- if this type conversion is invalid.

Implemented in **activemq::commands::ActiveMQMapMessage** (p. 274).

6.333.3.4 `virtual char cms::MapMessage::getChar (const std::string & name) const [pure virtual]`

Returns the Char value of the Specified name.

Parameters

<i>name</i>	name of the value to fetch from the map
-------------	---

Exceptions

<i>CMSException</i> (p. 826)	- if the operation fails due to an internal error.
<i>MessageFormat-Exception</i> (p. 1906)	- if this type conversion is invalid.

Implemented in **activemq::commands::ActiveMQMapMessage** (p. 274).

6.333.3.5 virtual double cms::MapMessage::getDouble (const std::string & *name*) const
[pure virtual]

Returns the Double value of the Specified name.

Parameters

<i>name</i>	Name of the value to fetch from the map
-------------	---

Exceptions

CMSException (p. 826)	- if the operation fails due to an internal error.
MessageFormatException (p. 1906)	- if this type conversion is invalid.

Implemented in **activemq::commands::ActiveMQMapMessage** (p. 275).

6.333.3.6 virtual float cms::MapMessage::getFloat (const std::string & *name*) const
[pure virtual]

Returns the Float value of the Specified name.

Parameters

<i>name</i>	Name of the value to fetch from the map
-------------	---

Exceptions

CMSException (p. 826)	- if the operation fails due to an internal error.
MessageFormatException (p. 1906)	- if this type conversion is invalid.

Implemented in **activemq::commands::ActiveMQMapMessage** (p. 275).

6.333.3.7 virtual int cms::MapMessage::getInt (const std::string & *name*) const
[pure virtual]

Returns the Int value of the Specified name.

Parameters

<i>name</i>	Name of the value to fetch from the map
-------------	---

Exceptions

<i>CMSEException</i> (p. 826)	- if the operation fails due to an internal error.
<i>MessageFormatException</i> (p. 1906)	- if this type conversion is invalid.

Implemented in **activemq::commands::ActiveMQMapMessage** (p. 276).

6.333.3.8 `virtual long long cms::MapMessage::getLong (const std::string & name) const`
[pure virtual]

Returns the Long value of the Specified name.

Parameters

<i>name</i>	Name of the value to fetch from the map
-------------	---

Exceptions

<i>CMSEException</i> (p. 826)	- if the operation fails due to an internal error.
<i>MessageFormatException</i> (p. 1906)	- if this type conversion is invalid.

Implemented in **activemq::commands::ActiveMQMapMessage** (p. 276).

6.333.3.9 `virtual std::vector< std::string > cms::MapMessage::getMapNames () const`
[pure virtual]

Returns an Enumeration of all the names in the **MapMessage** (p. 1779) object.

Returns

STL Vector of String values, each of which is the name of an item in the **MapMessage** (p. 1779)

Exceptions

<i>CMSEException</i> (p. 826)	- if the operation fails due to an internal error.
---	--

Implemented in **activemq::commands::ActiveMQMapMessage** (p. 277).

6.333.3.10 virtual short cms::MapMessage::getShort (const std::string & name) const
[pure virtual]

Returns the Short value of the Specified name.

Parameters

<i>name</i>	Name of the value to fetch from the map
-------------	---

Exceptions

CMSException (p. 826)	- if the operation fails due to an internal error.
MessageFormatException (p. 1906)	- if this type conversion is invalid.

Implemented in **activemq::commands::ActiveMQMapMessage** (p. 277).

6.333.3.11 virtual std::string cms::MapMessage::getString (const std::string & name)
const [pure virtual]

Returns the String value of the Specified name.

Parameters

<i>name</i>	Name of the value to fetch from the map
-------------	---

Exceptions

CMSException (p. 826)	- if the operation fails due to an internal error.
MessageFormatException (p. 1906)	- if this type conversion is invalid.

Implemented in **activemq::commands::ActiveMQMapMessage** (p. 278).

6.333.3.12 virtual bool cms::MapMessage::isEmpty () const [pure virtual]

Returns true if there are no values stored in the **MapMessage** (p. 1779) body.

Returns

true if the body of the **MapMessage** (p. 1779) contains no elements.

Exceptions

<i>CMSException</i> (p. 826)	if the operation fails due to an internal error.
--	--

Implemented in **activemq::commands::ActiveMQMapMessage** (p. 278).

6.333.3.13 **virtual bool cms::MapMessage::itemExists (const std::string & name) const**
[pure virtual]

Indicates whether an item exists in this **MapMessage** (p. 1779) object.

Parameters

<i>name</i>	String name of the Object in question
-------------	---------------------------------------

Returns

boolean value indicating if the name is in the map

Exceptions

<i>CMSException</i> (p. 826)	- if the operation fails due to an internal error.
--	--

Implemented in **activemq::commands::ActiveMQMapMessage** (p. 279).

6.333.3.14 **virtual void cms::MapMessage::setBoolean (const std::string & name, bool value)** [pure virtual]

Sets a boolean value with the specified name into the Map.

Parameters

<i>name</i>	the name of the boolean
<i>value</i>	the boolean value to set in the Map

Exceptions

<i>CMSException</i> (p. 826)	- if the operation fails due to an internal error.
<i>MessageNotWritableException</i>	- if the Message (p. 1839) is in Read-only Mode.

Implemented in **activemq::commands::ActiveMQMapMessage** (p. 279).

6.333.3.15 virtual void **cms::MapMessage::setByte** (const std::string & *name*, unsigned char *value*) [pure virtual]

Sets a Byte value with the specified name into the Map.

Parameters

<i>name</i>	the name of the Byte
<i>value</i>	the Byte value to set in the Map

Exceptions

CMSException (p. 826)	- if the operation fails due to an internal error.
MessageNot-WriteableException (p. 1923)	- if the Message (p. 1839) is in Read-only Mode.

Implemented in **activemq::commands::ActiveMQMapMessage** (p. 279).

6.333.3.16 virtual void **cms::MapMessage::setBytes** (const std::string & *name*, const std::vector< unsigned char > & *value*) [pure virtual]

Sets a Bytes value with the specified name into the Map.

Parameters

<i>name</i>	The name of the Bytes
<i>value</i>	The Bytes value to set in the Map

Exceptions

CMSException (p. 826)	- if the operation fails due to an internal error.
MessageNot-WriteableException (p. 1923)	- if the Message (p. 1839) is in Read-only Mode.

Implemented in **activemq::commands::ActiveMQMapMessage** (p. 280).

6.333.3.17 virtual void **cms::MapMessage::setChar** (const std::string & *name*, char *value*) [pure virtual]

Sets a Char value with the specified name into the Map.

Parameters

<i>name</i>	the name of the Char
<i>value</i>	the Char value to set in the Map

Exceptions

CMSException (p. 826)	- if the operation fails due to an internal error.
MessageNot-WriteableException (p. 1923)	- if the Message (p. 1839) is in Read-only Mode.

Implemented in **activemq::commands::ActiveMQMapMessage** (p. 280).

6.333.3.18 virtual void **cms::MapMessage::setDouble** (const std::string & *name*, double *value*) [pure virtual]

Sets a Double value with the specified name into the Map.

Parameters

<i>name</i>	The name of the Double
<i>value</i>	The Double value to set in the Map

Exceptions

CMSException (p. 826)	- if the operation fails due to an internal error.
MessageNot-WriteableException (p. 1923)	- if the Message (p. 1839) is in Read-only Mode.

Implemented in **activemq::commands::ActiveMQMapMessage** (p. 281).

6.333.3.19 virtual void **cms::MapMessage::setFloat** (const std::string & *name*, float *value*) [pure virtual]

Sets a Float value with the specified name into the Map.

Parameters

<i>name</i>	The name of the Float
<i>value</i>	The Float value to set in the Map

Exceptions

<i>CMSEException</i> (p. 826)	- if the operation fails due to an internal error.
<i>MessageNotWriteableException</i> (p. 1923)	- if the Message (p. 1839) is in Read-only Mode.

Implemented in **activemq::commands::ActiveMQMapMessage** (p. 281).

6.333.3.20 virtual void **cms::MapMessage::setInt** (const std::string & *name*, int *value*)
[pure virtual]

Sets a Int value with the specified name into the Map.

Parameters

<i>name</i>	The name of the Int
<i>value</i>	The Int value to set in the Map

Exceptions

<i>CMSEException</i> (p. 826)	- if the operation fails due to an internal error.
<i>MessageNotWriteableException</i> (p. 1923)	- if the Message (p. 1839) is in Read-only Mode.

Implemented in **activemq::commands::ActiveMQMapMessage** (p. 281).

6.333.3.21 virtual void **cms::MapMessage::setLong** (const std::string & *name*, long long *value*) [pure virtual]

Sets a Long value with the specified name into the Map.

Parameters

<i>name</i>	The name of the Long
<i>value</i>	The Long value to set in the Map

Exceptions

<i>CMSEException</i> (p. 826)	- if the operation fails due to an internal error.
<i>MessageNotWriteableException</i> (p. 1923)	- if the Message (p. 1839) is in Read-only Mode.

Implemented in **activemq::commands::ActiveMQMapMessage** (p. 282).

6.333.3.22 **virtual void cms::MapMessage::setShort** (const std::string & *name*, short *value*) [pure virtual]

Sets a Short value with the specified name into the Map.

Parameters

<i>name</i>	The name of the Short
<i>value</i>	The Short value to set in the Map

Exceptions

CMSException (p. 826)	- if the operation fails due to an internal error.
MessageNot-WriteableException (p. 1923)	- if the Message (p. 1839) is in Read-only Mode.

Implemented in **activemq::commands::ActiveMQMapMessage** (p. 282).

6.333.3.23 **virtual void cms::MapMessage::setString** (const std::string & *name*, const std::string & *value*) [pure virtual]

Sets a String value with the specified name into the Map.

Parameters

<i>name</i>	The name of the String
<i>value</i>	The String value to set in the Map

Exceptions

CMSException (p. 826)	- if the operation fails due to an internal error.
MessageNot-WriteableException (p. 1923)	- if the Message (p. 1839) is in Read-only Mode.

Implemented in **activemq::commands::ActiveMQMapMessage** (p. 283).

The documentation for this class was generated from the following file:

- src/main/cms/**MapMessage.h**

6.334 decaf::util::logging::MarkBlockLogger Class Reference

Defines a class that can be used to mark the entry and exit from scoped blocks.

```
#include <src/main/decaf/util/logging/MarkBlockLogger.h>
```

Public Member Functions

- **MarkBlockLogger** (**Logger** *logger, const std::string &blockName)
Constructor - Marks Block entry.
- virtual ~**MarkBlockLogger** ()

6.334.1 Detailed Description

Defines a class that can be used to mark the entry and exit from scoped blocks.

Create an instance of this class at the start of a scoped block, passing it the logger to use and the name of the block. The block entry and exit will be marked using the scope name, logger to the logger at the MARKBLOCK log level.

6.334.2 Constructor & Destructor Documentation

6.334.2.1 **decaf::util::logging::MarkBlockLogger::MarkBlockLogger (Logger * logger, const std::string & blockName)** [*inline*]

Constructor - Marks Block entry.

Parameters

<i>logger</i>	Logger (p. 1693) to use
<i>blockName</i>	Block name

6.334.2.2 **virtual decaf::util::logging::MarkBlockLogger::~MarkBlockLogger ()**
[*inline, virtual*]

The documentation for this class was generated from the following file:

- src/main/decaf/util/logging/**MarkBlockLogger.h**

6.335 activemq::wireformat::MarshalAware Class Reference

```
#include <src/main/activemq/wireformat/MarshalAware.h>
```

Inheritance diagram for `activemq::wireformat::MarshalAware`:

Public Member Functions

- virtual `~MarshalAware()`
- virtual bool `isMarshalAware()` const =0
Determine if the class implementing this interface is really wanting to be told about marshaling.
- virtual void `beforeMarshal(WireFormat *wireFormat)=0`
Called before marshaling is started to prepare the object to be marshaled.
- virtual void `afterMarshal(WireFormat *wireFormat)=0`
Called after marshaling is started to cleanup the object being marshaled.
- virtual void `beforeUnmarshal(WireFormat *wireFormat)=0`
Called before unmarshaling is started to prepare the object to be unmarshaled.
- virtual void `afterUnmarshal(WireFormat *wireFormat)=0`
Called after unmarshaling is started to cleanup the object being unmarshaled.
- virtual void `setMarshaledForm(WireFormat *wireFormat, const std::vector<char> &data)=0`
Called to set the data to this object that will contain the objects marshaled form.
- virtual std::vector< unsigned char > `getMarshaledForm(WireFormat *wireFormat)=0`
Called to get the data to this object that will contain the objects marshaled form.

6.335.1 Constructor & Destructor Documentation

6.335.1.1 virtual `activemq::wireformat::MarshalAware::~MarshalAware()` [inline, virtual]

6.335.2 Member Function Documentation

6.335.2.1 virtual void `activemq::wireformat::MarshalAware::afterMarshal(WireFormat * wireFormat)` [pure virtual]

Called after marshaling is started to cleanup the object being marshaled.

Parameters

<i>wireFormat</i>	The wireformat object to control marshaling
-------------------	---

Exceptions

<i>IOException</i>	if an I/O error occurs.
--------------------	-------------------------

6.335.2.2 virtual void **activemq::wireformat::MarshalAware::afterUnmarshal** (
WireFormat * wireFormat) [pure virtual]

Called after unmarshaling is started to cleanup the object being unmarshaled.

Parameters

<i>wireFormat</i>	The wireformat object to control marshaling
-------------------	---

Exceptions

<i>IOException</i>	if an I/O error occurs.
--------------------	-------------------------

6.335.2.3 virtual void **activemq::wireformat::MarshalAware::beforeMarshal** (
WireFormat * wireFormat) [pure virtual]

Called before marshaling is started to prepare the object to be marshaled.

Parameters

<i>wireFormat</i>	The wireformat object to control marshaling
-------------------	---

Exceptions

<i>IOException</i>	if an I/O error occurs.
--------------------	-------------------------

Implemented in **activemq::commands::ActiveMQMapMessage** (p.271), and **activemq::commands::ActiveMQTextMessage** (p.406).

6.335.2.4 virtual void **activemq::wireformat::MarshalAware::beforeUnmarshal** (
WireFormat * wireFormat) [pure virtual]

Called before unmarshaling is started to prepare the object to be unmarshaled.

Parameters

<i>wireFormat</i>	The wireformat object to control marshaling
-------------------	---

Exceptions

<i>IOException</i>	if an I/O error occurs.
--------------------	-------------------------

6.335.2.5 `virtual std::vector<unsigned char> activemq::wireformat::MarshalAware::getMarshaledForm (WireFormat * wireFormat) [pure virtual]`

Called to get the data to this object that will contain the objects marshaled form.

Parameters

<i>wireFormat</i>	The wireformat object to control unmarshaling
-------------------	---

Returns

buffer that holds the objects data.

6.335.2.6 `virtual bool activemq::wireformat::MarshalAware::isMarshalAware () const [pure virtual]`

Determine if the class implementing this interface is really wanting to be told about marshaling.

Normally if you didn't want to be marshal aware you just wouldn't implement this interface but since this is C++ and we don't have true interfaces we need a flat inheritance hierarchy, so we always implement this.

Returns

true if this class cares about marshaling.

Implemented in **activemq::commands::Message** (p. 1832), **activemq::commands::WireFormatInfo** (p. 2895), **activemq::commands::ActiveMQMapMessage** (p. 278), and **activemq::commands::BaseDataStructure** (p. 530).

6.335.2.7 `virtual void activemq::wireformat::MarshalAware::setMarshaledForm (WireFormat * wireFormat, const std::vector< char > & data) [pure virtual]`

Called to set the data to this object that will contain the objects marshaled form.

Parameters

<i>wireFormat</i>	- the wireformat object to control unmarshaling
<i>data</i>	- vector of object binary data
<i>wireFormat</i>	The wireformat object to control marshaling
<i>data</i>	A vector of bytes that contains the object in marshaled form.

Exceptions

<i>IOException</i>	if an I/O error occurs.
--------------------	-------------------------

The documentation for this class was generated from the following file:

- src/main/activemq/wireformat/**MarshalAware.h**

6.336 activemq::wireformat::openwire::marshal::generated::- **MarshallerFactory Class Reference**

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

```
#include <src/main/activemq/wireformat/openwire/marshal/generated/-  
MarshallerFactory.h>
```

Public Member Functions

- virtual **~MarshallerFactory** ()
- virtual void **configure** (**OpenWireFormat** *format)

6.336.1 Detailed Description

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

NOTE!: This file is auto generated - do not modify! if you need to make a change, please see the Groovy scripts in the activemq-openwire-generator module

6.336.2 Constructor & Destructor Documentation

6.336.2.1 virtual activemq::wireformat::openwire::marshal::generated-
::MarshallerFactory::~MarshallerFactory () [inline,
virtual]

6.336.3 Member Function Documentation

6.336.3.1 virtual void activemq::wireformat::openwire::marshal::generated-
::MarshallerFactory::configure (OpenWireFormat * *format*)
[virtual]

The documentation for this class was generated from the following file:

- src/main/activemq/wireformat/openwire/marshal/generated/**MarshallerFactory.-h**

6.337 activemq::util::MarshallingSupport Class Reference

```
#include <src/main/activemq/util/MarshallingSupport.h>
```

Public Member Functions

- **MarshallingSupport** ()
- virtual **~MarshallingSupport** ()

Static Public Member Functions

- static void **writeString** (**decaf::io::DataOutputStream** &dataOut, const std::string &value)
Write the string object to the given DataOutputStream as Raw bytes, no string encoding is done on this char values in the string.
- static void **writeString16** (**decaf::io::DataOutputStream** &dataOut, const std::string &value)
Write the string object to the given DataOutputStream as Raw bytes, no string encoding is done on this char values in the string.
- static void **writeString32** (**decaf::io::DataOutputStream** &dataOut, const std::string &value)
Write the string object to the given DataOutputStream as Raw bytes, no string encoding is done on this char values in the string.
- static std::string **readString16** (**decaf::io::DataInputStream** &dataIn)
Reads an Openwire encoded string from the provided DataInputStream.
- static std::string **readString32** (**decaf::io::DataInputStream** &dataIn)
Reads an Openwire encoded string from the provided DataInputStream.
- static std::string **asciiToModifiedUtf8** (const std::string &asciiString)
Given an ASCII String with byte values [0..255] convert the string to a string containing the modified UTF-8 form of that same string.
- static std::string **modifiedUtf8ToAscii** (const std::string modifiedUtf8String)
Given a string that contains bytes in the Java Modified UTF-8 format convert that string back into ASCII values from [0..255].

6.337.1 Constructor & Destructor Documentation

6.337.1.1 **activemq::util::MarshallingSupport::MarshallingSupport** ()

6.337.1.2 **virtual activemq::util::MarshallingSupport::~~MarshallingSupport** ()
[virtual]

6.337.2 Member Function Documentation

6.337.2.1 **static std::string activemq::util::MarshallingSupport::asciiToModifiedUtf8** (const std::string & *asciiString*) [static]

Given an ASCII String with byte values [0..255] convert the string to a string containing the modified UTF-8 form of that same string.

This allows an ASCII string containing values greater than 127 as well as embedded NULLs to be sent to a Java client.

Parameters

<i>asciiString</i>	The ASCII string to encode as Modified UTF-8
--------------------	--

Returns

a string containing the Modified UTF-8 encoded form of the provided string.

Exceptions

<i>UTFDataFormat-Exception</i>	if the length of the encoded string would exceed the size of an signed integer.
--------------------------------	---

6.337.2.2 static std::string activemq::util::MarshallingSupport::modifiedUtf8ToAscii
(const std::string *modifiedUtf8String*) [static]

Given a string that contains bytes in the Java Modified UTF-8 format convert that string back into ASCII values from [0..255].

This will handle any string sent from a Java client which contains values within the [0..255] range or has embedded Nulls. Strings that have encoded values greater than 255 will cause an exception to be thrown.

Parameters

<i>modified-Utf8String</i>	The string to convert from Modified UTF-8 to ASCII.
----------------------------	---

Returns

the ASCII encoded version of the provided string.

Exceptions

<i>UTFDataFormat-Exception</i>	if the provided string contains invalid data or the character values encoded in the string exceed ASCII value 255.
--------------------------------	--

6.337.2.3 static std::string activemq::util::MarshallingSupport::readString16 (*decaf::io::DataInputStream & dataIn*) [static]

Reads an Openwire encoded string from the provided DataInputStream.

No string processing is performed by this method, clients that know the data contains UTF-8 encoded content must use one of the utility methods of this class to decode the UTF-8 data.

This version assumes a size prefix of 16bits.

Parameters

<i>dataIn</i>	The DataInputStream to read the String data from.
---------------	---

Returns

the String value.

Exceptions

<i>IOException</i>	if an I/O error occurs while writing the string.
--------------------	--

6.337.2.4 static std::string activemq::util::MarshallingSupport::readString32 (decaf::io::DataInputStream & *dataIn*) [static]

Reads an Openwire encoded string from the provided DataInputStream.

No string processing is performed by this method, clients that know the data contains UTF-8 encoded content must use one of the utility methods of this class to decode the UTF-8 data.

This version assumes a size prefix of 32bits.

Parameters

<i>dataIn</i>	The DataInputStream to read the String data from.
---------------	---

Returns

the String value.

Exceptions

<i>IOException</i>	if an I/O error occurs while writing the string.
--------------------	--

6.337.2.5 static void activemq::util::MarshallingSupport::writeString (decaf::io::DataOutputStream & *dataOut*, const std::string & *value*) [static]

Write the string object to the given DataOutputStream as Raw bytes, no string encoding is done on this char values in the string.

User must encode to Modified UTF-8 as needed.

Parameters

<i>dataOut</i>	The DataOutputStream to write the String data to.
<i>value</i>	Thre String value to write in Openwire form.

Exceptions

<i>IOException</i>	if an I/O error occurs while writing the string.
--------------------	--

6.337.2.6 static void **activemq::util::MarshallingSupport::writeString16** (
decaf::io::DataOutputStream & *dataOut*, const std::string & *value*)
[static]

Write the string object to the given DataOutputStream as Raw bytes, no string encoding is done on this char values in the string.

User must encode to Modified UTF-8 as needed. This method write out only the size as a short and the string data no Openwire Type tag is appended.

Parameters

<i>dataOut</i>	The DataOutputStream to write the String data to.
<i>value</i>	Thre String value to write in Openwire form.

Exceptions

<i>IOException</i>	if an I/O error occurs while writing the string.
--------------------	--

6.337.2.7 static void **activemq::util::MarshallingSupport::writeString32** (
decaf::io::DataOutputStream & *dataOut*, const std::string & *value*)
[static]

Write the string object to the given DataOutputStream as Raw bytes, no string encoding is done on this char values in the string.

User must encode to Modified UTF-8 as needed. This method write out only the size as a int and the string data no Openwire Type tag is appended.

Parameters

<i>dataOut</i>	The DataOutputStream to write the String data to.
<i>value</i>	Thre String value to write in Openwire form.

Exceptions

<i>IOException</i>	if an I/O error occurs while writing the string.
--------------------	--

The documentation for this class was generated from the following file:

- src/main/activemq/util/**MarshallingSupport.h**

6.338 decaf::lang::Math Class Reference

The class **Math** (p. 1800) contains methods for performing basic numeric operations such as the elementary exponential, logarithm, square root, and trigonometric functions.

```
#include <src/main/decaf/lang/Math.h>
```

Public Member Functions

- **Math** ()
- virtual ~**Math** ()

Static Public Member Functions

- static int **abs** (int value)
Returns the absolute value of an int value.
- static long long **abs** (long long value)
Returns the absolute value of an long long value.
- static float **abs** (float value)
Returns the absolute value of a float value.
- static double **abs** (double value)
Returns the absolute value of a double value.
- static double **sqrt** (double value)
Returns the arc cosine of an angle, in the range of 0.0 through pi.
- static double **pow** (double base, double exp)
Returns the value of the first argument raised to the power of the second argument.
- static short **min** (short a, short b)
Returns the double value that is closest in value to the argument and is equal to a mathematical integer.
- static int **min** (int a, int b)
Returns the smaller of two int values.
- static unsigned int **min** (unsigned int a, unsigned int b)
Returns the smaller of two unsigned int values.
- static long long **min** (long long a, long long b)
Returns the smaller of two long long values.
- static float **min** (float a, float b)
Returns the smaller of two float values.
- static double **min** (double a, double b)
Returns the smaller of two double values.
- static short **max** (short a, short b)
Returns the larger of two short values.
- static int **max** (int a, int b)
Returns the larger of two int values.
- static long long **max** (long long a, long long b)

Returns the larger of two long long values.

- static float **max** (float a, float b)

Returns the greater of two float values.

- static double **max** (double a, double b)

Returns the greater of two double values.

- static double **ceil** (double value)

Returns the natural logarithm (base e) of a double value.

- static double **floor** (double value)

Returns the largest (closest to positive infinity) double value that is less than or equal to the argument and is equal to a mathematical integer.

- static int **round** (float value)

Returns the closest int to the argument.

- static long long **round** (double value)

Returns the closest long long to the argument.

- static double **random** ()

Computes the remainder operation on two arguments as prescribed by the IEEE 754 standard.

- static float **signum** (float value)

Returns Euler's number e raised to the power of a double value.

- static double **signum** (double value)

Returns the signum function of the argument; zero if the argument is zero, 1.0f if the argument is greater than zero, -1.0f if the argument is less than zero.

- static double **toRadians** (double angdeg)

Returns the measure in radians of the supplied degree angle.

- static double **toDegrees** (double angrad)

Returns the measure in degrees of the supplied radian angle.

Static Public Attributes

- static const double **E**
- static const double **PI**

6.338.1 Detailed Description

The class **Math** (p. 1800) contains methods for performing basic numeric operations such as the elementary exponential, logarithm, square root, and trigonometric functions.

6.338.2 Constructor & Destructor Documentation

6.338.2.1 **decaf::lang::Math::Math** () [inline]

6.338.2.2 **virtual decaf::lang::Math::~~Math** () [inline, virtual]

6.338.3 Member Function Documentation

6.338.3.1 `static int decaf::lang::Math::abs (int value)` `[inline, static]`

Returns the absolute value of an int value.

If the argument is not negative, the argument is returned. If the argument is negative, the negation of the argument is returned.

Parameters

<i>value</i>	- the value to return the abs of
--------------	----------------------------------

Returns

the value if positive, otherwise the negative of value

6.338.3.2 `static long long decaf::lang::Math::abs (long long value)` `[inline, static]`

Returns the absolute value of an long long value.

If the argument is not negative, the argument is returned. If the argument is negative, the negation of the argument is returned.

Parameters

<i>value</i>	- the value to return the abs of
--------------	----------------------------------

Returns

the value if positive, otherwise the negative of value

6.338.3.3 `static float decaf::lang::Math::abs (float value)` `[static]`

Returns the absolute value of a float value.

If the argument is not negative, the argument is returned. If the argument is negative, the negation of the argument is returned. Special cases:

o If the argument is positive zero or negative zero, the result is positive zero. o If the argument is infinite, the result is positive infinity. o If the argument is NaN, the result is NaN.

In other words, the result is the same as the value of the expression: **Float::intBitsToFloat** (p. 1350)(0x7fffffff & Float::floatToIntBits(*value*))

Parameters

<i>value</i>	- the value to return the abs of
--------------	----------------------------------

Returns

the value if positive, otherwise the negative of value

6.338.3.4 static double decaf::lang::Math::abs (double *value*) [static]

Returns the absolute value of a double value.

If the argument is not negative, the argument is returned. If the argument is negative, the negation of the argument is returned. Special cases:

o If the argument is positive zero or negative zero, the result is positive zero. o If the argument is infinite, the result is positive infinity. o If the argument is NaN, the result is NaN.

In other words, the result is the same as the value of the expression: **Double::long-BitsToDouble** (p. 1243)(0x7fffffffffffffffULL & Double::doubleToLongBits(*value*))

Parameters

<i>value</i>	- the value to return the abs of
--------------	----------------------------------

Returns

the value if positive, otherwise the negative of value

6.338.3.5 static double decaf::lang::Math::ceil (double *value*) [static]

Returns the natural logarithm (base e) of a double value.

Special cases:

o If the argument is NaN or less than zero, then the result is NaN. o If the argument is positive infinity, then the result is positive infinity. o If the argument is positive zero or negative zero, then the result is negative infinity.

Parameters

<i>value</i>	the value to compute the natural log of.
--------------	--

Returns

the natural log of value. Returns the base 10 logarithm of a double value. Special cases:

o If the argument is NaN or less than zero, then the result is NaN. o If the argument is positive infinity, then the result is positive infinity. o If the argument is positive zero or negative zero, then the result is negative infinity. o If the argument is equal to 10ⁿ for integer n, then the result is n.

Parameters

<i>value</i>	- the value to operate on
--------------	---------------------------

Returns

the long base 10 of value Returns the natural logarithm of the sum of the argument and 1. Note that for small values x , the result of $\log_{10}(x)$ is much closer to the true result of $\ln(1 + x)$ than the floating-point evaluation of $\log(1.0+x)$.

Special cases:

o If the argument is NaN or less than -1, then the result is NaN. o If the argument is positive infinity, then the result is positive infinity. o If the argument is negative one, then the result is negative infinity. o If the argument is zero, then the result is a zero with the same sign as the argument.

Parameters

<i>value</i>	- the value to operate on
--------------	---------------------------

Returns

the the value $\ln(x + 1)$, the natural log of $x + 1$ Returns the smallest (closest to negative infinity) double value that is greater than or equal to the argument and is equal to a mathematical integer. Special cases:

o If the argument value is already equal to a mathematical integer, then the result is the same as the argument. o If the argument is NaN or an infinity or positive zero or negative zero, then the result is the same as the argument. o If the argument value is less than zero but greater than -1.0, then the result is negative zero.

Note that the value of `Math.ceil(x)` is exactly the value of `-Math.floor(-x)`.

Parameters

<i>value</i>	- the value to find the ceiling of
--------------	------------------------------------

Returns

the smallest (closest to negative infinity) floating-point value that is greater than or equal to the argument and is equal to a mathematical integer.

6.338.3.6 static double decaf::lang::Math::floor (double *value*) [static]

Returns the largest (closest to positive infinity) double value that is less than or equal to the argument and is equal to a mathematical integer.

Special cases:

o If the argument value is already equal to a mathematical integer, then the result is the same as the argument. o If the argument is NaN or an infinity or positive zero or negative zero, then the result is the same as the argument.

Parameters

<i>value</i>	- the value to find the floor of
--------------	----------------------------------

Returns

the largest (closest to positive infinity) floating-point value that less than or equal to the argument and is equal to a mathematical integer.

6.338.3.7 `static short decaf::lang::Math::max (short a, short b)` [`inline`, `static`]

Returns the larger of two `short` values.

That is, the result the argument closer to the value of **Short::MAX_VALUE** (p. 2408) . If the arguments have the same value, the result is that same value.

Parameters

<i>a</i>	- an argument.
<i>b</i>	- another argument.

Returns

the larger of *a* and *b*.

6.338.3.8 `static int decaf::lang::Math::max (int a, int b)` [`inline`, `static`]

Returns the larger of two `int` values.

That is, the result the argument closer to the value of **Integer::MAX_VALUE** (p. 1516) . If the arguments have the same value, the result is that same value.

Parameters

<i>a</i>	- an argument.
<i>b</i>	- another argument.

Returns

the larger of *a* and *b*.

6.338.3.9 `static long long decaf::lang::Math::max (long long a, long long b)`
`[inline, static]`

Returns the larger of two `long long` values.

That is, the result the argument closer to the value of `Long::MAX_VALUE` (p. 1741). If the arguments have the same value, the result is that same value.

Parameters

<i>a</i>	- an argument.
<i>b</i>	- another argument.

Returns

the larger of *a* and *b*.

6.338.3.10 `static float decaf::lang::Math::max (float a, float b)` `[static]`

Returns the greater of two `float` values.

That is, the result is the argument closer to positive infinity. If the arguments have the same value, the result is that same value. If either value is NaN, then the result is - NaN. Unlike the numerical comparison operators, this method considers negative zero to be strictly smaller than positive zero. If one argument is positive zero and the other negative zero, the result is positive zero.

Parameters

<i>a</i>	- an argument.
<i>b</i>	- another argument.

Returns

the larger of *a* and *b*.

6.338.3.11 `static double decaf::lang::Math::max (double a, double b)` `[static]`

Returns the greater of two `double` values.

That is, the result is the argument closer to positive infinity. If the arguments have the same value, the result is that same value. If either value is NaN, then the result is - NaN. Unlike the numerical comparison operators, this method considers negative zero to be strictly smaller than positive zero. If one argument is positive zero and the other negative zero, the result is positive zero.

Parameters

<i>a</i>	- an argument.
<i>b</i>	- another argument.

Returns

the larger of *a* and *b*.

6.338.3.12 `static short decaf::lang::Math::min (short a, short b)` [`inline`, `static`]

Returns the double value that is closest in value to the argument and is equal to a mathematical integer.

If two double values that are mathematical integers are equally close, the result is the integer value that is even. Special cases:

o If the argument value is already equal to a mathematical integer, then the result is the same as the argument. o If the argument is NaN or an infinity or positive zero or negative zero, then the result is the same as the argument.

Parameters

<i>value</i>	- the value to round to the nearest integer
--------------	---

Returns

the rounded value Returns the smaller of two short values. That is, the result is the argument closer to the value of **decaf.lang.Short::MIN_VALUE** (p. 2408) . If the arguments have the same value, the result is that same value.

Parameters

<i>a</i>	- an argument.
<i>b</i>	- another argument.

Returns

the smaller of *a* and *b*.

Referenced by `decaf::util::concurrent::LinkedBlockingQueue< E >::drainTo()`.

6.338.3.13 `static int decaf::lang::Math::min (int a, int b)` [`inline`, `static`]

Returns the smaller of two `int` values.

That is, the result the argument closer to the value of **Integer::MIN_VALUE** (p. 1516) . If the arguments have the same value, the result is that same value.

Parameters

<i>a</i>	- an argument.
<i>b</i>	- another argument.

Returns

the smaller of `a` and `b`.

6.338.3.14 `static unsigned int decaf::lang::Math::min (unsigned int a, unsigned int b)`
`[inline, static]`

Returns the smaller of two `unsigned int` values.

That is, the result the argument closer to the value of `Integer::MIN_VALUE` (p. 1516). If the arguments have the same value, the result is that same value.

Parameters

<i>a</i>	- an argument.
<i>b</i>	- another argument.

Returns

the smaller of `a` and `b`.

6.338.3.15 `static long long decaf::lang::Math::min (long long a, long long b)`
`[inline, static]`

Returns the smaller of two `long long` values.

That is, the result the argument closer to the value of `Long::MIN_VALUE` (p. 1741). If the arguments have the same value, the result is that same value.

Parameters

<i>a</i>	- an argument.
<i>b</i>	- another argument.

Returns

the smaller of `a` and `b`.

6.338.3.16 `static float decaf::lang::Math::min (float a, float b)` `[static]`

Returns the smaller of two `float` values.

That is, the result is the value closer to negative infinity. If the arguments have the same value, the result is that same value. If either value is NaN, then the result is NaN. Unlike the numerical comparison operators, this method considers negative zero to be strictly smaller than positive zero. If one argument is positive zero and the other is negative zero, the result is negative zero.

Parameters

<i>a</i>	- an argument.
<i>b</i>	- another argument.

Returns

the smaller of *a* and *b*.

6.338.3.17 static double decaf::lang::Math::min (double *a*, double *b*) [static]

Returns the smaller of two double values.

That is, the result is the value closer to negative infinity. If the arguments have the same value, the result is that same value. If either value is NaN, then the result is NaN. Unlike the numerical comparison operators, this method considers negative zero to be strictly smaller than positive zero. If one argument is positive zero and the other is negative zero, the result is negative zero.

Parameters

<i>a</i>	- an argument.
<i>b</i>	- another argument.

Returns

the smaller of *a* and *b*.

6.338.3.18 static double decaf::lang::Math::pow (double *base*, double *exp*) [static]

Returns the value of the first argument raised to the power of the second argument.

Special cases:

- o If the second argument is positive or negative zero, then the result is 1.0.
- o If the second argument is 1.0, then the result is the same as the first argument.
- o If the second argument is NaN, then the result is NaN.
- o If the first argument is NaN and the second argument is nonzero, then the result is NaN.

Parameters

<i>base</i>	- the base
<i>exp</i>	- the exponent

Returns

the base raised to the power of *exp*.

6.338.3.19 static double **decaf::lang::Math::random** () [static]

Computes the remainder operation on two arguments as prescribed by the IEEE 754 standard.

The remainder value is mathematically equal to $f1 - f2 \times n$, where n is the mathematical integer closest to the exact mathematical value of the quotient $f1/f2$, and if two mathematical integers are equally close to $f1/f2$, then n is the integer that is even. If the remainder is zero, its sign is the same as the sign of the first argument. Special cases:

- o If either argument is NaN, or the first argument is infinite, or the second argument is positive zero or negative zero, then the result is NaN.
- o If the first argument is finite and the second argument is infinite, then the result is the same as the first argument.

Parameters

<i>f1</i>	- the dividend.
<i>f2</i>	- the divisor

Returns

the IEEE remainder of value Returns a double value with a positive sign, greater than or equal to 0.0 and less than 1.0. Returned values are chosen pseudorandomly with (approximately) uniform distribution from that range.

When this method is first called, it creates a single new pseudorandom-number generator; This new pseudorandom-number generator is used thereafter for all calls to this method and is used nowhere else.

This method is properly synchronized to allow correct use by more than one thread. However, if many threads need to generate pseudorandom numbers at a great rate, it may reduce contention for each thread to have its own pseudorandom-number generator.

Returns

a pseudorandom double greater than or equal to 0.0 and less than 1.0.

6.338.3.20 static int **decaf::lang::Math::round** (float *value*) [static]

Returns the closest int to the argument.

The result is rounded to an integer by adding 1/2, taking the floor of the result, and casting the result to type int. In other words, the result is equal to the value of the expression: `(int)Math.floor (p. 1805)(a + 0.5f)`

- o If the argument is NaN, the result is 0.
- o If the argument is negative infinity or any value less than or equal to the value of **Integer::MIN_VALUE** (p. 1516), the result is equal to the value of **Integer::MIN_VALUE** (p. 1516).
- o If the argument is positive infinity or any value greater than or equal to the value of **Integer::MAX_VALUE** (p. 1516), the result is equal to the value of **Integer::MAX_VALUE** (p. 1516).

Parameters

<i>value</i>	- the value to round
--------------	----------------------

Returns

the value of the argument rounded to the nearest integral value.

6.338.3.21 `static long long decaf::lang::Math::round (double value) [static]`

Returns the closest long long to the argument.

The result is rounded to an integer by adding 1/2, taking the floor of the result, and casting the result to type long long. In other words, the result is equal to the value of the expression: (long long)**Math.floor** (p. 1805)(a + 0.5d)

o If the argument is NaN, the result is 0. o If the argument is negative infinity or any value less than or equal to the value of **Long::MIN_VALUE** (p. 1741), the result is equal to the value of **Long::MIN_VALUE** (p. 1741). o If the argument is positive infinity or any value greater than or equal to the value of **Long::MAX_VALUE** (p. 1741), the result is equal to the value of **Long::MAX_VALUE** (p. 1741).

Parameters

<i>value</i>	- the value to round
--------------	----------------------

Returns

the value of the argument rounded to the nearest integral value.

6.338.3.22 `static float decaf::lang::Math::signum (float value) [static]`

Returns Euler's number e raised to the power of a double value.

Special cases:

o If the argument is NaN, the result is NaN. o If the argument is positive infinity, then the result is positive infinity. o If the argument is negative infinity, then the result is positive zero.

Parameters

<i>value</i>	- the exponent to raise e to
--------------	------------------------------

Returns

the value e^a , where e is the base of the natural logarithms. Returns $e^x - 1$. Note that for values of x near 0, the exact sum of $\expm1(x) + 1$ is much closer to the true result of $\exp(x)$ than $\exp(x)$. Special cases:

o If the argument is NaN, the result is NaN. o If the argument is positive infinity, then the result is positive infinity. o If the argument is negative infinity, then the result is -1.0. o If the argument is zero, then the result is a zero with the same sign as the argument.

Parameters

<i>value</i>	- the value to raise $e^x - 1$
--------------	--------------------------------

Returns

the value $\exp(x^2 + y^2) - 1$. Returns $\sqrt{x^2 + y^2}$ without intermediate overflow or underflow. Special cases:

If either argument is infinite, then the result is positive infinity. If either argument is NaN and neither argument is infinite, then the result is NaN.

Parameters

<i>x</i>	- an argument
<i>y</i>	- another argument

Returns

the $\sqrt{x^2 + y^2}$ without intermediate overflow or underflow Returns the signum function of the argument; zero if the argument is zero, 1.0f if the argument is greater than zero, -1.0f if the argument is less than zero. Special Cases:

o If the argument is NaN, then the result is NaN. o If the argument is positive zero or negative zero, then the result is the same as the argument.

Parameters

<i>value</i>	- the floating-point value whose signum is to be returned
--------------	---

Returns

the signum function of the argument

6.338.3.23 `static double decaf::lang::Math::signum (double value)` [static]

Returns the signum function of the argument; zero if the argument is zero, 1.0f if the argument is greater than zero, -1.0f if the argument is less than zero.

Special Cases:

o If the argument is NaN, then the result is NaN. o If the argument is positive zero or negative zero, then the result is the same as the argument.

Parameters

<i>value</i>	- the floating-point value whose signum is to be returned
--------------	---

Returns

the signum function of the argument

6.338.3.24 static double decaf::lang::Math::sqrt (double *value*) [static]

Returns the arc cosine of an angle, in the range of 0.0 through pi.

Special case:

o If the argument is NaN or its absolute value is greater than 1, then the result is NaN.

Parameters

<i>value</i>	- the value to return the arc cosine of.
--------------	--

Returns

arc cosine of value in radians. Returns the arc sine of an angle, in the range of -pi/2 through pi/2. Special cases:

o If the argument is NaN or its absolute value is greater than 1, then the result is NaN.

o If the argument is zero, then the result is a zero with the same sign as the argument.

Parameters

<i>value</i>	- the value to return the arc cosine of.
--------------	--

Returns

arc cosine of value in radians. Returns the arc tangent of an angle, in the range of -pi/2 through pi/2. Special cases:

o If the argument is NaN, then the result is NaN. o If the argument is zero, then the result is a zero with the same sign as the argument.

Parameters

<i>value</i>	- the value to return the arc cosine of.
--------------	--

Returns

arc tangent of value in radians. Converts rectangular coordinates (x, y) to polar (r, theta). This method computes the phase theta by computing an arc tangent of y/x in the range of -pi to pi. Special cases:

o If either argument is NaN, then the result is NaN. o If the first argument is positive zero and the second argument is positive, or the first argument is positive and finite and the second argument is positive infinity, then the result is positive zero. o If the first argument is negative zero and the second argument is positive, or the first argument is negative and finite and the second argument is positive infinity, then the result is negative zero. o If the first argument is positive zero and the second argument is negative, or the first argument is positive and finite and the second argument is negative infinity, then the result is the double value closest to pi. o If the first argument is negative zero and the second argument is negative, or the first argument is negative and finite and the second argument is negative infinity, then the result is the double value closest to -pi. o If the first argument is positive and the second argument is positive zero or negative zero, or the first argument is positive infinity and the second argument is finite, then the result is the double value closest to pi/2. o If the first argument is negative and the second argument is positive zero or negative zero, or the first argument is negative infinity and the second argument is finite, then the result is the double value closest to -pi/2. o If both arguments are positive infinity, then the result is the double value closest to pi/4. o If the first argument is positive infinity and the second argument is negative infinity, then the result is the double value closest to 3*pi/4. o If the first argument is negative infinity and the second argument is positive infinity, then the result is the double value closest to -pi/4. o If both arguments are negative infinity, then the result is the double value closest to -3*pi/4.

Parameters

<i>y</i>	- the ordinate coordinate
<i>x</i>	- the abscissa coordinate

Returns

the theta component of the point (r, theta) in polar coordinates that corresponds to the point (x, y) in Cartesian coordinates. Returns the cube root of a double value. For positive finite x, $\text{cbrt}(-x) == -\text{cbrt}(x)$; that is, the cube root of a negative value is the negative of the cube root of that value's magnitude. Special cases:

o If the argument is NaN, then the result is NaN. o If the argument is infinite, then the result is an infinity with the same sign as the argument. o If the argument is zero, then the result is a zero with the same sign as the argument.

Parameters

<i>value</i>	- the double to compute the cube root of
--------------	--

Returns

the cube root of value Returns the trigonometric cosine of an angle. Special cases:

o If the argument is NaN or an infinity, then the result is NaN.

Parameters

<i>value</i>	- an value in radians
--------------	-----------------------

Returns

the cosine of the argument. Returns the hyperbolic cosine of a double value. -
The hyperbolic cosine of x is defined to be $(e^x + e^{-x})/2$ where e is Euler's number.
Special cases:

o If the argument is NaN, then the result is NaN. o If the argument is infinite, then the result is positive infinity. o If the argument is zero, then the result is 1.0.

Parameters

<i>value</i>	- the number whose hyperbolic cosine is to be found
--------------	---

Returns

the hyperbolic cosine of value Returns the trigonometric sine of an angle. Special case:

o If the argument is NaN or an infinity, then the result is NaN. o If the argument is zero, then the result is a zero with the same sign as the argument.

Parameters

<i>value</i>	- the number whose sin is to be found
--------------	---------------------------------------

Returns

the sine of value Returns the hyperbolic sine of a double value. The hyperbolic sine of x is defined to be $(e^x - e^{-x})/2$ where e is Euler's number. Special cases:

o If the argument is NaN, then the result is NaN. o If the argument is infinite, then the result is an infinity with the same sign as the argument. o If the argument is zero, then the result is a zero with the same sign as the argument.

Parameters

<i>value</i>	- the number whose hyperbolic sin is to be found
--------------	--

Returns

the hyperbolic sine of value Returns the trigonometric tangent of an angle. Special cases:

o If the argument is NaN or an infinity, then the result is NaN. o If the argument is zero, then the result is a zero with the same sign as the argument.

Parameters

<i>value</i>	- the number whose tangent is to be found
--------------	---

Returns

the tangent of value Returns the hyperbolic tangent of a double value. The hyperbolic tangent of x is defined to be $(e^x - e^{-x}) / (e^x + e^{-x})$, in other words, $\sinh(x) / \cosh(x)$. Note that the absolute value of the exact \tanh is always less than 1. Special cases:

o If the argument is NaN, then the result is NaN. o If the argument is zero, then the result is a zero with the same sign as the argument. o If the argument is positive infinity, then the result is +1.0. o If the argument is negative infinity, then the result is -1.0.

Parameters

<i>value</i>	- the number whose hyperbolic tangent is to be found
--------------	--

Returns

the hyperbolic cosine of value Returns the correctly rounded positive square root of a double value. Special cases:

o If the argument is NaN or less than zero, then the result is NaN. o If the argument is positive infinity, then the result is positive infinity. o If the argument is positive zero or negative zero, then the result is the same as the argument.

Otherwise, the result is the double value closest to the true mathematical square root of the argument value.

Parameters

<i>value</i>	- the value to find the square root of
<i>the</i>	square root of the argument.

6.338.3.25 `static double decaf::lang::Math::toDegrees (double angrad)` `[inline, static]`

Returns the measure in degrees of the supplied radian angle.

Parameters

<i>angrad</i>	- an angle in radians
---------------	-----------------------

Returns

the degree measure of the angle.

6.338.3.26 `static double decaf::lang::Math::toRadians (double angdeg)` [`inline`, `static`]

Returns the measure in radians of the supplied degree angle.

Parameters

<i>angdeg</i>	- an angle in degrees
---------------	-----------------------

Returns

the radian measure of the angle.

6.338.4 Field Documentation

6.338.4.1 `const double decaf::lang::Math::E` [`static`]

6.338.4.2 `const double decaf::lang::Math::PI` [`static`]

The documentation for this class was generated from the following file:

- `src/main/decaf/lang/Math.h`

6.339 activemq::util::MemoryUsage Class Reference

```
#include <src/main/activemq/util/MemoryUsage.h>
```

Inheritance diagram for `activemq::util::MemoryUsage`:

Public Member Functions

- **MemoryUsage** ()
Default Constructor.
- **MemoryUsage** (unsigned long long limit)
*Creates an instance of an **Usage** (p. 2872) monitor with a set limit.*

- virtual **~MemoryUsage** ()
- virtual void **waitForSpace** ()
 - Waits forever for more space to be returned to this **Usage** (p. 2872) Manager.*
- virtual void **waitForSpace** (unsigned int timeout)
 - Waits for more space to be returned to this **Usage** (p. 2872) Manager, times out when the given time span in milliseconds elapses.*
- virtual void **enqueueUsage** (unsigned long long value)
 - Tries to increase the usage by value amount but blocks if this object is currently full.*
- virtual void **increaseUsage** (unsigned long long value)
 - Increases the usage by the value amount.*
- virtual void **decreaseUsage** (unsigned long long value)
 - Decreases the usage by the value amount.*
- virtual bool **isFull** () const
 - Returns true if this **Usage** (p. 2872) instance is full, i.e.*
- unsigned long long **getUsage** () const
 - Gets the current usage amount.*
- void **setUsage** (unsigned long long usage)
 - Sets the current usage amount.*
- unsigned long long **getLimit** () const
 - Gets the current limit amount.*
- void **setLimit** (unsigned long long limit)
 - Sets the current limit amount.*

6.339.1 Constructor & Destructor Documentation

6.339.1.1 activemq::util::MemoryUsage::MemoryUsage ()

Default Constructor.

6.339.1.2 activemq::util::MemoryUsage::MemoryUsage (unsigned long long *limit*)

Creates an instance of an **Usage** (p. 2872) monitor with a set limit.

Parameters

<i>limit</i>	- amount of memory this manager allows.
--------------	---

6.339.1.3 virtual activemq::util::MemoryUsage::~~MemoryUsage () [virtual]

6.339.2 Member Function Documentation

6.339.2.1 virtual void **activemq::util::MemoryUsage::decreaseUsage** (unsigned long long *value*) [virtual]

Decreases the usage by the value amount.

Parameters

<i>value</i>	Amount of space to return to the pool
--------------	---------------------------------------

Implements **activemq::util::Usage** (p. 2873).

6.339.2.2 virtual void **activemq::util::MemoryUsage::enqueueUsage** (unsigned long long *value*) [inline, virtual]

Tries to increase the usage by value amount but blocks if this object is currently full.

Parameters

<i>value</i>	Amount of usage in bytes to add.
--------------	----------------------------------

Implements **activemq::util::Usage** (p. 2873).

6.339.2.3 unsigned long long **activemq::util::MemoryUsage::getLimit** () const [inline]

Gets the current limit amount.

Returns

the amount that can be used before full.

6.339.2.4 unsigned long long **activemq::util::MemoryUsage::getUsage** () const [inline]

Gets the current usage amount.

Returns

the amount of bytes currently used.

6.339.2.5 virtual void **activemq::util::MemoryUsage::increaseUsage** (unsigned long long *value*) [virtual]

Increases the usage by the value amount.

Parameters

<i>value</i>	Amount of usage to add.
--------------	-------------------------

Implements **activemq::util::Usage** (p. 2873).

6.339.2.6 **virtual bool activemq::util::MemoryUsage::isFull () const** [virtual]

Returns true if this **Usage** (p. 2872) instance is full, i.e.

Usage (p. 2872) $\geq 100\%$

Implements **activemq::util::Usage** (p. 2873).

6.339.2.7 **void activemq::util::MemoryUsage::setLimit (unsigned long long *limit*)**
[inline]

Sets the current limit amount.

Parameters

<i>limit</i>	- The amount that can be used before full.
--------------	--

6.339.2.8 **void activemq::util::MemoryUsage::setUsage (unsigned long long *usage*)**
[inline]

Sets the current usage amount.

Parameters

<i>usage</i>	- The amount to tag as used.
--------------	------------------------------

6.339.2.9 **virtual void activemq::util::MemoryUsage::waitForSpace ()**
[virtual]

Waits forever for more space to be returned to this **Usage** (p. 2872) Manager.

Implements **activemq::util::Usage** (p. 2874).

6.339.2.10 **virtual void activemq::util::MemoryUsage::waitForSpace (unsigned int *timeout*)** [virtual]

Waits for more space to be returned to this **Usage** (p. 2872) Manager, times out when the given time span in milliseconds elapses.

Parameters

<i>timeout</i>	The time to wait for more space.
----------------	----------------------------------

Implements **activemq::util::Usage** (p. 2874).

The documentation for this class was generated from the following file:

- src/main/activemq/util/**MemoryUsage.h**

6.340 activemq::commands::Message Class Reference

```
#include <src/main/activemq/commands/Message.h>
```

Inheritance diagram for activemq::commands::Message:

Public Member Functions

- **Message** ()
- virtual **~Message** ()
- virtual unsigned char **getDataStructureType** () const
*Get the **DataStructure** (p. 1133) Type as defined in CommandTypes.h.*
- virtual **Message** * **cloneDataStructure** () const
Clone this object and return a new instance that the caller now owns, this will be an exact copy of this one.
- virtual void **copyDataStructure** (const **DataStructure** *src)
Copy the contents of the passed object into this objects members, overwriting any existing data.
- virtual std::string **toString** () const
*Returns a string containing the information for this **DataStructure** (p. 1133) such as its type and value of its elements.*
- virtual bool **equals** (const **DataStructure** *value) const
*Compares the **DataStructure** (p. 1133) passed in to this one, and returns if they are equivalent.*
- virtual void **beforeMarshal** (wireformat::WireFormat *wireFormat AMQCPP_UNUSED)
Handles the marshaling of the objects properties into the internal byte array before the object is marshaled to the wire.
- virtual void **afterUnmarshal** (wireformat::WireFormat *wireFormat AMQCPP_UNUSED)
Called after unmarshaling is started to cleanup the object being unmarshaled.
- virtual bool **isMarshalAware** () const
Indicates that this command is aware of Marshaling, and needs to have its Marshaling methods invoked.

- virtual void **setAckHandler** (const **Pointer**< **core::ActiveMQAckHandler** > &handler)
*Sets the Acknowledgment Handler that this **Message** (p. 1821) will use when the - Acknowledge method is called.*
- virtual **Pointer** < **core::ActiveMQAckHandler** > **getAckHandler** () const
*Gets the Acknowledgment Handler that this **Message** (p. 1821) will use when the - Acknowledge method is called.*
- void **setConnection** (**core::ActiveMQConnection** *connection)
*Sets the ActiveMQConnection instance that this **Command** (p. 866) was created from when the session create methods are called to create a **Message** (p. 1821).*
- **core::ActiveMQConnection** * **getConnection** () const
*Gets the ActiveMQConnection instance that this **Command** (p. 866) was created from when the session create methods are called to create a **Message** (p. 1821).*
- virtual unsigned int **getSize** () const
Returns the Size of this message in Bytes.
- virtual bool **isExpired** () const
Returns if this message has expired, meaning that its Expiration time has elapsed.
- virtual void **onSend** ()
*Allows derived **Message** (p. 1821) classes to perform tasks before a message is sent.*
- **util::PrimitiveMap** & **getMessageProperties** ()
*Gets a reference to the **Message** (p. 1821)'s Properties object, allows the derived classes to get and set their own specific properties.*
- const **util::PrimitiveMap** & **getMessageProperties** () const
- bool **isReadOnlyProperties** () const
*Returns if the **Message** (p. 1821) Properties Are Read Only.*
- void **setReadOnlyProperties** (bool value)
*Set the Read Only State of the **Message** (p. 1821) Properties.*
- bool **isReadOnlyBody** () const
*Returns if the **Message** (p. 1821) Body is Read Only.*
- void **setReadOnlyBody** (bool value)
*Set the Read Only State of the **Message** (p. 1821) Content.*
- virtual const **Pointer** < **ProducerId** > & **getProducerId** () const
- virtual **Pointer**< **ProducerId** > & **getProducerId** ()
- virtual void **setProducerId** (const **Pointer**< **ProducerId** > &producerId)
- virtual const **Pointer** < **ActiveMQDestination** > & **getDestination** () const
- virtual **Pointer** < **ActiveMQDestination** > & **getDestination** ()
- virtual void **setDestination** (const **Pointer**< **ActiveMQDestination** > &destination)
tion)
- virtual const **Pointer** < **TransactionId** > & **getTransactionId** () const
- virtual **Pointer**< **TransactionId** > & **getTransactionId** ()
- virtual void **setTransactionId** (const **Pointer**< **TransactionId** > &transactionId)
Id)
- virtual const **Pointer** < **ActiveMQDestination** > & **getOriginalDestination** () const
- virtual **Pointer** < **ActiveMQDestination** > & **getOriginalDestination** ()

- virtual void **setOriginalDestination** (const **Pointer**< **ActiveMQDestination** > &**originalDestination**)
- virtual const **Pointer** < **MessageId** > & **getMessageId** () const
- virtual **Pointer**< **MessageId** > & **getMessageId** ()
- virtual void **setMessageId** (const **Pointer**< **MessageId** > &**messageId**)
- virtual const **Pointer** < **TransactionId** > & **getOriginalTransactionId** () const
- virtual **Pointer**< **TransactionId** > & **getOriginalTransactionId** ()
- virtual void **setOriginalTransactionId** (const **Pointer**< **TransactionId** > &**originalTransactionId**)
- virtual const std::string & **getGroupId** () const
- virtual std::string & **getGroupId** ()
- virtual void **setGroupId** (const std::string &**groupId**)
- virtual int **getGroupSequence** () const
- virtual void **setGroupSequence** (int **groupSequence**)
- virtual const std::string & **getCorrelationId** () const
- virtual std::string & **getCorrelationId** ()
- virtual void **setCorrelationId** (const std::string &**correlationId**)
- virtual bool **isPersistent** () const
- virtual void **setPersistent** (bool **persistent**)
- virtual long long **getExpiration** () const
- virtual void **setExpiration** (long long **expiration**)
- virtual unsigned char **getPriority** () const
- virtual void **setPriority** (unsigned char **priority**)
- virtual const **Pointer** < **ActiveMQDestination** > & **getReplyTo** () const
- virtual **Pointer** < **ActiveMQDestination** > & **getReplyTo** ()
- virtual void **setReplyTo** (const **Pointer**< **ActiveMQDestination** > &**replyTo**)
- virtual long long **getTimestamp** () const
- virtual void **setTimestamp** (long long **timestamp**)
- virtual const std::string & **getType** () const
- virtual std::string & **getType** ()
- virtual void **setType** (const std::string &**type**)
- virtual const std::vector< unsigned char > & **getContent** () const
- virtual std::vector< unsigned char > & **getContent** ()
- virtual void **setContent** (const std::vector< unsigned char > &**content**)
- virtual const std::vector< unsigned char > & **getMarshaledProperties** () const
- virtual std::vector< unsigned char > & **getMarshaledProperties** ()
- virtual void **setMarshaledProperties** (const std::vector< unsigned char > &**marshalledProperties**)
- virtual const **Pointer** < **DataStructure** > & **getDataStructure** () const
- virtual **Pointer**< **DataStructure** > & **getDataStructure** ()
- virtual void **setDataStructure** (const **Pointer**< **DataStructure** > &**data-Structure**)
- virtual const **Pointer** < **ConsumerId** > & **getTargetConsumerId** () const
- virtual **Pointer**< **ConsumerId** > & **getTargetConsumerId** ()
- virtual void **setTargetConsumerId** (const **Pointer**< **ConsumerId** > &**target-ConsumerId**)
- virtual bool **isCompressed** () const

- virtual void **setCompressed** (bool **compressed**)
- virtual int **getRedeliveryCounter** () const
- virtual void **setRedeliveryCounter** (int **redeliveryCounter**)
- virtual const std::vector < **decaf::lang::Pointer** < **BrokerId** > > & **getBrokerPath** () const
- virtual std::vector < **decaf::lang::Pointer** < **BrokerId** > > & **getBrokerPath** ()
- virtual void **setBrokerPath** (const std::vector< **decaf::lang::Pointer**< **BrokerId** > > &**brokerPath**)
- virtual long long **getArrival** () const
- virtual void **setArrival** (long long **arrival**)
- virtual const std::string & **getUserID** () const
- virtual std::string & **getUserID** ()
- virtual void **setUserID** (const std::string &**userID**)
- virtual bool **isRecievedByDFBridge** () const
- virtual void **setRecievedByDFBridge** (bool **recievedByDFBridge**)
- virtual bool **isDroppable** () const
- virtual void **setDroppable** (bool **droppable**)
- virtual const std::vector < **decaf::lang::Pointer** < **BrokerId** > > & **getCluster** () const
- virtual std::vector < **decaf::lang::Pointer** < **BrokerId** > > & **getCluster** ()
- virtual void **setCluster** (const std::vector< **decaf::lang::Pointer**< **BrokerId** > > &**cluster**)
- virtual long long **getBrokerInTime** () const
- virtual void **setBrokerInTime** (long long **brokerInTime**)
- virtual long long **getBrokerOutTime** () const
- virtual void **setBrokerOutTime** (long long **brokerOutTime**)
- virtual bool **isMessage** () const
- virtual **Pointer**< **Command** > **visit** (**activemq::state::CommandVisitor** *visitor)

Allows a Visitor to visit this command and return a response to the command based on the command type being visited.

Static Public Attributes

- static const unsigned char **ID_MESSAGE** = 0

Protected Attributes

- **Pointer**< **ProducerId** > **producerId**
- **Pointer**< **ActiveMQDestination** > **destination**
- **Pointer**< **TransactionId** > **transactionId**
- **Pointer**< **ActiveMQDestination** > **originalDestination**
- **Pointer**< **MessageId** > **messageId**
- **Pointer**< **TransactionId** > **originalTransactionId**
- std::string **groupId**
- int **groupSequence**

- std::string **correlationId**
- bool **persistent**
- long long **expiration**
- unsigned char **priority**
- **Pointer< ActiveMQDestination > replyTo**
- long long **timestamp**
- std::string **type**
- std::vector< unsigned char > **content**
- std::vector< unsigned char > **marshalledProperties**
- **Pointer< DataStructure > dataStructure**
- **Pointer< ConsumerId > targetConsumerId**
- bool **compressed**
- int **redeliveryCounter**
- std::vector< decaf::lang::Pointer< BrokerId > > **brokerPath**
- long long **arrival**
- std::string **userId**
- bool **recievedByDFBridge**
- bool **droppable**
- std::vector< decaf::lang::Pointer< BrokerId > > **cluster**
- long long **brokerInTime**
- long long **brokerOutTime**
- **core::ActiveMQConnection * connection**

Static Protected Attributes

- static const unsigned int **DEFAULT_MESSAGE_SIZE** = 1024

6.340.1 Constructor & Destructor Documentation

6.340.1.1 `activemq::commands::Message::Message ()`

6.340.1.2 `virtual activemq::commands::Message::~~Message ()` [virtual]

6.340.2 Member Function Documentation

6.340.2.1 `virtual void activemq::commands::Message::afterUnmarshal (wireformat::WireFormat *wireFormat AMQCPP_UNUSED)` [virtual]

Called after unmarshaling is started to cleanup the object being unmarshaled.

Parameters

<i>wireFormat</i>	- the wireformat object to control unmarshaling
-------------------	---

Reimplemented from **activemq::commands::BaseDataStructure** (p. 529).

6.340.2.2 `virtual void activemq::commands::Message::beforeMarshal (wireformat::WireFormat *wireFormat AMQCPP_UNUSED) [virtual]`

Handles the marshaling of the objects properties into the internal byte array before the object is marshaled to the wire.

Parameters

<i>wireFormat</i>	- the wireformat controller
-------------------	-----------------------------

Reimplemented from **activemq::commands::BaseDataStructure** (p. 530).

6.340.2.3 `virtual Message* activemq::commands::Message::cloneDataStructure ()const [virtual]`

Clone this object and return a new instance that the caller now owns, this will be an exact copy of this one.

Returns

new copy of this object.

Implements **activemq::commands::DataStructure** (p. 1133).

Reimplemented in **activemq::commands::ActiveMQBytesMessage** (p. 169), **activemq::commands::ActiveMQStreamMessage** (p. 362), **activemq::commands::ActiveMQMapMessage** (p. 272), **activemq::commands::ActiveMQBlobMessage** (p. 158), **activemq::commands::ActiveMQTextMessage** (p. 407), **activemq::commands::ActiveMQMessage** (p. 289), and **activemq::commands::ActiveMQObjectMessage** (p. 302).

6.340.2.4 `virtual void activemq::commands::Message::copyDataStructure (const DataStructure *src) [virtual]`

Copy the contents of the passed object into this objects members, overwriting any existing data.

Returns

src - Source Object

Reimplemented from **activemq::commands::BaseCommand** (p. 493).

Reimplemented in **activemq::commands::ActiveMQBytesMessage** (p. 169), **activemq::commands::ActiveMQStreamMessage** (p. 362), **activemq::commands::ActiveMQMapMessage** (p. 272), **activemq::commands::ActiveMQBlobMessage** (p. 158), **activemq::commands::ActiveMQTextMessage** (p. 407), **activemq::commands::ActiveMQObjectMessage** (p. 302), and **activemq::commands::ActiveMQMessage** (p. 289).

6.340.2.5 `virtual bool activemq::commands::Message::equals (const DataStructure
* value) const [virtual]`

Compares the **DataStructure** (p. 1133) passed in to this one, and returns if they are equivalent.

Equivalent here means that they are of the same type, and that each element of the objects are the same.

Returns

true if **DataStructure** (p. 1133)'s are Equal.

Reimplemented from **activemq::commands::BaseCommand** (p. 494).

Reimplemented in **activemq::commands::ActiveMQBytesMessage** (p. 169), **activemq::commands::ActiveMQStreamMessage** (p. 362), **activemq::commands::ActiveMQMapMessage** (p. 272), **activemq::commands::ActiveMQMessageTemplate< T >** (p. 297), **activemq::commands::ActiveMQMessageTemplate< cms::BytesMessage >** (p. 297), **activemq::commands::ActiveMQMessageTemplate< cms::MapMessage >** (p. 297), **activemq::commands::ActiveMQMessageTemplate< cms::Message >** (p. 297), **activemq::commands::ActiveMQMessageTemplate< cms::StreamMessage >** (p. 297), **activemq::commands::ActiveMQMessageTemplate< cms::TextMessage >** (p. 297), **activemq::commands::ActiveMQMessageTemplate< cms::ObjectMessage >** (p. 297), **activemq::commands::ActiveMQBlobMessage** (p. 159), **activemq::commands::ActiveMQTextMessage** (p. 408), **activemq::commands::ActiveMQObjectMessage** (p. 303), and **activemq::commands::ActiveMQMessage** (p. 290).

6.340.2.6 `virtual Pointer<core::ActiveMQAckHandler> activemq-
::commands::Message::getAckHandler () const [inline,
virtual]`

Gets the Acknowledgment Handler that this **Message** (p. 1821) will use when the Acknowledge method is called.

Returns

handler ActiveMQAckHandler to call or NULL if not set

6.340.2.7 `virtual long long activemq::commands::Message::getArrival () const
[virtual]`

6.340.2.8 `virtual long long activemq::commands::Message::getBrokerInTime ()
const [virtual]`

6.340.2.9 `virtual long long activemq::commands::Message::getBrokerOutTime ()
const [virtual]`

- 6.340.2.10 `virtual const std::vector< decaf::lang::Pointer<BrokerId> >& activemq::commands::Message::getBrokerPath () const` [virtual]
- 6.340.2.11 `virtual std::vector< decaf::lang::Pointer<BrokerId> >& activemq::commands::Message::getBrokerPath ()` [virtual]
- 6.340.2.12 `virtual const std::vector< decaf::lang::Pointer<BrokerId> >& activemq::commands::Message::getCluster () const` [virtual]
- 6.340.2.13 `virtual std::vector< decaf::lang::Pointer<BrokerId> >& activemq::commands::Message::getCluster ()` [virtual]
- 6.340.2.14 `core::ActiveMQConnection* activemq::commands::Message::getConnection () const` [inline]

Gets the ActiveMQConnection instance that this **Command** (p. 866) was created from when the session create methods are called to create a **Message** (p. 1821).

Returns

the ActiveMQConnection parent for this **Message** (p. 1821) or NULL if not set.

- 6.340.2.15 `virtual const std::vector<unsigned char>& activemq::commands::Message::getContent () const` [virtual]
- 6.340.2.16 `virtual std::vector<unsigned char>& activemq::commands::Message::getContent ()` [virtual]
- 6.340.2.17 `virtual const std::string& activemq::commands::Message::getCorrelationId () const` [virtual]
- 6.340.2.18 `virtual std::string& activemq::commands::Message::getCorrelationId ()` [virtual]
- 6.340.2.19 `virtual const Pointer<DataStructure>& activemq::commands::Message::getDataStructure () const` [virtual]
- 6.340.2.20 `virtual Pointer<DataStructure>& activemq::commands::Message::getDataStructure ()` [virtual]
- 6.340.2.21 `virtual unsigned char activemq::commands::Message::getDataStructureType () const` [virtual]

Get the **DataStructure** (p. 1133) Type as defined in CommandTypes.h.

Returns

The type of the data structure

Implements **activemq::commands::DataStructure** (p. 1137).

Reimplemented in **activemq::commands::ActiveMQBytesMessage** (p. 170), **activemq::commands::ActiveMQStreamMessage** (p. 363), **activemq::commands::ActiveMQMapMessage** (p. 274), **activemq::commands::ActiveMQBlobMessage** (p. 159), **activemq::commands::ActiveMQTextMessage** (p. 408), **activemq::commands::ActiveMQObjectMessage** (p. 303), and **activemq::commands::ActiveMQMessage** (p. 290).

- 6.340.2.22 `virtual const Pointer<ActiveMQDestination>& activemq::commands::Message::getDestination () const`
[virtual]
- 6.340.2.23 `virtual Pointer<ActiveMQDestination>& activemq::commands::Message::getDestination ()`
[virtual]
- 6.340.2.24 `virtual long long activemq::commands::Message::getExpiration () const`
[virtual]
- 6.340.2.25 `virtual const std::string& activemq::commands::Message::getGroupID () const` [virtual]
- 6.340.2.26 `virtual std::string& activemq::commands::Message::getGroupID ()`
[virtual]
- 6.340.2.27 `virtual int activemq::commands::Message::getGroupSequence () const`
[virtual]
- 6.340.2.28 `virtual const std::vector<unsigned char>& activemq::commands::Message::getMarshallledProperties () const`
[virtual]
- 6.340.2.29 `virtual std::vector<unsigned char>& activemq::commands::Message::getMarshallledProperties ()` [virtual]
- 6.340.2.30 `virtual const Pointer<MessageId>& activemq::commands::Message::getMessageId () const`
[virtual]
- 6.340.2.31 `virtual Pointer<MessageId>& activemq::commands::Message::getMessageId ()` [virtual]

6.340.2.32 **util::PrimitiveMap& activemq::commands::Message::getMessageProperties ()** [inline]

Gets a reference to the **Message** (p.1821)'s Properties object, allows the derived classes to get and set their own specific properties.

Returns

a reference to the Primitive Map that holds message properties.

6.340.2.33 **const util::PrimitiveMap& activemq::commands::Message::getMessageProperties ()** const [inline]

6.340.2.34 **virtual const Pointer<ActiveMQDestination>& activemq::commands::Message::getOriginalDestination ()** const [virtual]

6.340.2.35 **virtual Pointer<ActiveMQDestination>& activemq::commands::Message::getOriginalDestination ()** [virtual]

6.340.2.36 **virtual const Pointer<TransactionId>& activemq::commands::Message::getOriginalTransactionId ()** const [virtual]

6.340.2.37 **virtual Pointer<TransactionId>& activemq::commands::Message::getOriginalTransactionId ()** [virtual]

6.340.2.38 **virtual unsigned char activemq::commands::Message::getPriority ()** const [virtual]

6.340.2.39 **virtual const Pointer<ProducerId>& activemq::commands::Message::getProducerId ()** const [virtual]

6.340.2.40 **virtual Pointer<ProducerId>& activemq::commands::Message::getProducerId ()** [virtual]

6.340.2.41 **virtual int activemq::commands::Message::getRedeliveryCounter ()** const [virtual]

6.340.2.42 **virtual const Pointer<ActiveMQDestination>& activemq::commands::Message::getReplyTo ()** const [virtual]

6.340.2.43 **virtual Pointer<ActiveMQDestination>& activemq::commands::Message::getReplyTo ()** [virtual]

6.340.2.44 `virtual unsigned int activemq::commands::Message::getSize () const`
[virtual]

Returns the Size of this message in Bytes.

Returns

number of bytes this message equates to.

Reimplemented in **activemq::commands::ActiveMQTextMessage** (p. 408).

6.340.2.45 `virtual const Pointer<ConsumerId>& activemq-
::commands::Message::getTargetConsumerId () const`
[virtual]

6.340.2.46 `virtual Pointer<ConsumerId>& activemq::commands::Message::get-
TargetConsumerId ()` [virtual]

6.340.2.47 `virtual long long activemq::commands::Message::getTimestamp () const`
[virtual]

6.340.2.48 `virtual const Pointer<TransactionId>& activemq-
::commands::Message::getTransactionId () const`
[virtual]

6.340.2.49 `virtual Pointer<TransactionId>& activemq::commands::Message::get-
TransactionId ()` [virtual]

6.340.2.50 `virtual const std::string& activemq::commands::Message::getType ()
const` [virtual]

6.340.2.51 `virtual std::string& activemq::commands::Message::getType ()`
[virtual]

6.340.2.52 `virtual const std::string& activemq::commands::Message::getUserID ()
const` [virtual]

6.340.2.53 `virtual std::string& activemq::commands::Message::getUserID ()`
[virtual]

6.340.2.54 `virtual bool activemq::commands::Message::isCompressed () const`
[virtual]

6.340.2.55 `virtual bool activemq::commands::Message::isDroppable () const`
[virtual]

6.340.2.56 `virtual bool activemq::commands::Message::isExpired () const`
[virtual]

Returns if this message has expired, meaning that its Expiration time has elapsed.

Returns

true if message is expired.

6.340.2.57 `virtual bool activemq::commands::Message::isMarshalAware () const`
[inline, virtual]

Indicates that this command is aware of Marshaling, and needs to have its Marshaling methods invoked.

Returns

boolean indicating desire to be in marshaling stages

Reimplemented from **activemq::commands::BaseDataStructure** (p. 530).

Reimplemented in **activemq::commands::ActiveMQMapMessage** (p. 278).

6.340.2.58 `virtual bool activemq::commands::Message::isMessage () const`
[inline, virtual]

Returns

an answer of true to the **isMessage()** (p. 1832) query.

Reimplemented from **activemq::commands::BaseCommand** (p. 496).

6.340.2.59 `virtual bool activemq::commands::Message::isPersistent () const`
[virtual]

6.340.2.60 `bool activemq::commands::Message::isReadOnlyBody () const`
[inline]

Returns if the **Message** (p. 1821) Body is Read Only.

Returns

true if **Message** (p. 1821) Content is Read Only.

6.340.2.61 `bool activemq::commands::Message::isReadOnlyProperties () const`
[inline]

Returns if the **Message** (p. 1821) Properties Are Read Only.

Returns

true if **Message** (p. 1821) Properties are Read Only.

6.340.2.62 `virtual bool activemq::commands::Message::isRecievedByDFBridge ()`
`const [virtual]`

6.340.2.63 `virtual void activemq::commands::Message::onSend ()` `[inline,`
`virtual]`

Allows derived **Message** (p. 1821) classes to perform tasks before a message is sent.

Reimplemented in **activemq::commands::ActiveMQBytesMessage** (p. 171), **activemq::commands::ActiveMQStreamMessage** (p. 363), **activemq::commands::ActiveMQMessageTemplate< T >** (p. 299), **activemq::commands::ActiveMQMessageTemplate< cms::BytesMessage >** (p. 299), **activemq::commands::ActiveMQMessageTemplate< cms::MapMessage >** (p. 299), **activemq::commands::ActiveMQMessageTemplate< cms::Message >** (p. 299), **activemq::commands::ActiveMQMessageTemplate< cms::StreamMessage >** (p. 299), **activemq::commands::ActiveMQMessageTemplate< cms::TextMessage >** (p. 299), and **activemq::commands::ActiveMQMessageTemplate< cms::ObjectMessage >** (p. 299).

6.340.2.64 `virtual void activemq::commands::Message::setAckHandler (const`
`Pointer< core::ActiveMQAckHandler > & handler)` `[inline,`
`virtual]`

Sets the Acknowledgment Handler that this **Message** (p. 1821) will use when the - Acknowledge method is called.

Parameters

<i>handler</i>	ActiveMQAckHandler to call
----------------	----------------------------

6.340.2.65 `virtual void activemq::commands::Message::setArrival (long long arrival)`
`[virtual]`

6.340.2.66 `virtual void activemq::commands::Message::setBrokerInTime (long long`
`brokerInTime)` `[virtual]`

6.340.2.67 `virtual void activemq::commands::Message::setBrokerOutTime (long`
`long brokerOutTime)` `[virtual]`

6.340.2.68 `virtual void activemq::commands::Message::setBrokerPath (const`
`std::vector< decaf::lang::Pointer< BrokerId > > & brokerPath)`
`[virtual]`

6.340.2.69 `virtual void activemq::commands::Message::setCluster (const
std::vector< decaf::lang::Pointer< BrokerId > > & cluster) [virtual]`

6.340.2.70 `virtual void activemq::commands::Message::setCompressed (bool
compressed) [virtual]`

6.340.2.71 `void activemq::commands::Message::setConnection (
core::ActiveMQConnection * connection) [inline]`

Sets the ActiveMQConnection instance that this **Command** (p. 866) was created from when the session create methods are called to create a **Message** (p. 1821).

Parameters

<i>handler</i>	ActiveMQConnection parent for this message
----------------	--

6.340.2.72 `virtual void activemq::commands::Message::setContent (const
std::vector< unsigned char > & content) [virtual]`

6.340.2.73 `virtual void activemq::commands::Message::setCorrelationId (const
std::string & correlationId) [virtual]`

6.340.2.74 `virtual void activemq::commands::Message::setDataStructure (const
Pointer< DataStructure > & dataStructure) [virtual]`

6.340.2.75 `virtual void activemq::commands::Message::setDestination (const
Pointer< ActiveMQDestination > & destination) [virtual]`

6.340.2.76 `virtual void activemq::commands::Message::setDroppable (bool
droppable) [virtual]`

6.340.2.77 `virtual void activemq::commands::Message::setExpiration (long long
expiration) [virtual]`

6.340.2.78 `virtual void activemq::commands::Message::setGroupID (const std::string
& groupId) [virtual]`

6.340.2.79 `virtual void activemq::commands::Message::setGroupSequence (int
groupSequence) [virtual]`

6.340.2.80 `virtual void activemq::commands::Message::setMarshaledProperties (
const std::vector< unsigned char > & marshalledProperties) [virtual]`

6.340.2.81 `virtual void activemq::commands::Message::setMessageId (const
Pointer< MessageId > & messageId) [virtual]`

- 6.340.2.82 `virtual void activemq::commands::Message::setOriginalDestination (const Pointer< ActiveMQDestination > & originalDestination) [virtual]`
- 6.340.2.83 `virtual void activemq::commands::Message::setOriginalTransactionId (const Pointer< TransactionId > & originalTransactionId) [virtual]`
- 6.340.2.84 `virtual void activemq::commands::Message::setPersistent (bool persistent) [virtual]`
- 6.340.2.85 `virtual void activemq::commands::Message::setPriority (unsigned char priority) [virtual]`
- 6.340.2.86 `virtual void activemq::commands::Message::setProducerId (const Pointer< ProducerId > & producerId) [virtual]`
- 6.340.2.87 `void activemq::commands::Message::setReadOnlyBody (bool value) [inline]`

Set the Read Only State of the **Message** (p. 1821) Content.

Parameters

<i>value</i>	- true if Content should be read only.
--------------	--

- 6.340.2.88 `void activemq::commands::Message::setReadOnlyProperties (bool value) [inline]`

Set the Read Only State of the **Message** (p. 1821) Properties.

Parameters

<i>value</i>	- true if Properties should be read only.
--------------	---

- 6.340.2.89 `virtual void activemq::commands::Message::setRecievedByDFBridge (bool recievedByDFBridge) [virtual]`
- 6.340.2.90 `virtual void activemq::commands::Message::setRedeliveryCounter (int redeliveryCounter) [virtual]`
- 6.340.2.91 `virtual void activemq::commands::Message::setReplyTo (const Pointer< ActiveMQDestination > & replyTo) [virtual]`
- 6.340.2.92 `virtual void activemq::commands::Message::setTargetConsumerId (const Pointer< ConsumerId > & targetConsumerId) [virtual]`

- 6.340.2.93 `virtual void activemq::commands::Message::setTimestamp (long long timestamp) [virtual]`
- 6.340.2.94 `virtual void activemq::commands::Message::setTransactionId (const Pointer< TransactionId > & transactionId) [virtual]`
- 6.340.2.95 `virtual void activemq::commands::Message::setType (const std::string & type) [virtual]`
- 6.340.2.96 `virtual void activemq::commands::Message::setUserID (const std::string & userID) [virtual]`
- 6.340.2.97 `virtual std::string activemq::commands::Message::toString () const [virtual]`

Returns a string containing the information for this **DataStructure** (p. 1133) such as its type and value of its elements.

Returns

formatted string useful for debugging.

Reimplemented from **activemq::commands::BaseCommand** (p. 498).

Reimplemented in **activemq::commands::ActiveMQBytesMessage** (p. 178), **activemq::commands::ActiveMQStreamMessage** (p. 369), **activemq::commands::ActiveMQMapMessage** (p. 283), **activemq::commands::ActiveMQBlobMessage** (p. 161), **activemq::commands::ActiveMQTextMessage** (p. 409), **activemq::commands::ActiveMQObjectMessage** (p. 303), and **activemq::commands::ActiveMQMessage** (p. 290).

- 6.340.2.98 `virtual Pointer<Command> activemq::commands::Message::visit (activemq::state::CommandVisitor * visitor) [virtual]`

Allows a Visitor to visit this command and return a response to the command based on the command type being visited.

The command will call the proper processXXX method in the visitor.

Returns

a **Response** (p. 2298) to the visitor being called or NULL if no response.

Implements **activemq::commands::Command** (p. 871).

6.340.3 Field Documentation

- 6.340.3.1 `long long activemq::commands::Message::arrival [protected]`

- 6.340.3.2 `long long activemq::commands::Message::brokerInTime`
[protected]
- 6.340.3.3 `long long activemq::commands::Message::brokerOutTime`
[protected]
- 6.340.3.4 `std::vector< decaf::lang::Pointer<BrokerId> >`
`activemq::commands::Message::brokerPath` [protected]
- 6.340.3.5 `std::vector< decaf::lang::Pointer<BrokerId> >`
`activemq::commands::Message::cluster` [protected]
- 6.340.3.6 `bool activemq::commands::Message::compressed` [protected]
- 6.340.3.7 `core::ActiveMQConnection* activemq::commands::Message-`
`::connection` [protected]
- 6.340.3.8 `std::vector<unsigned char> activemq::commands::Message::content`
[protected]
- 6.340.3.9 `std::string activemq::commands::Message::correlationId`
[protected]
- 6.340.3.10 `Pointer<DataStructure> activemq::commands::Message::data-`
`Structure` [protected]
- 6.340.3.11 `const unsigned int activemq::commands::Message-`
`::DEFAULT_MESSAGE_SIZE = 1024` [static,
protected]
- 6.340.3.12 `Pointer<ActiveMQDestination> activemq::commands::Message-`
`::destination` [protected]
- 6.340.3.13 `bool activemq::commands::Message::droppable` [protected]
- 6.340.3.14 `long long activemq::commands::Message::expiration` [protected]
- 6.340.3.15 `std::string activemq::commands::Message::groupId` [protected]
- 6.340.3.16 `int activemq::commands::Message::groupSequence` [protected]
- 6.340.3.17 `const unsigned char activemq::commands::Message::ID_MESSAGE = 0`
[static]
- 6.340.3.18 `std::vector<unsigned char> activemq::commands::Message::marshalled-`
`Properties` [protected]

- 6.340.3.19 **Pointer<MessageId> activemq::commands::Message::messageId** [protected]
- 6.340.3.20 **Pointer<ActiveMQDestination> activemq::commands::Message::originalDestination** [protected]
- 6.340.3.21 **Pointer<TransactionId> activemq::commands::Message::originalTransactionId** [protected]
- 6.340.3.22 **bool activemq::commands::Message::persistent** [protected]
- 6.340.3.23 **unsigned char activemq::commands::Message::priority** [protected]
- 6.340.3.24 **Pointer<ProducerId> activemq::commands::Message::producerId** [protected]
- 6.340.3.25 **bool activemq::commands::Message::recievedByDFBridge** [protected]
- 6.340.3.26 **int activemq::commands::Message::redeliveryCounter** [protected]
- 6.340.3.27 **Pointer<ActiveMQDestination> activemq::commands::Message::replyTo** [protected]
- 6.340.3.28 **Pointer<ConsumerId> activemq::commands::Message::targetConsumerId** [protected]
- 6.340.3.29 **long long activemq::commands::Message::timestamp** [protected]
- 6.340.3.30 **Pointer<TransactionId> activemq::commands::Message::transactionId** [protected]
- 6.340.3.31 **std::string activemq::commands::Message::type** [protected]
- 6.340.3.32 **std::string activemq::commands::Message::userID** [protected]

The documentation for this class was generated from the following file:

- src/main/activemq/commands/**Message.h**

6.341 cms::Message Class Reference

Root of all messages.

```
#include <src/main/cms/Message.h>
```

Inheritance diagram for cms::Message:

Public Member Functions

- virtual **~Message** ()
- virtual **Message * clone** () const =0
Clone this message exactly, returns a new instance that the caller is required to delete.
- virtual void **acknowledge** () const =0
Acknowledges all consumed messages of the session of this consumed message.
- virtual void **clearBody** ()=0
Clears out the body of the message.
- virtual void **clearProperties** ()=0
Clears out the message body.
- virtual std::vector< std::string > **getPropertyNames** () const =0
Retrieves the property names.
- virtual bool **propertyExists** (const std::string &name) const =0
Indicates whether or not a given property exists.
- virtual bool **getBooleanProperty** (const std::string &name) const =0
Gets a boolean property.
- virtual unsigned char **getByteProperty** (const std::string &name) const =0
Gets a byte property.
- virtual double **getDoubleProperty** (const std::string &name) const =0
Gets a double property.
- virtual float **getFloatProperty** (const std::string &name) const =0
Gets a float property.
- virtual int **getIntProperty** (const std::string &name) const =0
Gets a int property.
- virtual long long **getLongProperty** (const std::string &name) const =0
Gets a long property.
- virtual short **getShortProperty** (const std::string &name) const =0
Gets a short property.
- virtual std::string **getStringProperty** (const std::string &name) const =0
Gets a string property.
- virtual void **setBooleanProperty** (const std::string &name, bool value)=0
Sets a boolean property.
- virtual void **setByteProperty** (const std::string &name, unsigned char value)=0
Sets a byte property.
- virtual void **setDoubleProperty** (const std::string &name, double value)=0
Sets a double property.
- virtual void **setFloatProperty** (const std::string &name, float value)=0
Sets a float property.

- virtual void **setIntProperty** (const std::string &name, int value)=0
Sets a int property.
- virtual void **setLongProperty** (const std::string &name, long long value)=0
Sets a long property.
- virtual void **setShortProperty** (const std::string &name, short value)=0
Sets a short property.
- virtual void **setStringProperty** (const std::string &name, const std::string &value)=0
Sets a string property.
- virtual std::string **getCMSCorrelationID** () const =0
Gets the correlation ID for the message.
- virtual void **setCMSCorrelationID** (const std::string &correlationId)=0
Sets the correlation ID for the message.
- virtual int **getCMSDeliveryMode** () const =0
*Gets the **DeliveryMode** (p. 1195) for this message.*
- virtual void **setCMSDeliveryMode** (int mode)=0
*Sets the **DeliveryMode** (p. 1195) for this message.*
- virtual const **Destination** * **getCMSDestination** () const =0
*Gets the **Destination** (p. 1210) object for this message.*
- virtual void **setCMSDestination** (const **Destination** *destination)=0
*Sets the **Destination** (p. 1210) object for this message.*
- virtual long long **getCMSExpiration** () const =0
Gets the message's expiration value.
- virtual void **setCMSExpiration** (long long expireTime)=0
Sets the message's expiration value.
- virtual std::string **getCMSMessageID** () const =0
The CMSMessageID header field contains a value that uniquely identifies each message sent by a provider.
- virtual void **setCMSMessageID** (const std::string &id)=0
Sets the message ID.
- virtual int **getCMSPriority** () const =0
Gets the message priority level.
- virtual void **setCMSPriority** (int priority)=0
Sets the Priority Value for this message.
- virtual bool **getCMSRedelivered** () const =0
Gets an indication of whether this message is being redelivered.
- virtual void **setCMSRedelivered** (bool redelivered)=0
Specifies whether this message is being redelivered.
- virtual const **cms::Destination** * **getCMSReplyTo** () const =0
*Gets the **Destination** (p. 1210) object to which a reply to this message should be sent.*
- virtual void **setCMSReplyTo** (const **cms::Destination** *destination)=0
*Sets the **Destination** (p. 1210) object to which a reply to this message should be sent.*
- virtual long long **getCMSTimestamp** () const =0

Gets the message timestamp.

- virtual void **setCMSTimestamp** (long long timeStamp)=0

Sets the message timestamp.

- virtual std::string **getCMSType** () const =0

Gets the message type identifier supplied by the client when the message was sent.

- virtual void **setCMSType** (const std::string &type)=0

Sets the message type.

Static Public Attributes

- static const int **DEFAULT_DELIVERY_MODE**

*The Default delivery mode for **Message** (p. 1839) Producers is PERSISTENT.*

- static const int **DEFAULT_MSG_PRIORITY**

*The Default priority assigned to a **Message** (p. 1839) is 4.*

- static const long long **DEFAULT_TIME_TO_LIVE**

*The Default Time to Live for a **Message** (p. 1839) Producer is unlimited, the message will never expire.*

6.341.1 Detailed Description

Root of all messages.

As in JMS, a message is comprised of 3 parts: CMS-specific headers, user-defined properties, and the body.

Message (p. 1839) Bodies

The CMS API defines four types of message bodies, each type is contained within its own **Message** (p. 1839) Interface definition.

- Stream - A **StreamMessage** (p. 2606) object's message body contains a stream of primitive values in the C++ language. It is filled and read sequentially. - Unlike the **BytesMessage** (p. 718) type the values written to a **StreamMessage** (p. 2606) retain information on their type and rules for type conversion are enforced when reading back the values from the **Message** (p. 1839) Body.
- Map - A **MapMessage** (p. 1779) object's message body contains a set of name-value pairs, where names are std::string objects, and values are C++ primitives. The entries can be accessed sequentially or randomly by name. The **MapMessage** (p. 1779) makes no guarantee on the order of the elements within the **Message** (p. 1839) body.
- Text - A **TextMessage** (p. 2701) object's message body contains a std::string object. This message type can be used to transport plain-text messages, and XML messages.
- Bytes - A **BytesMessage** (p. 718) object's message body contains a stream of uninterpreted bytes. This message type is for literally encoding a body to match an existing message format. In many cases, it is possible to use one of the other body types, which are easier to use.

Message (p. 1839) Properties

Message (p. 1839) properties support the following conversion table. The marked cases must be supported. The unmarked cases must throw a **CMSEException** (p. 826). The String-to-primitive conversions may throw a runtime exception if the primitive's `valueOf` method does not accept the String as a valid representation of the primitive.

A value written as the row type can be read as the column type.

	boolean	byte	short	int	long	float	double	String
boolean	X							X
byte		X	X	X	X			X
short			X	X	X			X
int				X	X			X
long					X			X
float						X	X	X
double							X	X
String	X	X	X	X	X	X	X	X

When a **Message** (p. 1839) is delivered its properties are considered to be in a read-only mode and cannot be changed. Attempting to change the value of a delivered **Message** (p. 1839)'s properties will result in a **CMSEException** (p. 826) being thrown.

See also

JMS API

Since

1.0

6.341.2 Constructor & Destructor Documentation

6.341.2.1 `virtual cms::Message::~~Message () [virtual]`

6.341.3 Member Function Documentation

6.341.3.1 `virtual void cms::Message::acknowledge () const [pure virtual]`

Acknowledges all consumed messages of the session of this consumed message.

All consumed CMS messages support the `acknowledge` method for use when a client has specified that its CMS session's consumed messages are to be explicitly acknowledged. By invoking `acknowledge` on a consumed message, a client acknowledges all messages consumed by the session that the message was delivered to.

Calls to `acknowledge` are ignored for both transacted sessions and sessions specified to use implicit acknowledgment modes.

A client may individually acknowledge each message as it is consumed, or it may choose to acknowledge messages as an application-defined group (which is done by calling `acknowledge` on the last received message of the group, thereby acknowledging all messages consumed by the session.)

Messages that have been received but not acknowledged may be redelivered.

Exceptions

<i>CMSException</i> (p. 826)	- if an internal error occurs.
<i>IllegalStateException</i> (p. 1419)	- if this method is called on a closed session.

Implemented in `activemq::commands::ActiveMQMessageTemplate< cms::BytesMessage >` (p. 296), `activemq::commands::ActiveMQMessageTemplate< cms::MapMessage >` (p. 296), `activemq::commands::ActiveMQMessageTemplate< cms::Message >` (p. 296), `activemq::commands::ActiveMQMessageTemplate< cms::StreamMessage >` (p. 296), `activemq::commands::ActiveMQMessageTemplate< cms::TextMessage >` (p. 296), and `activemq::commands::ActiveMQMessageTemplate< cms::ObjectMessage >` (p. 296).

6.341.3.2 virtual void cms::Message::clearBody() [pure virtual]

Clears out the body of the message.

This does not clear the headers or properties.

Exceptions

<i>CMSException</i> (p. 826)	- if an internal error occurs.
--	--------------------------------

Implemented in `activemq::commands::ActiveMQBytesMessage` (p. 168), `activemq::commands::ActiveMQMessageTemplate< cms::BytesMessage >` (p. 296), `activemq::commands::ActiveMQMessageTemplate< cms::MapMessage >` (p. 296), `activemq::commands::ActiveMQMessageTemplate< cms::Message >` (p. 296), `activemq::commands::ActiveMQMessageTemplate< cms::StreamMessage >` (p. 296), `activemq::commands::ActiveMQMessageTemplate< cms::TextMessage >` (p. 296), `activemq::commands::ActiveMQMessageTemplate< cms::ObjectMessage >` (p. 296), `activemq::commands::ActiveMQStreamMessage` (p. 361), `activemq::commands::ActiveMQMapMessage` (p. 271), and `activemq::commands::ActiveMQTextMessage` (p. 406).

6.341.3.3 virtual void cms::Message::clearProperties() [pure virtual]

Clears out the message body.

Clearing a message's body does not clear its header values or property entries.

If this message body was read-only, calling this method leaves the message body in the same state as an empty body in a newly created message.

Exceptions

<i>CMSException</i> (p. 826)	- if an internal error occurs.
--	--------------------------------

Implemented in **activemq::commands::ActiveMQMessageTemplate**< **cms::BytesMessage** > (p. 297), **activemq::commands::ActiveMQMessageTemplate**< **cms::MapMessage** > (p. 297), **activemq::commands::ActiveMQMessageTemplate**< **cms::Message** > (p. 297), **activemq::commands::ActiveMQMessageTemplate**< **cms::StreamMessage** > (p. 297), **activemq::commands::ActiveMQMessageTemplate**< **cms::TextMessage** > (p. 297), and **activemq::commands::ActiveMQMessageTemplate**< **cms::ObjectMessage** > (p. 297).

6.341.3.4 **virtual Message* cms::Message::clone ()const** [pure virtual]

Clone this message exactly, returns a new instance that the caller is required to delete.

Returns

new copy of this message

Implemented in **cms::BytesMessage** (p. 720), **activemq::commands::ActiveMQBytesMessage** (p. 168), **activemq::commands::ActiveMQStreamMessage** (p. 361), **activemq::commands::ActiveMQMapMessage** (p. 272), **activemq::commands::ActiveMQBlobMessage** (p. 158), **activemq::commands::ActiveMQTextMessage** (p. 407), **activemq::commands::ActiveMQObjectMessage** (p. 302), and **activemq::commands::ActiveMQMessage** (p. 289).

6.341.3.5 **virtual bool cms::Message::getBooleanProperty (const std::string & name)**
const [pure virtual]

Gets a boolean property.

Parameters

<i>name</i>	The name of the property to retrieve.
-------------	---------------------------------------

Returns

The value for the named property.

Exceptions

<i>CMSException</i> (p. 826)	if the property does not exist.
--	---------------------------------

<i>MessageFormatException</i> (p. 1906)	- if this type conversion is invalid.
---	---------------------------------------

Implemented in `activemq::commands::ActiveMQMessageTemplate< cms::BytesMessage >` (p. 297), `activemq::commands::ActiveMQMessageTemplate< cms::MapMessage >` (p. 297), `activemq::commands::ActiveMQMessageTemplate< cms::Message >` (p. 297), `activemq::commands::ActiveMQMessageTemplate< cms::StreamMessage >` (p. 297), `activemq::commands::ActiveMQMessageTemplate< cms::TextMessage >` (p. 297), and `activemq::commands::ActiveMQMessageTemplate< cms::ObjectMessage >` (p. 297).

6.341.3.6 virtual unsigned char `cms::Message::getBytesProperty (const std::string & name) const` [pure virtual]

Gets a byte property.

Parameters

<i>name</i>	The name of the property to retrieve.
-------------	---------------------------------------

Returns

The value for the named property.

Exceptions

<i>CMSEException</i> (p. 826)	if the property does not exist.
<i>MessageFormatException</i> (p. 1906)	- if this type conversion is invalid.

Implemented in `activemq::commands::ActiveMQMessageTemplate< cms::BytesMessage >` (p. 297), `activemq::commands::ActiveMQMessageTemplate< cms::MapMessage >` (p. 297), `activemq::commands::ActiveMQMessageTemplate< cms::Message >` (p. 297), `activemq::commands::ActiveMQMessageTemplate< cms::StreamMessage >` (p. 297), `activemq::commands::ActiveMQMessageTemplate< cms::TextMessage >` (p. 297), and `activemq::commands::ActiveMQMessageTemplate< cms::ObjectMessage >` (p. 297).

6.341.3.7 virtual std::string `cms::Message::getCMSCorrelationID () const` [pure virtual]

Gets the correlation ID for the message.

This method is used to return correlation ID values that are either provider-specific mes-

sage IDs or application-specific String values.

Returns

string representation of the correlation Id

Exceptions

<i>CMSException</i> (p. 826)	- if an internal error occurs.
--	--------------------------------

Implemented in **activemq::commands::ActiveMQMessageTemplate< cms::BytesMessage >** (p. 297), **activemq::commands::ActiveMQMessageTemplate< cms::MapMessage >** (p. 297), **activemq::commands::ActiveMQMessageTemplate< cms::Message >** (p. 297), **activemq::commands::ActiveMQMessageTemplate< cms::StreamMessage >** (p. 297), **activemq::commands::ActiveMQMessageTemplate< cms::TextMessage >** (p. 297), and **activemq::commands::ActiveMQMessageTemplate< cms::ObjectMessage >** (p. 297).

```
6.341.3.8 virtual int cms::Message::getCMSDeliveryMode ( ) const [pure
virtual]
```

Gets the **DeliveryMode** (p. 1195) for this message.

Returns

DeliveryMode (p. 1195) enumerated value.

Exceptions

<i>CMSException</i> (p. 826)	- if an internal error occurs.
--	--------------------------------

Implemented in **activemq::commands::ActiveMQMessageTemplate< cms::BytesMessage >** (p. 298), **activemq::commands::ActiveMQMessageTemplate< cms::MapMessage >** (p. 298), **activemq::commands::ActiveMQMessageTemplate< cms::Message >** (p. 298), **activemq::commands::ActiveMQMessageTemplate< cms::StreamMessage >** (p. 298), **activemq::commands::ActiveMQMessageTemplate< cms::TextMessage >** (p. 298), and **activemq::commands::ActiveMQMessageTemplate< cms::ObjectMessage >** (p. 298).

```
6.341.3.9 virtual const Destination* cms::Message::getCMSDestination ( ) const
[pure virtual]
```

Gets the **Destination** (p. 1210) object for this message.

The CMSDestination header field contains the destination to which the message is being sent.

When a message is sent, this field is ignored. After completion of the send or publish method, the field holds the destination specified by the method.

When a message is received, its CMSDestination value must be equivalent to the value assigned when it was sent.

Returns

Destination (p. 1210) object

Exceptions

CMSException (p. 826)	- if an internal error occurs.
---------------------------------	--------------------------------

Implemented in **activemq::commands::ActiveMQMessageTemplate**< **cms::BytesMessage** > (p. 298), **activemq::commands::ActiveMQMessageTemplate**< **cms::MapMessage** > (p. 298), **activemq::commands::ActiveMQMessageTemplate**< **cms::Message** > (p. 298), **activemq::commands::ActiveMQMessageTemplate**< **cms::StreamMessage** > (p. 298), **activemq::commands::ActiveMQMessageTemplate**< **cms::TextMessage** > (p. 298), and **activemq::commands::ActiveMQMessageTemplate**< **cms::ObjectMessage** > (p. 298).

6.341.3.10 `virtual long long cms::Message::getCMSExpiration () const` [pure virtual]

Gets the message's expiration value.

When a message is sent, the CMSExpiration header field is left unassigned. After completion of the send or publish method, it holds the expiration time of the message. This is the sum of the time-to-live value specified by the client and the GMT at the time of the send or publish.

If the time-to-live is specified as zero, CMSExpiration is set to zero to indicate that the message does not expire.

When a message's expiration time is reached, a provider should discard it. The CMS API does not define any form of notification of message expiration.

Clients should not receive messages that have expired; however, the CMS API does not guarantee that this will not happen.

Returns

the time the message expires, which is the sum of the time-to-live value specified by the client and the GMT at the time of the send

Exceptions

CMSException (p. 826)	- if an internal error occurs.
---------------------------------	--------------------------------

Implemented in **activemq::commands::ActiveMQMessageTemplate**< **cms::BytesMessage** > (p. 298), **activemq::commands::ActiveMQMessageTemplate**< **cms::MapMessage** > (p. 298), **activemq::commands::ActiveMQMessageTemplate**< **cms::Message** > (p. 298), **activemq::commands::ActiveMQMessageTemplate**< **cms::StreamMessage** > (p. 298), **activemq::commands::ActiveMQMessageTemplate**< **cms::TextMessage** > (p. 298), and **activemq::commands::ActiveMQMessageTemplate**< **cms::ObjectMessage** > (p. 298).

6.341.3.11 `virtual std::string cms::Message::getCMSMessageID () const` [pure virtual]

The CMSMessageID header field contains a value that uniquely identifies each message sent by a provider.

When a message is sent, CMSMessageID can be ignored. When the send or publish method returns, it contains a provider-assigned value.

A CMSMessageID is a String value that should function as a unique key for identifying messages in a historical repository. The exact scope of uniqueness is provider-defined. It should at least cover all messages for a specific installation of a provider, where an installation is some connected set of message routers.

All CMSMessageID values must start with the prefix 'ID:'. Uniqueness of message ID values across different providers is not required.

Since message IDs take some effort to create and increase a message's size, some CMS providers may be able to optimize message overhead if they are given a hint that the message ID is not used by an application. By calling the **MessageProducer.setDisableMessageID** (p. 1931) method, a CMS client enables this potential optimization for all messages sent by that message producer. If the CMS provider accepts this hint, these messages must have the message ID set to null; if the provider ignores the hint, the message ID must be set to its normal unique value.

Returns

provider-assigned message id

Exceptions

CMSException (p. 826)	- if an internal error occurs.
---------------------------------	--------------------------------

Implemented in **activemq::commands::ActiveMQMessageTemplate**< **cms::BytesMessage** > (p. 298), **activemq::commands::ActiveMQMessageTemplate**< **cms::MapMessage** > (p. 298), **activemq::commands::ActiveMQMessageTemplate**< **cms::Message** > (p. 298), **activemq::commands::ActiveMQMessageTemplate**< **cms::StreamMessage** > (p. 298), **activemq::commands::ActiveMQMessageTemplate**< **cms::TextMessage** > (p. 298), and **activemq::commands::ActiveMQMessageTemplate**< **cms::ObjectMessage** > (p. 298).

6.341.3.12 `virtual int cms::Message::getCMSPriority () const [pure virtual]`

Gets the message priority level.

The CMS API defines ten levels of priority value, with 0 as the lowest priority and 9 as the highest. In addition, clients should consider priorities 0-4 as gradations of normal priority and priorities 5-9 as gradations of expedited priority.

The CMS API does not require that a provider strictly implement priority ordering of messages; however, it should do its best to deliver expedited messages ahead of normal messages.

Returns

priority value

Exceptions

<i>CMSException</i> (p. 826)	- if an internal error occurs.
--	--------------------------------

Implemented in `activemq::commands::ActiveMQMessageTemplate< cms::BytesMessage >` (p. 298), `activemq::commands::ActiveMQMessageTemplate< cms::MapMessage >` (p. 298), `activemq::commands::ActiveMQMessageTemplate< cms::Message >` (p. 298), `activemq::commands::ActiveMQMessageTemplate< cms::StreamMessage >` (p. 298), `activemq::commands::ActiveMQMessageTemplate< cms::TextMessage >` (p. 298), and `activemq::commands::ActiveMQMessageTemplate< cms::ObjectMessage >` (p. 298).

6.341.3.13 `virtual bool cms::Message::getCMSRedelivered () const [pure virtual]`

Gets an indication of whether this message is being redelivered.

If a client receives a message with the CMSRedelivered field set, it is likely, but not guaranteed, that this message was delivered earlier but that its receipt was not acknowledged at that time.

Returns

true if this message is being redelivered

Exceptions

<i>CMSException</i> (p. 826)	- if an internal error occurs.
--	--------------------------------

Implemented in `activemq::commands::ActiveMQMessageTemplate< cms::BytesMessage >` (p. 298), `activemq::commands::ActiveMQMessageTemplate< cms::MapMessage >` (p. 298), `activemq::commands::ActiveMQMessageTemplate<`

cms::Message > (p. 298), **activemq::commands::ActiveMQMessageTemplate**< **cms::StreamMessage** > (p. 298), **activemq::commands::ActiveMQMessageTemplate**< **cms::TextMessage** > (p. 298), and **activemq::commands::ActiveMQMessageTemplate**< **cms::ObjectMessage** > (p. 298).

6.341.3.14 `virtual const cms::Destination* cms::Message::getCMSReplyTo () const` [pure virtual]

Gets the **Destination** (p. 1210) object to which a reply to this message should be sent.

Returns

Destination (p. 1210) to which to send a response to this message

Exceptions

CMSException (p. 826)	- if an internal error occurs.
---------------------------------	--------------------------------

Implemented in **activemq::commands::ActiveMQMessageTemplate**< **cms::BytesMessage** > (p. 298), **activemq::commands::ActiveMQMessageTemplate**< **cms::MapMessage** > (p. 298), **activemq::commands::ActiveMQMessageTemplate**< **cms::Message** > (p. 298), **activemq::commands::ActiveMQMessageTemplate**< **cms::StreamMessage** > (p. 298), **activemq::commands::ActiveMQMessageTemplate**< **cms::TextMessage** > (p. 298), and **activemq::commands::ActiveMQMessageTemplate**< **cms::ObjectMessage** > (p. 298).

6.341.3.15 `virtual long long cms::Message::getCMSTimestamp () const` [pure virtual]

Gets the message timestamp.

The **CMSTimestamp** header field contains the time a message was handed off to a provider to be sent. It is not the time the message was actually transmitted, because the actual send may occur later due to transactions or other client-side queuing of messages.

When a message is sent, **CMSTimestamp** is ignored. When the send or publish method returns, it contains a time value somewhere in the interval between the call and the return. The value is in the format of a normal millis time value in the Java programming language.

Since timestamps take some effort to create and increase a message's size, some CMS providers may be able to optimize message overhead if they are given a hint that the timestamp is not used by an application. By calling the `MessageProducer.setDisableMessageTimestamp` method, a CMS client enables this potential optimization for all messages sent by that message producer. If the CMS provider accepts this hint, these messages must have the timestamp set to zero; if the provider ignores the hint, the timestamp must be set to its normal value.

Returns

the message timestamp

Exceptions

<i>CMSEException</i> (p. 826)	- if an internal error occurs.
---	--------------------------------

Implemented in `activemq::commands::ActiveMQMessageTemplate< cms::BytesMessage >` (p. 298), `activemq::commands::ActiveMQMessageTemplate< cms::MapMessage >` (p. 298), `activemq::commands::ActiveMQMessageTemplate< cms::Message >` (p. 298), `activemq::commands::ActiveMQMessageTemplate< cms::StreamMessage >` (p. 298), `activemq::commands::ActiveMQMessageTemplate< cms::TextMessage >` (p. 298), and `activemq::commands::ActiveMQMessageTemplate< cms::ObjectMessage >` (p. 298).

6.341.3.16 `virtual std::string cms::Message::getCMSType () const` [pure virtual]

Gets the message type identifier supplied by the client when the message was sent.

Returns

the message type

See also

`setCMSType` (p. 1863)

Exceptions

<i>CMSEException</i> (p. 826)	- if an internal error occurs.
---	--------------------------------

Implemented in `activemq::commands::ActiveMQMessageTemplate< cms::BytesMessage >` (p. 298), `activemq::commands::ActiveMQMessageTemplate< cms::MapMessage >` (p. 298), `activemq::commands::ActiveMQMessageTemplate< cms::Message >` (p. 298), `activemq::commands::ActiveMQMessageTemplate< cms::StreamMessage >` (p. 298), `activemq::commands::ActiveMQMessageTemplate< cms::TextMessage >` (p. 298), and `activemq::commands::ActiveMQMessageTemplate< cms::ObjectMessage >` (p. 298).

6.341.3.17 `virtual double cms::Message::getDoubleProperty (const std::string & name) const` [pure virtual]

Gets a double property.

Parameters

<i>name</i>	The name of the property to retrieve.
-------------	---------------------------------------

Returns

The value for the named property.

Exceptions

<i>CMSException</i> (p. 826)	if the property does not exist.
<i>MessageFormatException</i> (p. 1906)	- if this type conversion is invalid.

Implemented in `activemq::commands::ActiveMQMessageTemplate< cms::BytesMessage >` (p. 298), `activemq::commands::ActiveMQMessageTemplate< cms::MapMessage >` (p. 298), `activemq::commands::ActiveMQMessageTemplate< cms::Message >` (p. 298), `activemq::commands::ActiveMQMessageTemplate< cms::StreamMessage >` (p. 298), `activemq::commands::ActiveMQMessageTemplate< cms::TextMessage >` (p. 298), and `activemq::commands::ActiveMQMessageTemplate< cms::ObjectMessage >` (p. 298).

6.341.3.18 `virtual float cms::Message::getFloatProperty (const std::string & name)`
`const [pure virtual]`

Gets a float property.

Parameters

<i>name</i>	The name of the property to retrieve.
-------------	---------------------------------------

Returns

The value for the named property.

Exceptions

<i>CMSException</i> (p. 826)	if the property does not exist.
<i>MessageFormatException</i> (p. 1906)	- if this type conversion is invalid.

Implemented in `activemq::commands::ActiveMQMessageTemplate< cms::BytesMessage >` (p. 298), `activemq::commands::ActiveMQMessageTemplate< cms::MapMessage >` (p. 298), `activemq::commands::ActiveMQMessageTemplate<`

cms::Message > (p. 298), **activemq::commands::ActiveMQMessageTemplate< cms::StreamMessage >** (p. 298), **activemq::commands::ActiveMQMessageTemplate< cms::TextMessage >** (p. 298), and **activemq::commands::ActiveMQMessageTemplate< cms::ObjectMessage >** (p. 298).

6.341.3.19 `virtual int cms::Message::getIntProperty (const std::string & name) const`
`[pure virtual]`

Gets a int property.

Parameters

<i>name</i>	The name of the property to retrieve.
-------------	---------------------------------------

Returns

The value for the named property.

Exceptions

<i>CMSException</i> (p. 826)	if the property does not exist.
<i>MessageFormatException</i> (p. 1906)	- if this type conversion is invalid.

Implemented in **activemq::commands::ActiveMQMessageTemplate< cms::BytesMessage >** (p. 298), **activemq::commands::ActiveMQMessageTemplate< cms::MapMessage >** (p. 298), **activemq::commands::ActiveMQMessageTemplate< cms::Message >** (p. 298), **activemq::commands::ActiveMQMessageTemplate< cms::StreamMessage >** (p. 298), **activemq::commands::ActiveMQMessageTemplate< cms::TextMessage >** (p. 298), and **activemq::commands::ActiveMQMessageTemplate< cms::ObjectMessage >** (p. 298).

6.341.3.20 `virtual long long cms::Message::getLongProperty (const std::string & name) const`
`[pure virtual]`

Gets a long property.

Parameters

<i>name</i>	The name of the property to retrieve.
-------------	---------------------------------------

Returns

The value for the named property.

Exceptions

<i>CMSException</i> (p. 826)	if the property does not exist.
<i>MessageFormatException</i> (p. 1906)	- if this type conversion is invalid.

Implemented in `activemq::commands::ActiveMQMessageTemplate< cms::BytesMessage >` (p. 299), `activemq::commands::ActiveMQMessageTemplate< cms::MapMessage >` (p. 299), `activemq::commands::ActiveMQMessageTemplate< cms::Message >` (p. 299), `activemq::commands::ActiveMQMessageTemplate< cms::StreamMessage >` (p. 299), `activemq::commands::ActiveMQMessageTemplate< cms::TextMessage >` (p. 299), and `activemq::commands::ActiveMQMessageTemplate< cms::ObjectMessage >` (p. 299).

6.341.3.21 `virtual std::vector<std::string> cms::Message::getPropertyNames () const`
[pure virtual]

Retrieves the property names.

Returns

The complete set of property names currently in this message.

Exceptions

<i>CMSException</i> (p. 826)	- if an internal error occurs.
--	--------------------------------

Implemented in `activemq::commands::ActiveMQMessageTemplate< cms::BytesMessage >` (p. 299), `activemq::commands::ActiveMQMessageTemplate< cms::MapMessage >` (p. 299), `activemq::commands::ActiveMQMessageTemplate< cms::Message >` (p. 299), `activemq::commands::ActiveMQMessageTemplate< cms::StreamMessage >` (p. 299), `activemq::commands::ActiveMQMessageTemplate< cms::TextMessage >` (p. 299), and `activemq::commands::ActiveMQMessageTemplate< cms::ObjectMessage >` (p. 299).

6.341.3.22 `virtual short cms::Message::getShortProperty (const std::string & name)`
`const` [pure virtual]

Gets a short property.

Parameters

<i>name</i>	The name of the property to retrieve.
-------------	---------------------------------------

Returns

The value for the named property.

Exceptions

<i>CMSException</i> (p. 826)	if the property does not exist.
<i>MessageFormatException</i> (p. 1906)	- if this type conversion is invalid.

Implemented in `activemq::commands::ActiveMQMessageTemplate< cms::BytesMessage >` (p. 299), `activemq::commands::ActiveMQMessageTemplate< cms::MapMessage >` (p. 299), `activemq::commands::ActiveMQMessageTemplate< cms::Message >` (p. 299), `activemq::commands::ActiveMQMessageTemplate< cms::StreamMessage >` (p. 299), `activemq::commands::ActiveMQMessageTemplate< cms::TextMessage >` (p. 299), and `activemq::commands::ActiveMQMessageTemplate< cms::ObjectMessage >` (p. 299).

6.341.3.23 `virtual std::string cms::Message::getStringProperty (const std::string & name) const` [pure virtual]

Gets a string property.

Parameters

<i>name</i>	The name of the property to retrieve.
-------------	---------------------------------------

Returns

The value for the named property.

Exceptions

<i>CMSException</i> (p. 826)	if the property does not exist.
<i>MessageFormatException</i> (p. 1906)	- if this type conversion is invalid.

Implemented in `activemq::commands::ActiveMQMessageTemplate< cms::BytesMessage >` (p. 299), `activemq::commands::ActiveMQMessageTemplate< cms::MapMessage >` (p. 299), `activemq::commands::ActiveMQMessageTemplate< cms::Message >` (p. 299), `activemq::commands::ActiveMQMessageTemplate< cms::StreamMessage >` (p. 299), `activemq::commands::ActiveMQMessageTemplate< cms::TextMessage >` (p. 299), and `activemq::commands::ActiveMQMessageTemplate< cms::ObjectMessage >` (p. 299).

6.341.3.24 `virtual bool cms::Message::propertyExists (const std::string & name) const`
`[pure virtual]`

Indicates whether or not a given property exists.

Parameters

<i>name</i>	The name of the property to look up.
-------------	--------------------------------------

Returns

True if the property exists in this message.

Exceptions

<i>CMSException</i> (p. 826)	- if an internal error occurs.
--	--------------------------------

Implemented in `activemq::commands::ActiveMQMessageTemplate< cms::BytesMessage >` (p. 299), `activemq::commands::ActiveMQMessageTemplate< cms::MapMessage >` (p. 299), `activemq::commands::ActiveMQMessageTemplate< cms::Message >` (p. 299), `activemq::commands::ActiveMQMessageTemplate< cms::StreamMessage >` (p. 299), `activemq::commands::ActiveMQMessageTemplate< cms::TextMessage >` (p. 299), and `activemq::commands::ActiveMQMessageTemplate< cms::ObjectMessage >` (p. 299).

6.341.3.25 `virtual void cms::Message::setBooleanProperty (const std::string & name, bool value)` `[pure virtual]`

Sets a boolean property.

Parameters

<i>name</i>	The name of the property to retrieve.
<i>value</i>	The value for the named property.

Exceptions

<i>CMSException</i> (p. 826)	- if the name is an empty string.
<i>MessageNotWritableException</i> (p. 1923)	- if properties are read-only

Implemented in `activemq::commands::ActiveMQMessageTemplate< cms::BytesMessage >` (p. 299), `activemq::commands::ActiveMQMessageTemplate< cms::MapMessage >` (p. 299), `activemq::commands::ActiveMQMessageTemplate< cms::Message >` (p. 299), `activemq::commands::ActiveMQMessageTemplate<`

cms::StreamMessage > (p. 299), **activemq::commands::ActiveMQMessageTemplate< cms::TextMessage >** (p. 299), and **activemq::commands::ActiveMQMessageTemplate< cms::ObjectMessage >** (p. 299).

6.341.3.26 virtual void **cms::Message::setByteProperty** (const std::string & *name*, unsigned char *value*) [pure virtual]

Sets a byte property.

Parameters

<i>name</i>	The name of the property to retrieve.
<i>value</i>	The value for the named property.

Exceptions

CMSException (p. 826)	- if the name is an empty string.
MessageNotWritableException (p. 1923)	- if properties are read-only

Implemented in **activemq::commands::ActiveMQMessageTemplate< cms::BytesMessage >** (p. 299), **activemq::commands::ActiveMQMessageTemplate< cms::MapMessage >** (p. 299), **activemq::commands::ActiveMQMessageTemplate< cms::Message >** (p. 299), **activemq::commands::ActiveMQMessageTemplate< cms::StreamMessage >** (p. 299), **activemq::commands::ActiveMQMessageTemplate< cms::TextMessage >** (p. 299), and **activemq::commands::ActiveMQMessageTemplate< cms::ObjectMessage >** (p. 299).

6.341.3.27 virtual void **cms::Message::setCMSCorrelationID** (const std::string & *correlationId*) [pure virtual]

Sets the correlation ID for the message.

A client can use the CMSCorrelationID header field to link one message with another. A typical use is to link a response message with its request message.

CMSCorrelationID can hold one of the following:

- A provider-specific message ID
- An application-specific String
- A provider-native byte[] value

Since each message sent by a CMS provider is assigned a message ID value, it is convenient to link messages via message ID. All message ID values must start with the 'ID:' prefix.

In some cases, an application (made up of several clients) needs to use an application-specific value for linking messages. For instance, an application may use CMS-CorrelationID to hold a value referencing some external information. Application-specified values must not start with the 'ID:' prefix; this is reserved for provider-generated message ID values.

If a provider supports the native concept of correlation ID, a CMS client may need to assign specific CMSCorrelationID values to match those expected by clients that do not use the CMS API. A `byte[]` value is used for this purpose. CMS providers without native correlation ID values are not required to support `byte[]` values. The use of a `byte[]` value for CMSCorrelationID is non-portable.

Parameters

<i>correlationId</i>	The message ID of a message being referred to.
----------------------	--

Exceptions

CMSException (p. 826)	- if an internal error occurs.
---------------------------------	--------------------------------

Implemented in `activemq::commands::ActiveMQMessageTemplate< cms::BytesMessage >` (p. 299), `activemq::commands::ActiveMQMessageTemplate< cms::MapMessage >` (p. 299), `activemq::commands::ActiveMQMessageTemplate< cms::Message >` (p. 299), `activemq::commands::ActiveMQMessageTemplate< cms::StreamMessage >` (p. 299), `activemq::commands::ActiveMQMessageTemplate< cms::TextMessage >` (p. 299), and `activemq::commands::ActiveMQMessageTemplate< cms::ObjectMessage >` (p. 299).

6.341.3.28 `virtual void cms::Message::setCMSDeliveryMode (int mode)` [pure virtual]

Sets the **DeliveryMode** (p. 1195) for this message.

CMS providers set this field when a message is sent. This method can be used to change the value for a message that has been received.

Parameters

<i>mode</i>	DeliveryMode (p. 1195) enumerated value.
-------------	---

Exceptions

CMSException (p. 826)	- if an internal error occurs.
---------------------------------	--------------------------------

Implemented in `activemq::commands::ActiveMQMessageTemplate< cms::BytesMessage >` (p. 299), `activemq::commands::ActiveMQMessageTemplate< cms::MapMessage >` (p. 299), `activemq::commands::ActiveMQMessageTemplate< cms::Message >` (p. 299), `activemq::commands::ActiveMQMessageTemplate<`

cms::StreamMessage > (p. 299), **activemq::commands::ActiveMQMessageTemplate< cms::TextMessage >** (p. 299), and **activemq::commands::ActiveMQMessageTemplate< cms::ObjectMessage >** (p. 299).

6.341.3.29 **virtual void cms::Message::setCMSDestination (const Destination * destination)** [pure virtual]

Sets the **Destination** (p. 1210) object for this message.

CMS providers set this field when a message is sent. This method can be used to change the value for a message that has been received.

Parameters

<i>destination</i>	Destination (p. 1210) Object
--------------------	-------------------------------------

Exceptions

CMSException (p. 826)	- if an internal error occurs.
---------------------------------	--------------------------------

Implemented in **activemq::commands::ActiveMQMessageTemplate< cms::BytesMessage >** (p. 300), **activemq::commands::ActiveMQMessageTemplate< cms::MapMessage >** (p. 300), **activemq::commands::ActiveMQMessageTemplate< cms::Message >** (p. 300), **activemq::commands::ActiveMQMessageTemplate< cms::StreamMessage >** (p. 300), **activemq::commands::ActiveMQMessageTemplate< cms::TextMessage >** (p. 300), and **activemq::commands::ActiveMQMessageTemplate< cms::ObjectMessage >** (p. 300).

6.341.3.30 **virtual void cms::Message::setCMSExpiration (long long expireTime)** [pure virtual]

Sets the message's expiration value.

CMS providers set this field when a message is sent. This method can be used to change the value for a message that has been received.

Parameters

<i>expireTime</i>	the message's expiration time
-------------------	-------------------------------

Exceptions

CMSException (p. 826)	- if an internal error occurs.
---------------------------------	--------------------------------

Implemented in **activemq::commands::ActiveMQMessageTemplate< cms::BytesMessage >** (p. 300), **activemq::commands::ActiveMQMessageTemplate< cms::MapMessage >** (p. 300), **activemq::commands::ActiveMQMessageTemplate<**

cms::Message > (p. 300), **activemq::commands::ActiveMQMessageTemplate**< **cms::StreamMessage** > (p. 300), **activemq::commands::ActiveMQMessageTemplate**< **cms::TextMessage** > (p. 300), and **activemq::commands::ActiveMQMessageTemplate**< **cms::ObjectMessage** > (p. 300).

6.341.3.31 `virtual void cms::Message::setCMSMessageID (const std::string & id)`
[pure virtual]

Sets the message ID.

CMS providers set this field when a message is sent. This method can be used to change the value for a message that has been received.

Parameters

<i>id</i>	the ID of the message
-----------	-----------------------

Exceptions

CMSException (p. 826)	- if an internal error occurs.
---------------------------------	--------------------------------

6.341.3.32 `virtual void cms::Message::setCMSPriority (int priority)` [pure virtual]

Sets the Priority Value for this message.

CMS providers set this field when a message is sent. This method can be used to change the value for a message that has been received.

Parameters

<i>priority</i>	priority value for this message
-----------------	---------------------------------

Exceptions

CMSException (p. 826)	- if an internal error occurs.
---------------------------------	--------------------------------

Implemented in **activemq::commands::ActiveMQMessageTemplate**< **cms::BytesMessage** > (p. 300), **activemq::commands::ActiveMQMessageTemplate**< **cms::MapMessage** > (p. 300), **activemq::commands::ActiveMQMessageTemplate**< **cms::Message** > (p. 300), **activemq::commands::ActiveMQMessageTemplate**< **cms::StreamMessage** > (p. 300), **activemq::commands::ActiveMQMessageTemplate**< **cms::TextMessage** > (p. 300), and **activemq::commands::ActiveMQMessageTemplate**< **cms::ObjectMessage** > (p. 300).

6.341.3.33 virtual void **cms::Message::setCMSRedelivered** (bool *redelivered*)
[pure virtual]

Specifies whether this message is being redelivered.

This field is set at the time the message is delivered. This method can be used to change the value for a message that has been received.

Parameters

<i>redelivered</i>	boolean redelivered value
--------------------	---------------------------

Exceptions

CMSException (p. 826)	- if an internal error occurs.
---------------------------------	--------------------------------

Implemented in **activemq::commands::ActiveMQMessageTemplate**< **cms::BytesMessage** > (p. 300), **activemq::commands::ActiveMQMessageTemplate**< **cms::MapMessage** > (p. 300), **activemq::commands::ActiveMQMessageTemplate**< **cms::Message** > (p. 300), **activemq::commands::ActiveMQMessageTemplate**< **cms::StreamMessage** > (p. 300), **activemq::commands::ActiveMQMessageTemplate**< **cms::TextMessage** > (p. 300), and **activemq::commands::ActiveMQMessageTemplate**< **cms::ObjectMessage** > (p. 300).

6.341.3.34 virtual void **cms::Message::setCMSReplyTo** (const **cms::Destination** * *destination*) [pure virtual]

Sets the **Destination** (p. 1210) object to which a reply to this message should be sent.

The CMSReplyTo header field contains the destination where a reply to the current message should be sent. If it is null, no reply is expected. The destination may be either a **Queue** (p. 2221) object or a **Topic** (p. 2764) object.

Messages sent with a null CMSReplyTo value may be a notification of some event, or they may just be some data the sender thinks is of interest.

Messages with a CMSReplyTo value typically expect a response. A response is optional; it is up to the client to decide. These messages are called requests. A message sent in response to a request is called a reply.

In some cases a client may wish to match a request it sent earlier with a reply it has just received. The client can use the CMSCorrelationID header field for this purpose.

Parameters

<i>destination</i>	Destination (p. 1210) to which to send a response to this message
--------------------	--

Exceptions

CMSException (p. 826)	- if an internal error occurs.
---------------------------------	--------------------------------

Implemented in **activemq::commands::ActiveMQMessageTemplate**< **cms::BytesMessage** > (p. 300), **activemq::commands::ActiveMQMessageTemplate**< **cms::MapMessage** > (p. 300), **activemq::commands::ActiveMQMessageTemplate**< **cms::Message** > (p. 300), **activemq::commands::ActiveMQMessageTemplate**< **cms::StreamMessage** > (p. 300), **activemq::commands::ActiveMQMessageTemplate**< **cms::TextMessage** > (p. 300), and **activemq::commands::ActiveMQMessageTemplate**< **cms::ObjectMessage** > (p. 300).

6.341.3.35 **virtual void cms::Message::setCMSTimestamp (long long timeStamp)**
[pure virtual]

Sets the message timestamp.

CMS providers set this field when a message is sent. This method can be used to change the value for a message that has been received.

Parameters

<i>timeStamp</i>	integer time stamp value
------------------	--------------------------

Exceptions

CMSException (p. 826)	- if an internal error occurs.
---------------------------------	--------------------------------

Implemented in **activemq::commands::ActiveMQMessageTemplate**< **cms::BytesMessage** > (p. 300), **activemq::commands::ActiveMQMessageTemplate**< **cms::MapMessage** > (p. 300), **activemq::commands::ActiveMQMessageTemplate**< **cms::Message** > (p. 300), **activemq::commands::ActiveMQMessageTemplate**< **cms::StreamMessage** > (p. 300), **activemq::commands::ActiveMQMessageTemplate**< **cms::TextMessage** > (p. 300), and **activemq::commands::ActiveMQMessageTemplate**< **cms::ObjectMessage** > (p. 300).

6.341.3.36 **virtual void cms::Message::setCMSType (const std::string & type)** [pure virtual]

Sets the message type.

Some CMS providers use a message repository that contains the definitions of messages sent by applications. The CMSType header field may reference a message's definition in the provider's repository.

The CMS API does not define a standard message definition repository, nor does it define a naming policy for the definitions it contains.

Some messaging systems require that a message type definition for each application message be created and that each message specify its type. In order to work with such CMS providers, CMS clients should assign a value to CMSType, whether the application makes use of it or not. This ensures that the field is properly set for those providers that require it.

To ensure portability, CMS clients should use symbolic values for CMSType that can be configured at installation time to the values defined in the current provider's message repository. If string literals are used, they may not be valid type names for some CMS providers.

Parameters

<i>type</i>	the message type
-------------	------------------

See also

getCMSType (p. 1851)

Exceptions

CMSException (p. 826)	- if an internal error occurs.
---------------------------------	--------------------------------

Implemented in **activemq::commands::ActiveMQMessageTemplate< cms::BytesMessage >** (p. 300), **activemq::commands::ActiveMQMessageTemplate< cms::MapMessage >** (p. 300), **activemq::commands::ActiveMQMessageTemplate< cms::Message >** (p. 300), **activemq::commands::ActiveMQMessageTemplate< cms::StreamMessage >** (p. 300), **activemq::commands::ActiveMQMessageTemplate< cms::TextMessage >** (p. 300), and **activemq::commands::ActiveMQMessageTemplate< cms::ObjectMessage >** (p. 300).

6.341.3.37 **virtual void cms::Message::setDoubleProperty (const std::string & name, double value)** [pure virtual]

Sets a double property.

Parameters

<i>name</i>	The name of the property to retrieve.
<i>value</i>	The value for the named property.

Exceptions

CMSException (p. 826)	- if the name is an empty string.
MessageNotWritableException (p. 1923)	- if properties are read-only

Implemented in **activemq::commands::ActiveMQMessageTemplate< cms::BytesMessage >** (p. 300), **activemq::commands::ActiveMQMessageTemplate< cms::MapMessage >** (p. 300), **activemq::commands::ActiveMQMessageTemplate< cms::Message >** (p. 300), **activemq::commands::ActiveMQMessageTemplate<**

cms::StreamMessage > (p. 300), **activemq::commands::ActiveMQMessageTemplate**< **cms::TextMessage** > (p. 300), and **activemq::commands::ActiveMQMessageTemplate**< **cms::ObjectMessage** > (p. 300).

6.341.3.38 `virtual void cms::Message::setFloatProperty (const std::string & name, float value) [pure virtual]`

Sets a float property.

Parameters

<i>name</i>	The name of the property to retrieve.
<i>value</i>	The value for the named property.

Exceptions

CMSException (p. 826)	- if the name is an empty string.
MessageNotWriteableException (p. 1923)	- if properties are read-only

Implemented in **activemq::commands::ActiveMQMessageTemplate**< **cms::BytesMessage** > (p. 300), **activemq::commands::ActiveMQMessageTemplate**< **cms::MapMessage** > (p. 300), **activemq::commands::ActiveMQMessageTemplate**< **cms::Message** > (p. 300), **activemq::commands::ActiveMQMessageTemplate**< **cms::StreamMessage** > (p. 300), **activemq::commands::ActiveMQMessageTemplate**< **cms::TextMessage** > (p. 300), and **activemq::commands::ActiveMQMessageTemplate**< **cms::ObjectMessage** > (p. 300).

6.341.3.39 `virtual void cms::Message::setIntProperty (const std::string & name, int value) [pure virtual]`

Sets a int property.

Parameters

<i>name</i>	The name of the property to retrieve.
<i>value</i>	The value for the named property.

Exceptions

CMSException (p. 826)	- if the name is an empty string.
MessageNotWriteableException (p. 1923)	- if properties are read-only

Implemented in `activemq::commands::ActiveMQMessageTemplate< cms::BytesMessage >` (p. 300), `activemq::commands::ActiveMQMessageTemplate< cms::MapMessage >` (p. 300), `activemq::commands::ActiveMQMessageTemplate< cms::Message >` (p. 300), `activemq::commands::ActiveMQMessageTemplate< cms::StreamMessage >` (p. 300), `activemq::commands::ActiveMQMessageTemplate< cms::TextMessage >` (p. 300), and `activemq::commands::ActiveMQMessageTemplate< cms::ObjectMessage >` (p. 300).

6.341.3.40 `virtual void cms::Message::setLongProperty (const std::string & name, long long value) [pure virtual]`

Sets a long property.

Parameters

<i>name</i>	The name of the property to retrieve.
<i>value</i>	The value for the named property.

Exceptions

<i>CMSException</i> (p. 826)	- if the name is an empty string.
<i>MessageNotWritableException</i> (p. 1923)	- if properties are read-only

Implemented in `activemq::commands::ActiveMQMessageTemplate< cms::BytesMessage >` (p. 300), `activemq::commands::ActiveMQMessageTemplate< cms::MapMessage >` (p. 300), `activemq::commands::ActiveMQMessageTemplate< cms::Message >` (p. 300), `activemq::commands::ActiveMQMessageTemplate< cms::StreamMessage >` (p. 300), `activemq::commands::ActiveMQMessageTemplate< cms::TextMessage >` (p. 300), and `activemq::commands::ActiveMQMessageTemplate< cms::ObjectMessage >` (p. 300).

6.341.3.41 `virtual void cms::Message::setShortProperty (const std::string & name, short value) [pure virtual]`

Sets a short property.

Parameters

<i>name</i>	The name of the property to retrieve.
<i>value</i>	The value for the named property.

Exceptions

<i>CMSException</i> (p. 826)	- if the name is an empty string.
--	-----------------------------------

<i>MessageNot-WriteableException</i> (p. 1923)	- if properties are read-only
--	-------------------------------

Implemented in **activemq::commands::ActiveMQMessageTemplate**< **cms::BytesMessage** > (p. 301), **activemq::commands::ActiveMQMessageTemplate**< **cms::MapMessage** > (p. 301), **activemq::commands::ActiveMQMessageTemplate**< **cms::Message** > (p. 301), **activemq::commands::ActiveMQMessageTemplate**< **cms::StreamMessage** > (p. 301), **activemq::commands::ActiveMQMessageTemplate**< **cms::TextMessage** > (p. 301), and **activemq::commands::ActiveMQMessageTemplate**< **cms::ObjectMessage** > (p. 301).

6.341.3.42 **virtual void cms::Message::setStringProperty (const std::string & name, const std::string & value)** [pure virtual]

Sets a string property.

Parameters

<i>name</i>	The name of the property to retrieve.
<i>value</i>	The value for the named property.

Exceptions

<i>CMSException</i> (p. 826)	- if the name is an empty string.
<i>MessageNot-WriteableException</i> (p. 1923)	- if properties are read-only

Implemented in **activemq::commands::ActiveMQMessageTemplate**< **cms::BytesMessage** > (p. 301), **activemq::commands::ActiveMQMessageTemplate**< **cms::MapMessage** > (p. 301), **activemq::commands::ActiveMQMessageTemplate**< **cms::Message** > (p. 301), **activemq::commands::ActiveMQMessageTemplate**< **cms::StreamMessage** > (p. 301), **activemq::commands::ActiveMQMessageTemplate**< **cms::TextMessage** > (p. 301), and **activemq::commands::ActiveMQMessageTemplate**< **cms::ObjectMessage** > (p. 301).

6.341.4 Field Documentation

6.341.4.1 **const int cms::Message::DEFAULT_DELIVERY_MODE** [static]

The Default delivery mode for **Message** (p. 1839) Producers is PERSISTENT.

6.341.4.2 `const int cms::Message::DEFAULT_MSG_PRIORITY` `[static]`

The Default priority assigned to a **Message** (p. 1839) is 4.

6.341.4.3 `const long long cms::Message::DEFAULT_TIME_TO_LIVE` `[static]`

The Default Time to Live for a **Message** (p. 1839) Producer is unlimited, the message will never expire.

The documentation for this class was generated from the following file:

- `src/main/cms/Message.h`

6.342 activemq::commands::MessageAck Class Reference

```
#include <src/main/activemq/commands/MessageAck.h>
```

Inheritance diagram for `activemq::commands::MessageAck`:

Public Member Functions

- **MessageAck** ()
- virtual **~MessageAck** ()
- virtual unsigned char **getDataStructureType** () const
*Get the **DataStructure** (p. 1133) Type as defined in CommandTypes.h.*
- virtual **MessageAck * cloneDataStructure** () const
Clone this object and return a new instance that the caller now owns, this will be an exact copy of this one.
- virtual void **copyDataStructure** (const **DataStructure** *src)
Copy the contents of the passed object into this objects members, overwriting any existing data.
- virtual std::string **toString** () const
*Returns a string containing the information for this **DataStructure** (p. 1133) such as its type and value of its elements.*
- virtual bool **equals** (const **DataStructure** *value) const
*Compares the **DataStructure** (p. 1133) passed in to this one, and returns if they are equivalent.*
- virtual const **Pointer** < **ActiveMQDestination** > & **getDestination** () const
- virtual **Pointer** < **ActiveMQDestination** > & **getDestination** ()
- virtual void **setDestination** (const **Pointer**< **ActiveMQDestination** > &destination)
- virtual const **Pointer** < **TransactionId** > & **getTransactionId** () const
- virtual **Pointer**< **TransactionId** > & **getTransactionId** ()

- virtual void **setTransactionId** (const **Pointer**< **TransactionId** > &**transactionId**)
- virtual const **Pointer** < **ConsumerId** > & **getConsumerId** () const
- virtual **Pointer**< **ConsumerId** > & **getConsumerId** ()
- virtual void **setConsumerId** (const **Pointer**< **ConsumerId** > &**consumerId**)
- virtual unsigned char **getAckType** () const
- virtual void **setAckType** (unsigned char **ackType**)
- virtual const **Pointer** < **MessageId** > & **getFirstMessageId** () const
- virtual **Pointer**< **MessageId** > & **getFirstMessageId** ()
- virtual void **setFirstMessageId** (const **Pointer**< **MessageId** > &**firstMessageId**)
- virtual const **Pointer** < **MessageId** > & **getLastMessageId** () const
- virtual **Pointer**< **MessageId** > & **getLastMessageId** ()
- virtual void **setLastMessageId** (const **Pointer**< **MessageId** > &**lastMessageId**)
- virtual int **getMessageCount** () const
- virtual void **setMessageCount** (int **messageCount**)
- virtual bool **isMessageAck** () const
- virtual **Pointer**< **Command** > **visit** (**activemq::state::CommandVisitor** ***visitor**)

Allows a Visitor to visit this command and return a response to the command based on the command type being visited.

Static Public Attributes

- static const unsigned char **ID_MESSAGEACK** = 22

Protected Attributes

- **Pointer**< **ActiveMQDestination** > **destination**
- **Pointer**< **TransactionId** > **transactionId**
- **Pointer**< **ConsumerId** > **consumerId**
- unsigned char **ackType**
- **Pointer**< **MessageId** > **firstMessageId**
- **Pointer**< **MessageId** > **lastMessageId**
- int **messageCount**

6.342.1 Constructor & Destructor Documentation

6.342.1.1 **activemq::commands::MessageAck::MessageAck** ()

6.342.1.2 **virtual activemq::commands::MessageAck::~MessageAck** ()
[virtual]

6.342.2 Member Function Documentation

6.342.2.1 virtual **MessageAck*** **activemq::commands::MessageAck::cloneData-Structure ()** const [virtual]

Clone this obbject and return a new instance that the caller now owns, this will be an exact copy of this one.

Returns

new copy of this object.

Implements **activemq::commands::DataStructure** (p. 1133).

6.342.2.2 virtual void **activemq::commands::MessageAck::copyDataStructure (const DataStructure * src)** [virtual]

Copy the contents of the passed object into this objects members, overwriting any existing data.

Returns

src - Source Object

Reimplemented from **activemq::commands::BaseCommand** (p. 493).

6.342.2.3 virtual bool **activemq::commands::MessageAck::equals (const DataStructure * value)** const [virtual]

Compares the **DataStructure** (p. 1133) passed in to this one, and returns if they are equivalent.

Equivalent here means that they are of the same type, and that each element of the objects are the same.

Returns

true if **DataStructure** (p. 1133)'s are Equal.

Reimplemented from **activemq::commands::BaseCommand** (p. 494).

6.342.2.4 virtual unsigned char **activemq::commands::MessageAck::getAckType ()** const [virtual]

6.342.2.5 virtual const **Pointer<ConsumerId>&** **activemq::commands::MessageAck::getConsumerId ()** const [virtual]

6.342.2.6 virtual **Pointer<ConsumerId>&** **activemq::commands::MessageAck::getConsumerId ()** [virtual]

6.342.2.7 virtual unsigned char **activemq::commands::MessageAck::getData-StructureType**() const [virtual]

Get the **DataStructure** (p. 1133) Type as defined in CommandTypes.h.

Returns

The type of the data structure

Implements **activemq::commands::DataStructure** (p. 1137).

6.342.2.8 virtual const **Pointer**<**ActiveMQDestination**>& **activemq::commands::MessageAck::getDestination**() const [virtual]

6.342.2.9 virtual **Pointer**<**ActiveMQDestination**>& **activemq::commands::MessageAck::getDestination**() [virtual]

6.342.2.10 virtual const **Pointer**<**MessageId**>& **activemq::commands::MessageAck::getFirstMessageId**() const [virtual]

6.342.2.11 virtual **Pointer**<**MessageId**>& **activemq::commands::MessageAck::getFirstMessageId**() [virtual]

6.342.2.12 virtual const **Pointer**<**MessageId**>& **activemq::commands::MessageAck::getLastMessageId**() const [virtual]

6.342.2.13 virtual **Pointer**<**MessageId**>& **activemq::commands::MessageAck::getLastMessageId**() [virtual]

6.342.2.14 virtual int **activemq::commands::MessageAck::getMessageCount**() const [virtual]

6.342.2.15 virtual const **Pointer**<**TransactionId**>& **activemq::commands::MessageAck::getTransactionId**() const [virtual]

6.342.2.16 virtual **Pointer**<**TransactionId**>& **activemq::commands::MessageAck::getTransactionId**() [virtual]

6.342.2.17 virtual bool **activemq::commands::MessageAck::isMessageAck**() const [inline, virtual]

Returns

an answer of true to the **isMessageAck()** (p. 1871) query.

Reimplemented from **activemq::commands::BaseCommand** (p. 496).

6.342.2.18 `virtual void activemq::commands::MessageAck::setAckType (unsigned char ackType) [virtual]`

6.342.2.19 `virtual void activemq::commands::MessageAck::setConsumerId (const Pointer< ConsumerId > & consumerId) [virtual]`

6.342.2.20 `virtual void activemq::commands::MessageAck::setDestination (const Pointer< ActiveMQDestination > & destination) [virtual]`

6.342.2.21 `virtual void activemq::commands::MessageAck::setFirstMessageld (const Pointer< Messageld > & firstMessageld) [virtual]`

6.342.2.22 `virtual void activemq::commands::MessageAck::setLastMessageld (const Pointer< Messageld > & lastMessageld) [virtual]`

6.342.2.23 `virtual void activemq::commands::MessageAck::setMessageCount (int messageCount) [virtual]`

6.342.2.24 `virtual void activemq::commands::MessageAck::setTransactionId (const Pointer< TransactionId > & transactionId) [virtual]`

6.342.2.25 `virtual std::string activemq::commands::MessageAck::toString () const [virtual]`

Returns a string containing the information for this **DataStructure** (p. 1133) such as its type and value of its elements.

Returns

formatted string useful for debugging.

Reimplemented from **activemq::commands::BaseCommand** (p. 498).

6.342.2.26 `virtual Pointer<Command> activemq::commands::MessageAck::visit (activemq::state::CommandVisitor * visitor) [virtual]`

Allows a Visitor to visit this command and return a response to the command based on the command type being visited.

The command will call the proper processXXX method in the visitor.

Returns

a **Response** (p. 2298) to the visitor being called or NULL if no response.

Implements **activemq::commands::Command** (p. 871).

6.342.3 Field Documentation

- 6.342.3.1 **unsigned char activemq::commands::MessageAck::ackType**
[protected]
- 6.342.3.2 **Pointer<ConsumerId> activemq::commands::MessageAck::consumerId** [protected]
- 6.342.3.3 **Pointer<ActiveMQDestination> activemq::commands::MessageAck::destination** [protected]
- 6.342.3.4 **Pointer<MessageId> activemq::commands::MessageAck::first-MessageId** [protected]
- 6.342.3.5 **const unsigned char activemq::commands::MessageAck::ID_MESSAGEACK = 22** [static]
- 6.342.3.6 **Pointer<MessageId> activemq::commands::MessageAck::last-MessageId** [protected]
- 6.342.3.7 **int activemq::commands::MessageAck::messageCount**
[protected]
- 6.342.3.8 **Pointer<TransactionId> activemq::commands::MessageAck::transactionId** [protected]

The documentation for this class was generated from the following file:

- src/main/activemq/commands/**MessageAck.h**

6.343 **activemq::wireformat::openwire::marshal::generated::MessageAckMarshaller** Class Reference

Marshaling code for Open Wire Format for **MessageAckMarshaller** (p. 1873).

```
#include <src/main/activemq/wireformat/openwire/marshal/generated/MessageAckMarshaller.h>
```

Inheritance diagram for **activemq::wireformat::openwire::marshal::generated::MessageAckMarshaller**:

- **MessageAckMarshaller** ()
- virtual **~MessageAckMarshaller** ()
- virtual **commands::DataStructure * createObject** () const
Creates a new instance of the class that this class is a marshaling director for.
- virtual unsigned char **getDataStructureType** () const
Gets the DataStructureType that this class marshals/unmarshals.
- virtual void **tightUnmarshal** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataInputStream** *dataIn, **utils::BooleanStream** *bs)
Tight Un-marhsal to the given stream.
- virtual int **tightMarshal1** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **utils::BooleanStream** *bs)
Tight Marshal to the given stream.
- virtual void **tightMarshal2** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataOutputStream** *dataOut, **utils::BooleanStream** *bs)
Tight Marshal to the given stream.
- virtual void **looseUnmarshal** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataInputStream** *dataIn)
Loose Un-marhsal to the given stream.
- virtual void **looseMarshal** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataOutputStream** *dataOut)
Tight Marshal to the given stream.

6.343.1 Detailed Description

Marshaling code for Open Wire Format for **MessageAckMarshaller** (p. 1873).

NOTE!: This file is auto generated - do not modify! if you need to make a change, please see the Java Classes in the activemq-openwire-generator module

6.343.2 Constructor & Destructor Documentation

6.343.2.1 **activemq::wireformat::openwire::marshal::generated::MessageAckMarshaller::MessageAckMarshaller** ()
[inline]

6.343.2.2 **virtual activemq::wireformat::openwire::marshal::generated::MessageAckMarshaller::~~MessageAckMarshaller** () [inline, virtual]

6.343.3 Member Function Documentation

6.343.3.1 `virtual commands::DataStructure* activemq::wireformat::openwire-
::marshal::generated::MessageAckMarshaller::createObject () const
[virtual]`

Creates a new instance of the class that this class is a marshaling director for.

Returns

newly allocated Command

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller**
(p. 1120).

6.343.3.2 `virtual unsigned char activemq::wireformat::openwire::marshal-
::generated::MessageAckMarshaller::getDataStructureType () const
[virtual]`

Gets the DataStructureType that this class marshals/unmarshals.

Returns

byte Id of this classes DataStructureType

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller**
(p. 1122).

6.343.3.3 `virtual void activemq::wireformat::openwire::marshal::generated-
::MessageAckMarshaller::looseMarshal (OpenWireFormat * format,
commands::DataStructure * command, decaf::io::DataOutputStream * ds
) [virtual]`

Tight Marshal to the given stream.

Parameters

<i>format</i>	- The OpenWireFormat properties
<i>command</i>	- the object to Marshal
<i>ds</i>	- DataOutputStream to marshal to

Exceptions

<i>IOException</i> if an error occurs.
--

Reimplemented from **activemq::wireformat::openwire::marshal::generated::Base-
CommandMarshaller** (p. 500).

6.343

activemq::wireformat::openwire::marshal::generated::MessageAckMarshaller

Class Reference

1881

6.343.3.4 virtual void **activemq::wireformat::openwire::marshal::generated::MessageAckMarshaller::looseUnmarshal** (**OpenWireFormat** * *format*, **commands::DataStructure** * *command*, **decaf::io::DataInputStream** * *dis*)
[virtual]

Loose Un-marhsal to the given stream.

Parameters

<i>format</i>	- The OpenWireFormat properties
<i>command</i>	- the object to Un-Marshal
<i>dis</i>	- the DataInputStream to Un-Marshal from

Exceptions

<i>IOException</i>	if an error occurs.
--------------------	---------------------

Reimplemented from **activemq::wireformat::openwire::marshal::generated::BaseCommandMarshaller** (p. 501).

6.343.3.5 virtual int **activemq::wireformat::openwire::marshal::generated::MessageAckMarshaller::tightMarshal1** (**OpenWireFormat** * *format*, **commands::DataStructure** * *command*, **utils::BooleanStream** * *bs*)
[virtual]

Tight Marhsal to the given stream.

Parameters

<i>format</i>	The OpenWireFormat properties
<i>command</i>	The object to Marshal
<i>bs</i>	The boolean stream to marshal to.

Exceptions

<i>IOException</i>	if an error occurs.
--------------------	---------------------

Reimplemented from **activemq::wireformat::openwire::marshal::generated::BaseCommandMarshaller** (p. 503).

6.343.3.6 virtual void **activemq::wireformat::openwire::marshal::generated::MessageAckMarshaller::tightMarshal2** (**OpenWireFormat** * *format*, **commands::DataStructure** * *command*, **decaf::io::DataOutputStream** * *ds*, **utils::BooleanStream** * *bs*) [virtual]

Tight Marhsal to the given stream.

Parameters

<i>format</i>	- The OpenwireFormat properties
<i>command</i>	- the object to Marshal
<i>ds</i>	- the DataOutputStream to Marshal to
<i>bs</i>	- boolean stream to marshal to.

Exceptions

<i>IOException</i>	if an error occurs.
--------------------	---------------------

Reimplemented from **activemq::wireformat::openwire::marshal::generated::Base-CommandMarshaller** (p. 504).

6.343.3.7 virtual void **activemq::wireformat::openwire::marshal::generated::MessageAckMarshaller::tightUnmarshal** (**OpenWireFormat** * *format*, **commands::DataStructure** * *command*, **decaf::io::DataInputStream** * *dis*, **utils::BooleanStream** * *bs*) [virtual]

Tight Un-marhsal to the given stream.

Parameters

<i>format</i>	- The OpenwireFormat properties
<i>command</i>	- the object to Un-Marshal
<i>dis</i>	- the DataInputStream to Un-Marshal from
<i>bs</i>	- boolean stream to unmarshal from.

Exceptions

<i>IOException</i>	if an error occurs.
--------------------	---------------------

Reimplemented from **activemq::wireformat::openwire::marshal::generated::Base-CommandMarshaller** (p. 505).

The documentation for this class was generated from the following file:

- `src/main/activemq/wireformat/openwire/marshal/generated/MessageAck-Marshaller.h`

6.344 cms::MessageConsumer Class Reference

A client uses a **MessageConsumer** (p. 1877) to received messages from a destination.

```
#include <src/main/cms/MessageConsumer.h>
```


Inheritance diagram for cms::MessageConsumer:

Public Member Functions

- virtual **~MessageConsumer** () throw ()
- virtual **Message * receive** ()=0
*Synchronously Receive a **Message** (p. 1839).*
- virtual **Message * receive** (int millisecs)=0
*Synchronously Receive a **Message** (p. 1839), time out after defined interval.*
- virtual **Message * receiveNoWait** ()=0
*Receive a **Message** (p. 1839), does not wait if there isn't a new message to read, returns NULL if nothing read.*
- virtual void **setMessageListener** (MessageListener *listener)=0
*Sets the **MessageListener** (p. 1916) that this class will send notifis on.*
- virtual **MessageListener * getMessageListener** () const =0
*Gets the **MessageListener** (p. 1916) that this class will send mew **Message** (p. 1839) notification events to.*
- virtual std::string **getMessageSelector** () const =0
Gets this message consumer's message selector expression.

6.344.1 Detailed Description

A client uses a **MessageConsumer** (p. 1877) to received messages from a destination.

A client may either synchronously receive a message consumer's messages or have the consumer asynchronously deliver them as they arrive.

For synchronous receipt, a client can request the next message from a message consumer using one of its `receive` methods. There are several variations of `receive` that allow a client to poll or wait for the next message.

For asynchronous delivery, a client can register a **MessageListener** (p. 1916) object with a message consumer. As messages arrive at the message consumer, it delivers them by calling the **MessageListener** (p. 1916)'s `onMessage` method.

When the **MessageConsumer** (p. 1877)'s `close` method is called the method can block while an asynchronous message delivery is in progress or until a `receive` operation completes. A blocked consumer in a `receive` call will return a Null when the `close` method is called.

While the **MessageConsumer** (p. 1877) implements the **Startable** (p. 2534) and **Stoppable** (p. 2602) interfaces it is not required to implement these methods and can throw an `UnsupportedOperation` exception if they are not available for the given CMS provider.

See also

MessageListener (p. 1916)

Since

1.0

6.344.2 Constructor & Destructor Documentation

6.344.2.1 `virtual cms::MessageConsumer::~MessageConsumer () throw ()`
[virtual]

6.344.3 Member Function Documentation

6.344.3.1 `virtual MessageListener* cms::MessageConsumer::getMessageListener () const` [pure virtual]

Gets the **MessageListener** (p. 1916) that this class will send new **Message** (p. 1839) notification events to.

Returns

The listener of messages received by this consumer

Exceptions

CMSException (p. 826)	- If an internal error occurs.
---------------------------------	--------------------------------

Implemented in **activemq::core::ActiveMQConsumer** (p. 240), and **activemq::cmsutil::CachedConsumer** (p. 736).

6.344.3.2 `virtual std::string cms::MessageConsumer::getMessageSelector () const`
[pure virtual]

Gets this message consumer's message selector expression.

Returns

This Consumer's selector expression or "".

Exceptions

CMSException (p. 826)	- If an internal error occurs.
---------------------------------	--------------------------------

Implemented in **activemq::core::ActiveMQConsumer** (p.240), and **activemq::cmsutil::CachedConsumer** (p.736).

6.344.3.3 virtual **Message*** **cms::MessageConsumer::receive** () [pure virtual]

Synchronously Receive a **Message** (p.1839).

Returns

new message which the caller owns and must delete.

Exceptions

CMSException (p.826)	- If an internal error occurs.
--------------------------------	--------------------------------

Implemented in **activemq::core::ActiveMQConsumer** (p.241), and **activemq::cmsutil::CachedConsumer** (p.736).

6.344.3.4 virtual **Message*** **cms::MessageConsumer::receive** (int *milliseconds*) [pure virtual]

Synchronously Receive a **Message** (p.1839), time out after defined interval.

Returns null if nothing read.

Returns

new message which the caller owns and must delete.

Exceptions

CMSException (p.826)	- If an internal error occurs.
--------------------------------	--------------------------------

Implemented in **activemq::core::ActiveMQConsumer** (p.242), and **activemq::cmsutil::CachedConsumer** (p.737).

6.344.3.5 virtual **Message*** **cms::MessageConsumer::receiveNoWait** () [pure virtual]

Receive a **Message** (p.1839), does not wait if there isn't a new message to read, returns NULL if nothing read.

Returns

new message which the caller owns and must delete.

Exceptions

<i>CMSEException</i> (p. 826)	- If an internal error occurs.
---	--------------------------------

Implemented in **activemq::core::ActiveMQConsumer** (p.242), and **activemq-
::cmsutil::CachedConsumer** (p. 737).

6.344.3.6 virtual void **cms::MessageConsumer::setMessageListener** (
MessageListener * listener) [pure virtual]

Sets the **MessageListener** (p. 1916) that this class will send notifs on.

Parameters

<i>listener</i>	The listener of messages received by this consumer.
-----------------	---

Exceptions

<i>CMSEException</i> (p. 826)	- If an internal error occurs.
---	--------------------------------

Implemented in **activemq::core::ActiveMQConsumer** (p.243), and **activemq-
::cmsutil::CachedConsumer** (p. 737).

The documentation for this class was generated from the following file:

- src/main/cms/**MessageConsumer.h**

6.345 activemq::cmsutil::MessageCreator Class Reference

Creates the user-defined message to be sent by the **CmsTemplate** (p. 836).

```
#include <src/main/activemq/cmsutil/MessageCreator.h>
```

Public Member Functions

- virtual **~MessageCreator** () throw ()
- virtual **cms::Message * createMessage** (**cms::Session *session**)=0
Creates a message from the given session.

6.345.1 Detailed Description

Creates the user-defined message to be sent by the **CmsTemplate** (p. 836).

6.345.2 Constructor & Destructor Documentation

6.345.2.1 `virtual activemq::cmsutil::MessageCreator::~MessageCreator () throw ()`
[inline, virtual]

6.345.3 Member Function Documentation

6.345.3.1 `virtual cms::Message* activemq::cmsutil::MessageCreator-
::createMessage (cms::Session * session)` [pure
virtual]

Creates a message from the given session.

Parameters

<i>session</i>	the CMS Session
----------------	-----------------

Exceptions

<i>cms::CMS- Exception</i> (p. 826)	if thrown by CMS API methods
---	------------------------------

The documentation for this class was generated from the following file:

- src/main/activemq/cmsutil/**MessageCreator.h**

6.346 activemq::commands::MessageDispatch Class Reference

```
#include <src/main/activemq/commands/MessageDispatch.h>
```

Inheritance diagram for activemq::commands::MessageDispatch:

Public Member Functions

- **MessageDispatch** ()
- virtual **~MessageDispatch** ()
- virtual unsigned char **getDataStructureType** () const

Get the **DataStructure** (p. 1133) Type as defined in *CommandTypes.h*.

- virtual **MessageDispatch** * **cloneDataStructure** () const
Clone this object and return a new instance that the caller now owns, this will be an exact copy of this one.
- virtual void **copyDataStructure** (const **DataStructure** *src)
Copy the contents of the passed object into this objects members, overwriting any existing data.
- virtual std::string **toString** () const
*Returns a string containing the information for this **DataStructure** (p. 1133) such as its type and value of its elements.*
- virtual bool **equals** (const **DataStructure** *value) const
*Compares the **DataStructure** (p. 1133) passed in to this one, and returns if they are equivalent.*
- virtual const **Pointer** < **ConsumerId** > & **getConsumerId** () const
- virtual **Pointer**< **ConsumerId** > & **getConsumerId** ()
- virtual void **setConsumerId** (const **Pointer**< **ConsumerId** > &consumerId)
- virtual const **Pointer** < **ActiveMQDestination** > & **getDestination** () const
- virtual **Pointer** < **ActiveMQDestination** > & **getDestination** ()
- virtual void **setDestination** (const **Pointer**< **ActiveMQDestination** > &destination)
- virtual const **Pointer**< **Message** > & **getMessage** () const
- virtual **Pointer**< **Message** > & **getMessage** ()
- virtual void **setMessage** (const **Pointer**< **Message** > &message)
- virtual int **getRedeliveryCounter** () const
- virtual void **setRedeliveryCounter** (int redeliveryCounter)
- virtual bool **isMessageDispatch** () const
- virtual **Pointer**< **Command** > **visit** (activemq::state::CommandVisitor *visitor)
Allows a Visitor to visit this command and return a response to the command based on the command type being visited.

Static Public Attributes

- static const unsigned char **ID_MESSAGEDISPATCH** = 21

Protected Attributes

- **Pointer**< **ConsumerId** > consumerId
- **Pointer**< **ActiveMQDestination** > destination
- **Pointer**< **Message** > message
- int redeliveryCounter

6.346.1 Constructor & Destructor Documentation

6.346.1.1 `activemq::commands::MessageDispatch::MessageDispatch ()`

6.346.1.2 `virtual activemq::commands::MessageDispatch::~~MessageDispatch ()`
[virtual]

6.346.2 Member Function Documentation

6.346.2.1 `virtual MessageDispatch* activemq::commands::-`
`MessageDispatch::cloneDataStructure () const`
[virtual]

Clone this obbject and return a new instance that the caller now owns, this will be an exact copy of this one.

Returns

new copy of this object.

Implements **activemq::commands::DataStructure** (p. 1133).

6.346.2.2 `virtual void activemq::commands::MessageDispatch::copyDataStructure`
`(const DataStructure * src)` [virtual]

Copy the contents of the passed object into this objects members, overwriting any existing data.

Returns

src - Source Object

Reimplemented from **activemq::commands::BaseCommand** (p. 493).

6.346.2.3 `virtual bool activemq::commands::MessageDispatch::equals (const`
`DataStructure * value) const` [virtual]

Compares the **DataStructure** (p. 1133) passed in to this one, and returns if they are equivalent.

Equivalent here means that they are of the same type, and that each element of the objects are the same.

Returns

true if **DataStructure** (p. 1133)'s are Equal.

Reimplemented from **activemq::commands::BaseCommand** (p. 494).

- 6.346.2.4 `virtual const Pointer<ConsumerId>& activemq::commands-
::MessageDispatch::getConsumerId () const
[virtual]`
- 6.346.2.5 `virtual Pointer<ConsumerId>& activemq::commands::Message-
Dispatch::getConsumerId () [virtual]`
- 6.346.2.6 `virtual unsigned char activemq::commands::MessageDispatch::getData-
StructureType () const [virtual]`

Get the **DataStructure** (p. 1133) Type as defined in CommandTypes.h.

Returns

The type of the data structure

Implements **activemq::commands::DataStructure** (p. 1137).

- 6.346.2.7 `virtual const Pointer<ActiveMQDestination>& activemq-
::commands::MessageDispatch::getDestination () const
[virtual]`
- 6.346.2.8 `virtual Pointer<ActiveMQDestination>& activemq-
::commands::MessageDispatch::getDestination ()
[virtual]`
- 6.346.2.9 `virtual const Pointer<Message>& activemq::commands::Message-
Dispatch::getMessage () const [virtual]`
- 6.346.2.10 `virtual Pointer<Message>& activemq::commands::MessageDispatch-
::getMessage () [virtual]`
- 6.346.2.11 `virtual int activemq::commands::MessageDispatch::getRedelivery-
Counter () const [virtual]`
- 6.346.2.12 `virtual bool activemq::commands::MessageDispatch-
::isMessageDispatch () const [inline,
virtual]`

Returns

an answer of true to the **isMessageDispatch()** (p. 1884) query.

Reimplemented from **activemq::commands::BaseCommand** (p. 496).

- 6.346.2.13 `virtual void activemq::commands::MessageDispatch::setConsumerId (
const Pointer< ConsumerId > & consumerId) [virtual]`

- 6.346.2.14 `virtual void activemq::commands::MessageDispatch::setDestination (const Pointer< ActiveMQDestination > & destination) [virtual]`
- 6.346.2.15 `virtual void activemq::commands::MessageDispatch::setMessage (const Pointer< Message > & message) [virtual]`
- 6.346.2.16 `virtual void activemq::commands::MessageDispatch-
::setRedeliveryCounter (int redeliveryCounter)
[virtual]`
- 6.346.2.17 `virtual std::string activemq::commands::MessageDispatch::toString ()
const [virtual]`

Returns a string containing the information for this **DataSet** (p. 1133) such as its type and value of its elements.

Returns

formatted string useful for debugging.

Reimplemented from **activemq::commands::BaseCommand** (p. 498).

- 6.346.2.18 `virtual Pointer<Command> activemq::commands::Message-
Dispatch::visit (activemq::state::CommandVisitor * visitor)
[virtual]`

Allows a Visitor to visit this command and return a response to the command based on the command type being visited.

The command will call the proper processXXX method in the visitor.

Returns

a **Response** (p. 2298) to the visitor being called or NULL if no response.

Implements **activemq::commands::Command** (p. 871).

6.346.3 Field Documentation

- 6.346.3.1 `Pointer<ConsumerId> activemq::commands::MessageDispatch-
::consumerId [protected]`
- 6.346.3.2 `Pointer<ActiveMQDestination> activemq::commands::Message-
Dispatch::destination [protected]`
- 6.346.3.3 `const unsigned char activemq::commands::MessageDispatch::ID_MESSA-
GEDISPATCH = 21 [static]`

6.346.3.4 **Pointer<Message> activemq::commands::MessageDispatch::message**
[protected]

6.346.3.5 **int activemq::commands::MessageDispatch::redeliveryCounter**
[protected]

The documentation for this class was generated from the following file:

- src/main/activemq/commands/**MessageDispatch.h**

6.347 activemq::core::MessageDispatchChannel Class Reference

```
#include <src/main/activemq/core/MessageDispatchChannel.h>
```

Inheritance diagram for activemq::core::MessageDispatchChannel:

Public Member Functions

- virtual **~MessageDispatchChannel** ()
- virtual void **enqueue** (const **Pointer< MessageDispatch >** &message)=0
Add a Message to the Channel behind all pending message.
- virtual void **enqueueFirst** (const **Pointer< MessageDispatch >** &message)=0
Add a message to the front of the Channel.
- virtual bool **isEmpty** () const =0
- virtual bool **isClosed** () const =0
- virtual bool **isRunning** () const =0
- virtual **Pointer< MessageDispatch >** **dequeue** (long long timeout)=0
Used to get an enqueued message.
- virtual **Pointer< MessageDispatch >** **dequeueNowait** ()=0
Used to get an enqueued message if there is one queued right now.
- virtual **Pointer< MessageDispatch >** **peek** () const =0
Peek in the Queue and return the first message in the Channel without removing it from the channel.
- virtual void **start** ()=0
Starts dispatch of messages from the Channel.
- virtual void **stop** ()=0
Stops dispatch of message from the Channel.
- virtual void **close** ()=0
Close this channel no messages will be dispatched after this method is called.
- virtual void **clear** ()=0

Clear the Channel, all pending messages are removed.

- virtual int **size** () const =0
- virtual std::vector< **Pointer** < **MessageDispatch** > > **removeAll** ()=0

Remove all messages that are currently in the Channel and return them as a list of Messages.

6.347.1 Constructor & Destructor Documentation

6.347.1.1 virtual **activemq::core::MessageDispatchChannel**
::~MessageDispatchChannel () [inline,
 virtual]

6.347.2 Member Function Documentation

6.347.2.1 virtual void **activemq::core::MessageDispatchChannel::clear** () [pure
 virtual]

Clear the Channel, all pending messages are removed.

Implemented in **activemq::core::SimplePriorityMessageDispatchChannel** (p. 2445),
 and **activemq::core::FifoMessageDispatchChannel** (p. 1324).

6.347.2.2 virtual void **activemq::core::MessageDispatchChannel::close** () [pure
 virtual]

Close this channel no messages will be dispatched after this method is called.

Implemented in **activemq::core::SimplePriorityMessageDispatchChannel** (p. 2446),
 and **activemq::core::FifoMessageDispatchChannel** (p. 1324).

6.347.2.3 virtual **Pointer**<**MessageDispatch**> **activemq::core::Message-**
DispatchChannel::dequeue (long long *timeout*) [pure
 virtual]

Used to get an enqueued message.

The amount of time this method blocks is based on the timeout value. - if timeout==1 then it blocks until a message is received. - if timeout==0 then it tries to not block at all, it returns a message if it is available - if timeout>0 then it blocks up to timeout amount of time. Expired messages will be consumed by this method.

Returns

null if we timeout or if the consumer is closed.

Exceptions

<i>ActiveMQException</i>	
--------------------------	--

Implemented in **activemq::core::SimplePriorityMessageDispatchChannel** (p. 2446),
and **activemq::core::FifoMessageDispatchChannel** (p. 1324).

6.347.2.4 `virtual Pointer<MessageDispatch> activemq::core::MessageDispatchChannel::dequeueNoWait () [pure virtual]`

Used to get an enqueued message if there is one queued right now.

If there is no waiting message than this method returns Null.

Returns

a message if there is one in the queue.

Implemented in **activemq::core::SimplePriorityMessageDispatchChannel** (p. 2446),
and **activemq::core::FifoMessageDispatchChannel** (p. 1325).

6.347.2.5 `virtual void activemq::core::MessageDispatchChannel::enqueue (const Pointer< MessageDispatch > & message) [pure virtual]`

Add a Message to the Channel behind all pending message.

Parameters

<i>message</i>	- The message to add to the Channel.
----------------	--------------------------------------

Implemented in **activemq::core::SimplePriorityMessageDispatchChannel** (p. 2446),
and **activemq::core::FifoMessageDispatchChannel** (p. 1325).

6.347.2.6 `virtual void activemq::core::MessageDispatchChannel::enqueueFirst (const Pointer< MessageDispatch > & message) [pure virtual]`

Add a message to the front of the Channel.

Parameters

<i>message</i>	- The Message to add to the front of the Channel.
----------------	---

Implemented in **activemq::core::SimplePriorityMessageDispatchChannel** (p. 2447),
and **activemq::core::FifoMessageDispatchChannel** (p. 1325).

6.347.2.7 `virtual bool activemq::core::MessageDispatchChannel::isClosed () const [pure virtual]`

Returns

has the Queue been closed.

Implemented in **activemq::core::SimplePriorityMessageDispatchChannel** (p. 2447),
and **activemq::core::FifoMessageDispatchChannel** (p. 1325).

6.347.2.8 `virtual bool activemq::core::MessageDispatchChannel::isEmpty () const`
`[pure virtual]`

Returns

true if there are no messages in the Channel.

Implemented in **activemq::core::SimplePriorityMessageDispatchChannel** (p. 2447),
and **activemq::core::FifoMessageDispatchChannel** (p. 1326).

6.347.2.9 `virtual bool activemq::core::MessageDispatchChannel::isRunning ()`
`const [pure virtual]`

Returns

true if the Channel currently running and will dequeue message.

Implemented in **activemq::core::SimplePriorityMessageDispatchChannel** (p. 2447),
and **activemq::core::FifoMessageDispatchChannel** (p. 1326).

6.347.2.10 `virtual Pointer<MessageDispatch> activemq::core-`
`::MessageDispatchChannel::peek () const [pure`
`virtual]`

Peek in the Queue and return the first message in the Channel without removing it from the channel.

Returns

a message if there is one in the queue.

Implemented in **activemq::core::SimplePriorityMessageDispatchChannel** (p. 2449),
and **activemq::core::FifoMessageDispatchChannel** (p. 1327).

6.347.2.11 `virtual std::vector< Pointer<MessageDispatch> >`
`activemq::core::MessageDispatchChannel::removeAll () [pure`
`virtual]`

Remove all messages that are currently in the Channel and return them as a list of Messages.

Returns

a list of Messages that was previously in the Channel.

Implemented in **activemq::core::SimplePriorityMessageDispatchChannel** (p. 2449),
and **activemq::core::FifoMessageDispatchChannel** (p. 1327).

6.347.2.12 `virtual int activemq::core::MessageDispatchChannel::size () const`
[pure virtual]

Returns

the number of Messages currently in the Channel.

Implemented in **activemq::core::SimplePriorityMessageDispatchChannel** (p. 2449),
and **activemq::core::FifoMessageDispatchChannel** (p. 1327).

6.347.2.13 `virtual void activemq::core::MessageDispatchChannel::start ()`
[pure virtual]

Starts dispatch of messages from the Channel.

Implemented in **activemq::core::SimplePriorityMessageDispatchChannel** (p. 2449),
and **activemq::core::FifoMessageDispatchChannel** (p. 1328).

6.347.2.14 `virtual void activemq::core::MessageDispatchChannel::stop ()` [pure
virtual]

Stops dispatch of message from the Channel.

Implemented in **activemq::core::SimplePriorityMessageDispatchChannel** (p. 2449),
and **activemq::core::FifoMessageDispatchChannel** (p. 1328).

The documentation for this class was generated from the following file:

- `src/main/activemq/core/MessageDispatchChannel.h`

6.348 **activemq::wireformat::openwire::marshal::generated::-** **MessageDispatchMarshaller Class Reference**

Marshaling code for Open Wire Format for **MessageDispatchMarshaller** (p. 1890).

```
#include <src/main/activemq/wireformat/openwire/marshal/generated/-  
MessageDispatchMarshaller.h>
```

Inheritance diagram for `activemq::wireformat::openwire::marshal::generated::Message-`
`DispatchMarshaller`:

Public Member Functions

- **MessageDispatchMarshaller** ()
- virtual **~MessageDispatchMarshaller** ()
- virtual **commands::DataStructure * createObject** () const
Creates a new instance of the class that this class is a marshaling director for.
- virtual unsigned char **getDataStructureType** () const
Gets the DataStructureType that this class marshals/unmarshals.
- virtual void **tightUnmarshal** (**OpenWireFormat *wireFormat**, **commands::DataStructure *dataStructure**, **decaf::io::DataInputStream *dataIn**, **utils::BooleanStream *bs**)
Tight Un-marhsal to the given stream.
- virtual int **tightMarshal1** (**OpenWireFormat *wireFormat**, **commands::DataStructure *dataStructure**, **utils::BooleanStream *bs**)
Tight Marshal to the given stream.
- virtual void **tightMarshal2** (**OpenWireFormat *wireFormat**, **commands::DataStructure *dataStructure**, **decaf::io::DataOutputStream *dataOut**, **utils::BooleanStream *bs**)
Tight Marshal to the given stream.
- virtual void **looseUnmarshal** (**OpenWireFormat *wireFormat**, **commands::DataStructure *dataStructure**, **decaf::io::DataInputStream *dataIn**)
Loose Un-marhsal to the given stream.
- virtual void **looseMarshal** (**OpenWireFormat *wireFormat**, **commands::DataStructure *dataStructure**, **decaf::io::DataOutputStream *dataOut**)
Tight Marshal to the given stream.

6.348.1 Detailed Description

Marshaling code for Open Wire Format for **MessageDispatchMarshaller** (p. 1890).

NOTE!: This file is auto generated - do not modify! if you need to make a change, please see the Java Classes in the activemq-openwire-generator module

6.348.2 Constructor & Destructor Documentation

6.348.2.1 **activemq::wireformat::openwire::marshal::generated::MessageDispatchMarshaller::MessageDispatchMarshaller** ()
[inline]

6.348.2.2 **virtual activemq::wireformat::openwire::marshal::generated::MessageDispatchMarshaller::~~MessageDispatchMarshaller** () [inline, virtual]

6.348.3 Member Function Documentation

6.348.3.1 `virtual commands::DataStructure* activemq::wireformat::openwire-
::marshal::generated::MessageDispatchMarshaller::createObject ()
const [virtual]`

Creates a new instance of the class that this class is a marshaling director for.

Returns

newly allocated Command

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller**
(p. 1120).

6.348.3.2 `virtual unsigned char activemq::wireformat::openwire::marshal::generated-
::MessageDispatchMarshaller::getDataStructureType () const
[virtual]`

Gets the DataStructureType that this class marshals/unmarshals.

Returns

byte Id of this classes DataStructureType

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller**
(p. 1122).

6.348.3.3 `virtual void activemq::wireformat::openwire::marshal::generated::-
MessageDispatchMarshaller::looseMarshal (OpenWireFormat * format,
commands::DataStructure * command, decaf::io::DataOutputStream * ds
) [virtual]`

Tight Marshal to the given stream.

Parameters

<i>format</i>	- The OpenWireFormat properties
<i>command</i>	- the object to Marshal
<i>ds</i>	- DataOutputStream to marshal to

Exceptions

<i>IOException</i>	if an error occurs.
--------------------	---------------------

Reimplemented from **activemq::wireformat::openwire::marshal::generated::Base-
CommandMarshaller** (p. 500).

6.348 activemq::wireformat::openwire::marshal::generated::MessageDispatch-Marshaller Class

Reference

1899

6.348.3.4 `virtual void activemq::wireformat::openwire::marshal-
::generated::MessageDispatchMarshaller::looseUnmarshal (
OpenWireFormat * format, commands::DataStructure * command,
decaf::io::DataInputStream * dis) [virtual]`

Loose Un-marhsal to the given stream.

Parameters

<i>format</i>	- The OpenwireFormat properties
<i>command</i>	- the object to Un-Marshal
<i>dis</i>	- the DataInputStream to Un-Marshal from

Exceptions

<i>IOException</i>	if an error occurs.
--------------------	---------------------

Reimplemented from **activemq::wireformat::openwire::marshal::generated::Base-CommandMarshaller** (p. 501).

6.348.3.5 `virtual int activemq::wireformat::openwire::marshal::generated::-
MessageDispatchMarshaller::tightMarshal1 (OpenWireFormat * format,
commands::DataStructure * command, utils::BooleanStream * bs)
[virtual]`

Tight Marhsal to the given stream.

Parameters

<i>format</i>	The OpenwireFormat properties
<i>command</i>	The object to Marshal
<i>bs</i>	The boolean stream to marshal to.

Exceptions

<i>IOException</i>	if an error occurs.
--------------------	---------------------

Reimplemented from **activemq::wireformat::openwire::marshal::generated::Base-CommandMarshaller** (p. 503).

6.348.3.6 `virtual void activemq::wireformat::openwire::marshal::generated::-
MessageDispatchMarshaller::tightMarshal2 (OpenWireFormat * format,
commands::DataStructure * command, decaf::io::DataOutputStream *
ds, utils::BooleanStream * bs) [virtual]`

Tight Marhsal to the given stream.

Parameters

<i>format</i>	- The OpenwireFormat properties
<i>command</i>	- the object to Marshal
<i>ds</i>	- the DataOutputStream to Marshal to
<i>bs</i>	- boolean stream to marshal to.

Exceptions

<i>IOException</i>	if an error occurs.
--------------------	---------------------

Reimplemented from **activemq::wireformat::openwire::marshal::generated::Base-CommandMarshaller** (p. 504).

```
6.348.3.7 virtual void activemq::wireformat::openwire::marshal-
::generated::MessageDispatchMarshaller::tightUnmarshal (
OpenWireFormat * format, commands::DataStructure * command,
decaf::io::DataInputStream * dis, utils::BooleanStream * bs )
[virtual]
```

Tight Un-marhsal to the given stream.

Parameters

<i>format</i>	- The OpenwireFormat properties
<i>command</i>	- the object to Un-Marshal
<i>dis</i>	- the DataInputStream to Un-Marshal from
<i>bs</i>	- boolean stream to unmarshal from.

Exceptions

<i>IOException</i>	if an error occurs.
--------------------	---------------------

Reimplemented from **activemq::wireformat::openwire::marshal::generated::Base-CommandMarshaller** (p. 505).

The documentation for this class was generated from the following file:

- src/main/activemq/wireformat/openwire/marshal/generated/**MessageDispatch-Marshaller.h**

6.349 activemq::commands::MessageDispatchNotification Class - Reference

```
#include <src/main/activemq/commands/MessageDispatch-
Notification.h>
```

6.349 activemq::commands::MessageDispatchNotification Class Reference 1901

Inheritance diagram for activemq::commands::MessageDispatchNotification:

Public Member Functions

- **MessageDispatchNotification** ()
- virtual **~MessageDispatchNotification** ()
- virtual unsigned char **getDataStructureType** () const
*Get the **DataStructure** (p. 1133) Type as defined in CommandTypes.h.*
- virtual **MessageDispatchNotification * cloneDataStructure** () const
Clone this obbect and return a new instance that the caller now owns, this will be an exact copy of this one.
- virtual void **copyDataStructure** (const **DataStructure** *src)
Copy the contents of the passed object into this objects members, overwriting any existing data.
- virtual std::string **toString** () const
*Returns a string containing the information for this **DataStructure** (p. 1133) such as its type and value of its elements.*
- virtual bool **equals** (const **DataStructure** *value) const
*Compares the **DataStructure** (p. 1133) passed in to this one, and returns if they are equivalent.*
- virtual const **Pointer** < **ConsumerId** > & **getConsumerId** () const
- virtual **Pointer**< **ConsumerId** > & **getConsumerId** ()
- virtual void **setConsumerId** (const **Pointer**< **ConsumerId** > &consumerId)
- virtual const **Pointer** < **ActiveMQDestination** > & **getDestination** () const
- virtual **Pointer** < **ActiveMQDestination** > & **getDestination** ()
- virtual void **setDestination** (const **Pointer**< **ActiveMQDestination** > &destination)
- virtual long long **getDeliverySequenceId** () const
- virtual void **setDeliverySequenceId** (long long deliverySequenceId)
- virtual const **Pointer** < **MessageId** > & **getMessageId** () const
- virtual **Pointer**< **MessageId** > & **getMessageId** ()
- virtual void **setMessageId** (const **Pointer**< **MessageId** > &messageId)
- virtual bool **isMessageDispatchNotification** () const
- virtual **Pointer**< **Command** > **visit** (activemq::state::CommandVisitor *visitor)
Allows a Visitor to visit this command and return a response to the command based on the command type being visited.

Static Public Attributes

- static const unsigned char **ID_MESSAGEDISPATCHNOTIFICATION** = 90

Protected Attributes

- **Pointer**< **ConsumerId** > **consumerId**
- **Pointer**< **ActiveMQDestination** > **destination**
- long long **deliverySequenceId**
- **Pointer**< **MessageId** > **messageId**

6.349.1 Constructor & Destructor Documentation

6.349.1.1 **activemq::commands::MessageDispatchNotification::MessageDispatchNotification ()**

6.349.1.2 **virtual activemq::commands::MessageDispatchNotification::~~MessageDispatchNotification ()**
[virtual]

6.349.2 Member Function Documentation

6.349.2.1 **virtual MessageDispatchNotification* activemq::commands::MessageDispatchNotification::cloneDataStructure () const**
[virtual]

Clone this object and return a new instance that the caller now owns, this will be an exact copy of this one.

Returns

new copy of this object.

Implements **activemq::commands::DataStructure** (p. 1133).

6.349.2.2 **virtual void activemq::commands::MessageDispatchNotification::copyDataStructure (const DataStructure * src)**
[virtual]

Copy the contents of the passed object into this objects members, overwriting any existing data.

Returns

src - Source Object

Reimplemented from **activemq::commands::BaseCommand** (p. 493).

6.349.2.3 **virtual bool activemq::commands::MessageDispatchNotification::equals (const DataStructure * value) const** [virtual]

Compares the **DataStructure** (p. 1133) passed in to this one, and returns if they are equivalent.

6.349 activemq::commands::MessageDispatchNotification Class Reference 1903

Equivalent here means that they are of the same type, and that each element of the objects are the same.

Returns

true if **DataSet** (p. 1133)'s are Equal.

Reimplemented from **activemq::commands::BaseCommand** (p. 494).

6.349.2.4 virtual const **Pointer**<**ConsumerId**>& **activemq::commands-
::MessageDispatchNotification::getConsumerId** () const
[virtual]

6.349.2.5 virtual **Pointer**<**ConsumerId**>& **activemq::commands-
::MessageDispatchNotification::getConsumerId** ()
[virtual]

6.349.2.6 virtual unsigned char **activemq::commands::Message-
DispatchNotification::getDataSetType** () const
[virtual]

Get the **DataSet** (p. 1133) Type as defined in CommandTypes.h.

Returns

The type of the data structure

Implements **activemq::commands::DataSet** (p. 1137).

6.349.2.7 virtual long long **activemq::commands::Message-
DispatchNotification::getDeliverySequenceId** () const
[virtual]

6.349.2.8 virtual const **Pointer**<**ActiveMQDestination**>& **activemq::commands-
::MessageDispatchNotification::getDestination** () const
[virtual]

6.349.2.9 virtual **Pointer**<**ActiveMQDestination**>& **activemq::commands-
::MessageDispatchNotification::getDestination** ()
[virtual]

6.349.2.10 virtual const **Pointer**<**MessageId**>& **activemq::commands-
::MessageDispatchNotification::getMessageId** () const
[virtual]

6.349.2.11 virtual **Pointer**<**MessageId**>& **activemq::commands-
::MessageDispatchNotification::getMessageId** ()
[virtual]

6.349.2.12 `virtual bool activemq::commands::MessageDispatchNotification-
::isMessageDispatchNotification () const [inline,
virtual]`

Returns

an answer of true to the **isMessageDispatchNotification()** (p. 1898) query.

Reimplemented from **activemq::commands::BaseCommand** (p. 496).

6.349.2.13 `virtual void activemq::commands::MessageDispatchNotification-
::setConsumerId (const Pointer< ConsumerId > & consumerId)
[virtual]`

6.349.2.14 `virtual void activemq::commands::MessageDispatchNotification-
::setDeliverySequenceId (long long deliverySequenceId)
[virtual]`

6.349.2.15 `virtual void activemq::commands::MessageDispatchNotification::set-
Destination (const Pointer< ActiveMQDestination > & destination)
[virtual]`

6.349.2.16 `virtual void activemq::commands::MessageDispatchNotification-
::setMessageId (const Pointer< MessageId > & messageId)
[virtual]`

6.349.2.17 `virtual std::string activemq::commands::MessageDispatchNotification-
::toString () const [virtual]`

Returns a string containing the information for this **DataStructure** (p. 1133) such as its type and value of its elements.

Returns

formatted string useful for debugging.

Reimplemented from **activemq::commands::BaseCommand** (p. 498).

6.349.2.18 `virtual Pointer<Command> activemq::commands::MessageDispatch-
Notification::visit (activemq::state::CommandVisitor * visitor)
[virtual]`

Allows a Visitor to visit this command and return a response to the command based on the command type being visited.

The command will call the proper processXXX method in the visitor.

6.350 activemq::wireformat::openwire::marshal::generated::MessageDispatchNotificationMarshaller Class Reference

1905

Returns a **Response** (p. 2298) to the visitor being called or NULL if no response.

Implements **activemq::commands::Command** (p. 871).

6.349.3 Field Documentation

- 6.349.3.1 **Pointer<ConsumerId> activemq::commands::MessageDispatchNotification::consumerId** [protected]
- 6.349.3.2 **long long activemq::commands::MessageDispatchNotification::deliverySequenceld** [protected]
- 6.349.3.3 **Pointer<ActiveMQDestination> activemq::commands::MessageDispatchNotification::destination** [protected]
- 6.349.3.4 **const unsigned char activemq::commands::MessageDispatchNotification::ID_MESSAGEDISPATCHNOTIFICATION = 90** [static]
- 6.349.3.5 **Pointer<MessageId> activemq::commands::MessageDispatchNotification::messageId** [protected]

The documentation for this class was generated from the following file:

- src/main/activemq/commands/**MessageDispatchNotification.h**

6.350 activemq::wireformat::openwire::marshal::generated::MessageDispatchNotificationMarshaller Class Reference

Marshaling code for Open Wire Format for **MessageDispatchNotificationMarshaller** (p. 1899).

```
#include <src/main/activemq/wireformat/openwire/marshal/generated/-  
MessageDispatchNotificationMarshaller.h>
```

Inheritance diagram for **activemq::wireformat::openwire::marshal::generated::MessageDispatchNotificationMarshaller**:

Public Member Functions

- **MessageDispatchNotificationMarshaller ()**
- **virtual ~MessageDispatchNotificationMarshaller ()**
- **virtual commands::DataStructure * createObject () const**

Creates a new instance of the class that this class is a marshaling director for.

- virtual unsigned char **getDataStructureType** () const

Gets the DataStructureType that this class marshals/unmarshals.

- virtual void **tightUnmarshal** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataInputStream** *dataIn, **utils::BooleanStream** *bs)

Tight Un-marhsal to the given stream.

- virtual int **tightMarshal1** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **utils::BooleanStream** *bs)

Tight Marshal to the given stream.

- virtual void **tightMarshal2** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataOutputStream** *dataOut, **utils::BooleanStream** *bs)

Tight Marshal to the given stream.

- virtual void **looseUnmarshal** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataInputStream** *dataIn)

Loose Un-marhsal to the given stream.

- virtual void **looseMarshal** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataOutputStream** *dataOut)

Tight Marshal to the given stream.

6.350.1 Detailed Description

Marshaling code for Open Wire Format for **MessageDispatchNotificationMarshaller** (p. 1899).

NOTE!: This file is auto generated - do not modify! if you need to make a change, please see the Java Classes in the activemq-openwire-generator module

6.350.2 Constructor & Destructor Documentation

6.350.2.1 **activemq::wireformat::openwire::marshal::generated::MessageDispatchNotificationMarshaller::MessageDispatchNotificationMarshaller** () [inline]

6.350.2.2 **virtual activemq::wireformat::openwire::marshal::generated::MessageDispatchNotificationMarshaller::~MessageDispatchNotificationMarshaller** () [inline, virtual]

6.350.3 Member Function Documentation

6.350 activemq::wireformat::openwire::marshal::generated::MessageDispatchNotificationMarshaller Class

Reference

1907

```
6.350.3.1 virtual commands::DataStructure* activemq::wireformat-  
::openwire::marshal::generated::MessageDispatch-  
NotificationMarshaller::createObject ( ) const  
[virtual]
```

Creates a new instance of the class that this class is a marshaling director for.

Returns

newly allocated Command

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1120).

```
6.350.3.2 virtual unsigned char activemq::wireformat::openwire::marshal::generated-  
::MessageDispatchNotificationMarshaller::getDataStructureType ( )  
const [virtual]
```

Gets the DataStructureType that this class marshals/unmarshals.

Returns

byte Id of this classes DataStructureType

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1122).

```
6.350.3.3 virtual void activemq::wireformat::openwire::marshal::generated-  
::MessageDispatchNotificationMarshaller::looseMarshal (  
OpenWireFormat * format, commands::DataStructure * command,  
decaf::io::DataOutputStream * ds ) [virtual]
```

Tight Marshal to the given stream.

Parameters

<i>format</i>	- The OpenWireFormat properties
<i>command</i>	- the object to Marshal
<i>ds</i>	- DataOutputStream to marshal to

Exceptions

<i>IOException</i>	if an error occurs.
--------------------	---------------------

Reimplemented from **activemq::wireformat::openwire::marshal::generated::BaseCommandMarshaller** (p. 500).

6.350.3.4 `virtual void activemq::wireformat::openwire::marshal::generated-
::MessageDispatchNotificationMarshaller::looseUnmarshal (
OpenWireFormat * format, commands::DataStructure * command,
decaf::io::DataInputStream * dis) [virtual]`

Loose Un-marhsal to the given stream.

Parameters

<i>format</i>	- The OpenWireFormat properties
<i>command</i>	- the object to Un-Marshal
<i>dis</i>	- the DataInputStream to Un-Marshal from

Exceptions

<i>IOException</i>	if an error occurs.
--------------------	---------------------

Reimplemented from **activemq::wireformat::openwire::marshal::generated::Base-
CommandMarshaller** (p. 501).

6.350.3.5 `virtual int activemq::wireformat::openwire::marshal::generated-
::MessageDispatchNotificationMarshaller::tightMarshal1 (
OpenWireFormat * format, commands::DataStructure * command,
utils::BooleanStream * bs) [virtual]`

Tight Marhsal to the given stream.

Parameters

<i>format</i>	The OpenWireFormat properties
<i>command</i>	The object to Marshal
<i>bs</i>	The boolean stream to marshal to.

Exceptions

<i>IOException</i>	if an error occurs.
--------------------	---------------------

Reimplemented from **activemq::wireformat::openwire::marshal::generated::Base-
CommandMarshaller** (p. 503).

6.350.3.6 `virtual void activemq::wireformat::openwire::marshal::generated-
::MessageDispatchNotificationMarshaller::tightMarshal2 (
OpenWireFormat * format, commands::DataStructure * command,
decaf::io::DataOutputStream * ds, utils::BooleanStream * bs)
[virtual]`

Tight Marhsal to the given stream.

Parameters

<i>format</i>	- The OpenwireFormat properties
<i>command</i>	- the object to Marshal
<i>ds</i>	- the DataOutputStream to Marshal to
<i>bs</i>	- boolean stream to marshal to.

Exceptions

<i>IOException</i>	if an error occurs.
--------------------	---------------------

Reimplemented from **activemq::wireformat::openwire::marshal::generated::Base-CommandMarshaller** (p. 504).

```
6.350.3.7 virtual void activemq::wireformat::openwire::marshal::generated-
::MessageDispatchNotificationMarshaller::tightUnmarshal (
    OpenWireFormat * format, commands::DataStructure * command,
    decaf::io::DataInputStream * dis, utils::BooleanStream * bs )
    [virtual]
```

Tight Un-marhsal to the given stream.

Parameters

<i>format</i>	- The OpenwireFormat properties
<i>command</i>	- the object to Un-Marshal
<i>dis</i>	- the DataInputStream to Un-Marshal from
<i>bs</i>	- boolean stream to unmarshal from.

Exceptions

<i>IOException</i>	if an error occurs.
--------------------	---------------------

Reimplemented from **activemq::wireformat::openwire::marshal::generated::Base-CommandMarshaller** (p. 505).

The documentation for this class was generated from the following file:

- src/main/activemq/wireformat/openwire/marshal/generated/**MessageDispatch-NotificationMarshaller.h**

6.351 cms::MessageEnumeration Class Reference

Defines an object that enumerates a collection of Messages.

```
#include <src/main/cms/MessageEnumeration.h>
```

Inheritance diagram for cms::MessageEnumeration:

Public Member Functions

- virtual `~MessageEnumeration ()` throw ()
- virtual bool `hasMoreMessages ()`=0
*Returns true if there are more **Message** (p. 1839) in the Browser that can be retrieved via the `nextMessage` method.*
- virtual `cms::Message * nextMessage ()`=0
*Returns the Next **Message** (p. 1839) in the **Queue** (p. 2221) if one is present, if no more **Message** (p. 1839)'s are available then an Exception is thrown.*

6.351.1 Detailed Description

Defines an object that enumerates a collection of Messages.

The client calls the `hasMoreMessages` method to determine if a **Message** (p. 1839) is available. If a **Message** (p. 1839) is available the client calls the `nextMessage` method to retrieve that **Message** (p. 1839), calling `nextMessage` when a **Message** (p. 1839) is not available results in an exception.

Since

2.1

6.351.2 Constructor & Destructor Documentation

6.351.2.1 virtual `cms::MessageEnumeration::~MessageEnumeration ()` throw ()
 [virtual]

6.351.3 Member Function Documentation

6.351.3.1 virtual bool `cms::MessageEnumeration::hasMoreMessages ()` [pure virtual]

Returns true if there are more **Message** (p. 1839) in the Browser that can be retrieved via the `nextMessage` method.

If this method returns false and the `nextMessage` method is called then an Exception will be thrown.

Returns

true if more **Message** (p. 1839)'s are available in the Browser.

Implemented in `activemq::core::ActiveMQQueueBrowser` (p. 328).

6.351.3.2 virtual cms::Message* cms::MessageEnumeration::nextMessage ()
[pure virtual]

Returns the Next **Message** (p. 1839) in the **Queue** (p. 2221) if one is present, if no more **Message** (p. 1839)'s are available then an Exception is thrown.

If a **Message** (p. 1839) object pointer is returned then that object becomes the property of the caller and must be deleted by the caller when finished.

Returns

The next **Message** (p. 1839) in the **Queue** (p. 2221).

Exceptions

CMSException (p. 826)	if no more Message (p. 1839)'s currently in the Queue (p. 2221).
---------------------------------	--

Implemented in **activemq::core::ActiveMQQueueBrowser** (p. 328).

The documentation for this class was generated from the following file:

- src/main/cms/**MessageEnumeration.h**

6.352 cms::MessageEOFException Class Reference

This exception must be thrown when an unexpected end of stream has been reached when a **StreamMessage** (p. 2606) or **BytesMessage** (p. 718) is being read.

```
#include <src/main/cms/MessageEOFException.h>
```

Inheritance diagram for cms::MessageEOFException:

Public Member Functions

- **MessageEOFException** ()
- **MessageEOFException** (const **MessageEOFException** &ex)
- **MessageEOFException** (const std::string &message)
- **MessageEOFException** (const std::string &message, const std::exception *cause)
- **MessageEOFException** (const std::string &message, const std::exception *cause, const std::vector< std::pair< std::string, int > > &stackTrace)
- virtual ~**MessageEOFException** () throw ()

6.352.1 Detailed Description

This exception must be thrown when an unexpected end of stream has been reached when a **StreamMessage** (p. 2606) or **BytesMessage** (p. 718) is being read.

Since

1.3

6.352.2 Constructor & Destructor Documentation

6.352.2.1 `cms::MessageEOFException::MessageEOFException ()`

6.352.2.2 `cms::MessageEOFException::MessageEOFException (const MessageEOFException & ex)`

6.352.2.3 `cms::MessageEOFException::MessageEOFException (const std::string & message)`

6.352.2.4 `cms::MessageEOFException::MessageEOFException (const std::string & message, const std::exception * cause)`

6.352.2.5 `cms::MessageEOFException::MessageEOFException (const std::string & message, const std::exception * cause, const std::vector< std::pair< std::string, int > > & stackTrace)`

6.352.2.6 `virtual cms::MessageEOFException::~~MessageEOFException () throw ()`
[virtual]

The documentation for this class was generated from the following file:

- `src/main/cms/MessageEOFException.h`

6.353 cms::MessageFormatException Class Reference

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

```
#include <src/main/cms/MessageFormatException.h>
```

Inheritance diagram for cms::MessageFormatException:

Public Member Functions

- `MessageFormatException ()`

- **MessageFormatException** (const **MessageFormatException** &ex)
- **MessageFormatException** (const std::string &message)
- **MessageFormatException** (const std::string &message, const std::exception *cause)
- **MessageFormatException** (const std::string &message, const std::exception *cause, const std::vector< std::pair< std::string, int > > &stackTrace)
- virtual ~**MessageFormatException** () throw ()

6.353.1 Detailed Description

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

It must also be thrown when equivalent type errors are made with message property values. For example, this exception must be thrown if **StreamMessage.readShort** (p.2614) is used to read a boolean value.

Since

1.3

6.353.2 Constructor & Destructor Documentation

- 6.353.2.1 **cms::MessageFormatException::MessageFormatException** ()
- 6.353.2.2 **cms::MessageFormatException::MessageFormatException** (const **MessageFormatException** & ex)
- 6.353.2.3 **cms::MessageFormatException::MessageFormatException** (const std::string & message)
- 6.353.2.4 **cms::MessageFormatException::MessageFormatException** (const std::string & message, const std::exception * cause)
- 6.353.2.5 **cms::MessageFormatException::MessageFormatException** (const std::string & message, const std::exception * cause, const std::vector< std::pair< std::string, int > > & stackTrace)
- 6.353.2.6 virtual **cms::MessageFormatException::~MessageFormatException** () throw () [virtual]

The documentation for this class was generated from the following file:

- src/main/cms/**MessageFormatException.h**

6.354 activemq::commands::MessageId Class Reference

```
#include <src/main/activemq/commands/MessageId.h>
```

Inheritance diagram for activemq::commands::MessageId:

Public Types

- typedef **decaf::lang::PointerComparator** < **MessageId** > **COMPARATOR**

Public Member Functions

- **MessageId** ()
- **MessageId** (const **MessageId** &other)
- **MessageId** (const std::string &messageKey)
- **MessageId** (const **Pointer**< **ProducerInfo** > &producerInfo, long long **producerSequenceld**)
- **MessageId** (const **Pointer**< **ProducerId** > &producerId, long long **producerSequenceld**)
- **MessageId** (const std::string &producerId, long long **producerSequenceld**)
- virtual ~**MessageId** ()
- virtual unsigned char **getDataStructureType** () const
*Get the **DataStructure** (p. 1133) Type as defined in CommandTypes.h.*
- virtual **MessageId** * **cloneDataStructure** () const
Clone this object and return a new instance that the caller now owns, this will be an exact copy of this one.
- virtual void **copyDataStructure** (const **DataStructure** *src)
Copy the contents of the passed object into this objects members, overwriting any existing data.
- virtual std::string **toString** () const
*Returns a string containing the information for this **DataStructure** (p. 1133) such as its type and value of its elements.*
- virtual bool **equals** (const **DataStructure** *value) const
*Compares the **DataStructure** (p. 1133) passed in to this one, and returns if they are equivalent.*
- void **setValue** (const std::string &key)
- void **setTextView** (const std::string &key)
- virtual const **Pointer** < **ProducerId** > & **getProducerId** () const
- virtual **Pointer**< **ProducerId** > & **getProducerId** ()
- virtual void **setProducerId** (const **Pointer**< **ProducerId** > &producerId)
- virtual long long **getProducerSequenceld** () const
- virtual void **setProducerSequenceld** (long long **producerSequenceld**)
- virtual long long **getBrokerSequenceld** () const
- virtual void **setBrokerSequenceld** (long long **brokerSequenceld**)

- virtual int **compareTo** (const **MessageId** &value) const
- virtual bool **equals** (const **MessageId** &value) const
- virtual bool **operator==** (const **MessageId** &value) const
- virtual bool **operator<** (const **MessageId** &value) const
- **MessageId** & **operator=** (const **MessageId** &other)

Static Public Attributes

- static const unsigned char **ID_MESSAGEID** = 110

Protected Attributes

- **Pointer< ProducerId > producerId**
- long long **producerSequenceId**
- long long **brokerSequenceId**

6.354.1 Member Typedef Documentation

6.354.1.1 `typedef decaf::lang::PointerComparator<MessageId>
activemq::commands::MessageId::COMPARATOR`

6.354.2 Constructor & Destructor Documentation

6.354.2.1 `activemq::commands::MessageId::MessageId ()`

6.354.2.2 `activemq::commands::MessageId::MessageId (const MessageId & other
)`

6.354.2.3 `activemq::commands::MessageId::MessageId (const std::string &
messageKey)`

6.354.2.4 `activemq::commands::MessageId::MessageId (const Pointer<
ProducerInfo > & producerInfo, long long producerSequenceId)`

6.354.2.5 `activemq::commands::MessageId::MessageId (const Pointer<
ProducerId > & producerId, long long producerSequenceId)`

6.354.2.6 `activemq::commands::MessageId::MessageId (const std::string &
producerId, long long producerSequenceId)`

6.354.2.7 `virtual activemq::commands::MessageId::~~MessageId () [virtual]`

6.354.3 Member Function Documentation

6.354.3.1 `virtual MessageId* activemq::commands::MessageId::cloneDataStructure () const [virtual]`

Clone this object and return a new instance that the caller now owns, this will be an exact copy of this one.

Returns

new copy of this object.

Implements **activemq::commands::DataStructure** (p. 1133).

6.354.3.2 `virtual int activemq::commands::MessageId::compareTo (const MessageId & value) const [virtual]`

6.354.3.3 `virtual void activemq::commands::MessageId::copyDataStructure (const DataStructure * src) [virtual]`

Copy the contents of the passed object into this objects members, overwriting any existing data.

Returns

src - Source Object

Implements **activemq::commands::DataStructure** (p. 1134).

6.354.3.4 `virtual bool activemq::commands::MessageId::equals (const DataStructure * value) const [virtual]`

Compares the **DataStructure** (p. 1133) passed in to this one, and returns if they are equivalent.

Equivalent here means that they are of the same type, and that each element of the objects are the same.

Returns

true if **DataStructure** (p. 1133)'s are Equal.

Implements **activemq::commands::DataStructure** (p. 1135).

6.354.3.5 `virtual bool activemq::commands::MessageId::equals (const MessageId & value) const [virtual]`

6.354.3.6 `virtual long long activemq::commands::MessageId::getBrokerSequenceId () const [virtual]`

6.354.3.7 virtual unsigned char **activemq::commands::MessageId::getDataStructureType**() const [virtual]

Get the **DataStructure** (p. 1133) Type as defined in CommandTypes.h.

Returns

The type of the data structure

Implements **activemq::commands::DataStructure** (p. 1137).

6.354.3.8 virtual const **Pointer**<**ProducerId**>& **activemq::commands::MessageId::getProducerId**() const [virtual]

6.354.3.9 virtual **Pointer**<**ProducerId**>& **activemq::commands::MessageId::getProducerId**() [virtual]

6.354.3.10 virtual long long **activemq::commands::MessageId::getProducerSequenceId**() const [virtual]

6.354.3.11 virtual bool **activemq::commands::MessageId::operator**<(const **MessageId** & *value*) const [virtual]

6.354.3.12 **MessageId**& **activemq::commands::MessageId::operator**=(const **MessageId** & *other*)

6.354.3.13 virtual bool **activemq::commands::MessageId::operator**==(const **MessageId** & *value*) const [virtual]

6.354.3.14 virtual void **activemq::commands::MessageId::setBrokerSequenceId**(long long *brokerSequenceId*) [virtual]

6.354.3.15 virtual void **activemq::commands::MessageId::setProducerId**(const **Pointer**<**ProducerId**> & *producerId*) [virtual]

6.354.3.16 virtual void **activemq::commands::MessageId::setProducerSequenceId**(long long *producerSequenceId*) [virtual]

6.354.3.17 void **activemq::commands::MessageId::setTextView**(const std::string & *key*)

6.354.3.18 void **activemq::commands::MessageId::setValue**(const std::string & *key*)

6.354.3.19 virtual std::string **activemq::commands::MessageId::toString**() const [virtual]

Returns a string containing the information for this **DataStructure** (p. 1133) such as its type and value of its elements.

Returns

formatted string useful for debugging.

Reimplemented from **activemq::commands::BaseDataStructure** (p. 531).

6.354.4 Field Documentation

6.354.4.1 `long long activemq::commands::MessageId::brokerSequenceId`
[protected]

6.354.4.2 `const unsigned char activemq::commands::MessageId::ID_MESSAGEID = 110` [static]

6.354.4.3 `Pointer<ProducerId> activemq::commands::MessageId::producerId`
[protected]

6.354.4.4 `long long activemq::commands::MessageId::producerSequenceId`
[protected]

The documentation for this class was generated from the following file:

- `src/main/activemq/commands/MessageId.h`

6.355 activemq::wireformat::openwire::marshal::generated::MessageIdMarshaller Class Reference

Marshaling code for Open Wire Format for **MessageIdMarshaller** (p. 1912).

```
#include <src/main/activemq/wireformat/openwire/marshal/generated/MessageIdMarshaller.h>
```

Inheritance diagram for `activemq::wireformat::openwire::marshal::generated::MessageIdMarshaller`:

Public Member Functions

- **MessageIdMarshaller** ()
- virtual **~MessageIdMarshaller** ()
- virtual **commands::DataStructure * createObject** () const
Creates a new instance of the class that this class is a marshaling director for.
- virtual unsigned char **getDataStructureType** () const
Gets the DataStructureType that this class marshals/unmarshals.

6.355

activemq::wireformat::openwire::marshal::generated::MessageIdMarshaller

Class Reference

1919

- virtual void **tightUnmarshal** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataInputStream** *dataIn, **utils::BooleanStream** *bs)

Tight Un-marhsal to the given stream.

- virtual int **tightMarshal1** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **utils::BooleanStream** *bs)

Tight Marhsal to the given stream.

- virtual void **tightMarshal2** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataOutputStream** *dataOut, **utils::BooleanStream** *bs)

Tight Marhsal to the given stream.

- virtual void **looseUnmarshal** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataInputStream** *dataIn)

Loose Un-marhsal to the given stream.

- virtual void **looseMarshal** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataOutputStream** *dataOut)

Tight Marhsal to the given stream.

6.355.1 Detailed Description

Marshaling code for Open Wire Format for **MessageIdMarshaller** (p. 1912).

NOTE!: This file is auto generated - do not modify! if you need to make a change, please see the Java Classes in the activemq-openwire-generator module

6.355.2 Constructor & Destructor Documentation

6.355.2.1 **activemq::wireformat::openwire::marshal::generated::MessageIdMarshaller::MessageIdMarshaller** ()
[inline]

6.355.2.2 **virtual activemq::wireformat::openwire::marshal::generated::MessageIdMarshaller::~~MessageIdMarshaller** () [inline, virtual]

6.355.3 Member Function Documentation

6.355.3.1 **virtual commands::DataStructure*** **activemq::wireformat::openwire::marshal::generated::MessageIdMarshaller::createObject** () const
[virtual]

Creates a new instance of the class that this class is a marshaling director for.

Returns

newly allocated Command

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1120).

6.355.3.2 `virtual unsigned char activemq::wireformat::openwire::marshal::generated::MessageIdMarshaller::getDataStructureType () const`
[virtual]

Gets the DataStructureType that this class marshals/unmarshals.

Returns

byte Id of this classes DataStructureType

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1122).

6.355.3.3 `virtual void activemq::wireformat::openwire::marshal::generated::MessageIdMarshaller::looseMarshal (OpenWireFormat * format, commands::DataStructure * command, decaf::io::DataOutputStream * ds)` [virtual]

Tight Marshal to the given stream.

Parameters

<i>format</i>	- The OpenWireFormat properties
<i>command</i>	- the object to Marshal
<i>ds</i>	- DataOutputStream to marshal to

Exceptions

<i>IOException</i>	if an error occurs.
--------------------	---------------------

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1123).

6.355.3.4 `virtual void activemq::wireformat::openwire::marshal::generated::MessageIdMarshaller::looseUnmarshal (OpenWireFormat * format, commands::DataStructure * command, decaf::io::DataInputStream * dis)`
[virtual]

Loose Un-marshal to the given stream.

6.355

activemq::wireformat::openwire::marshal::generated::MessageIdMarshaller

Class Reference

1921

Parameters

<i>format</i>	- The OpenWireFormat properties
<i>command</i>	- the object to Un-Marshal
<i>dis</i>	- the DataInputStream to Un-Marshal from

Exceptions

<i>IOException</i>	if an error occurs.
--------------------	---------------------

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller**
(p. 1125).

6.355.3.5 virtual int **activemq::wireformat::openwire::marshal::generated::-**
MessageIdMarshaller::tightMarshal1 (**OpenWireFormat** * *format*,
commands::DataStructure * *command*, **utils::BooleanStream** * *bs*)
[virtual]

Tight Marshal to the given stream.

Parameters

<i>format</i>	The OpenWireFormat properties
<i>command</i>	The object to Marshal
<i>bs</i>	The boolean stream to marshal to.

Exceptions

<i>IOException</i>	if an error occurs.
--------------------	---------------------

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller**
(p. 1127).

6.355.3.6 virtual void **activemq::wireformat::openwire::marshal::generated::-**
MessageIdMarshaller::tightMarshal2 (**OpenWireFormat** * *format*,
commands::DataStructure * *command*, **decaf::io::DataOutputStream** *
ds, **utils::BooleanStream** * *bs*) [virtual]

Tight Marshal to the given stream.

Parameters

<i>format</i>	- The OpenWireFormat properties
<i>command</i>	- the object to Marshal
<i>ds</i>	- the DataOutputStream to Marshal to
<i>bs</i>	- boolean stream to marshal to.

Exceptions

<i>IOException</i>	if an error occurs.
--------------------	---------------------

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1129).

6.355.3.7 virtual void **activemq::wireformat::openwire::marshal::generated-
::MessageIdMarshaller::tightUnmarshal** (**OpenWireFormat** * *format*,
commands::DataStructure * *command*, **decaf::io::DataInputStream** * *dis*,
utils::BooleanStream * *bs*) [virtual]

Tight Un-marhsal to the given stream.

Parameters

<i>format</i>	- The OpenWireFormat properties
<i>command</i>	- the object to Un-Marshal
<i>dis</i>	- the DataInputStream to Un-Marshal from
<i>bs</i>	- boolean stream to unmarshal from.

Exceptions

<i>IOException</i>	if an error occurs.
--------------------	---------------------

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1131).

The documentation for this class was generated from the following file:

- src/main/activemq/wireformat/openwire/marshal/generated/**MessageIdMarshaller.h**

6.356 cms::MessageListener Class Reference

A **MessageListener** (p. 1916) object is used to receive asynchronously delivered messages.

```
#include <src/main/cms/MessageListener.h>
```

Public Member Functions

- virtual **~MessageListener** () throw ()
- virtual void **onMessage** (const **Message** *message)=0 throw ()

*Called asynchronously when a new message is received, the message reference can be to any of the **Message** (p. 1839) types.*

6.356.1 Detailed Description

A **MessageListener** (p. 1916) object is used to receive asynchronously delivered messages.

Since

1.0

6.356.2 Constructor & Destructor Documentation

6.356.2.1 virtual cms::MessageListener::~MessageListener () throw ()
 [virtual]

6.356.3 Member Function Documentation

6.356.3.1 virtual void cms::MessageListener::onMessage (const Message * message) throw () [pure virtual]

Called asynchronously when a new message is received, the message reference can be to any of the **Message** (p. 1839) types.

a dynamic cast is used to find out what type of message this is. The lifetime of this object is only guaranteed to be for life of the onMessage function after this call-back returns the message may no longer exists. Users should copy the data or clone the message if they wish to retain information that was contained in this **Message** (p. 1839).

It is considered a programming error for this method to throw an exception. The method has been tagged with the 'throw()' qualifier, this implies that you application will segfault if you throw an error from an implementation of this method.

Parameters

<i>message</i>	Message (p. 1839) object {const} pointer recipient does not own.
----------------	---

The documentation for this class was generated from the following file:

- src/main/cms/**MessageListener.h**

6.357 activemq::wireformat::openwire::marshal::generated::- MessageMarshaller Class Reference

Marshaling code for Open Wire Format for **MessageMarshaller** (p. 1917).

```
#include <src/main/activemq/wireformat/openwire/marshal/generated/-  
MessageMarshaller.h>
```

Inheritance diagram for activemq::wireformat::openwire::marshal::generated::Message-

Marshaller:

Public Member Functions

- **MessageMarshaller** ()
- virtual **~MessageMarshaller** ()
- virtual void **tightUnmarshal** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataInputStream** *dataIn, **utils::BooleanStream** *bs)
Tight Un-marhsal to the given stream.
- virtual int **tightMarshal1** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **utils::BooleanStream** *bs)
Tight Marhsal to the given stream.
- virtual void **tightMarshal2** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataOutputStream** *dataOut, **utils::BooleanStream** *bs)
Tight Marhsal to the given stream.
- virtual void **looseUnmarshal** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataInputStream** *dataIn)
Loose Un-marhsal to the given stream.
- virtual void **looseMarshal** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataOutputStream** *dataOut)
Tight Marhsal to the given stream.

6.357.1 Detailed Description

Marshaling code for Open Wire Format for **MessageMarshaller** (p. 1917).

NOTE!: This file is auto generated - do not modify! if you need to make a change, please see the Java Classes in the activemq-openwire-generator module

6.357.2 Constructor & Destructor Documentation

- 6.357.2.1 **activemq::wireformat::openwire::marshal::generated::MessageMarshaller::MessageMarshaller** ()
 [inline]
- 6.357.2.2 **virtual activemq::wireformat::openwire::marshal::generated::MessageMarshaller::~~MessageMarshaller** () [inline, virtual]

6.357.3 Member Function Documentation

6.357 activemq::wireformat::openwire::marshal::generated::MessageMarshaller Class Reference 1925

6.357.3.1 virtual void **activemq::wireformat::openwire::marshal::generated::MessageMarshaller::looseMarshal** (**OpenWireFormat** * *format*,
commands::DataStructure * *command*, **decaf::io::DataOutputStream** * *ds*
) [virtual]

Tight Marhsal to the given stream.

Parameters

<i>format</i>	- The OpenWireFormat properties
<i>command</i>	- the object to Marshal
<i>ds</i>	- DataOutputStream to marshal to

Exceptions

<i>IOException</i>	if an error occurs.
--------------------	---------------------

Reimplemented from **activemq::wireformat::openwire::marshal::generated::Base-CommandMarshaller** (p. 500).

Reimplemented in **activemq::wireformat::openwire::marshal::generated::ActiveMQBlobMessageMarshaller** (p. 163), **activemq::wireformat::openwire::marshal::generated::ActiveMQBytesMessageMarshaller** (p. 185), **activemq::wireformat::openwire::marshal::generated::ActiveMQMapMessageMarshaller** (p. 286), **activemq::wireformat::openwire::marshal::generated::ActiveMQMessageMarshaller** (p. 292), **activemq::wireformat::openwire::marshal::generated::ActiveMQObjectMessageMarshaller** (p. 305), **activemq::wireformat::openwire::marshal::generated::ActiveMQStreamMessageMarshaller** (p. 376), and **activemq::wireformat::openwire::marshal::generated::ActiveMQTextMessageMarshaller** (p. 412).

6.357.3.2 virtual void **activemq::wireformat::openwire::marshal::generated::MessageMarshaller::looseUnmarshal** (**OpenWireFormat** * *format*,
commands::DataStructure * *command*, **decaf::io::DataInputStream** * *dis*)
[virtual]

Loose Un-marhsal to the given stream.

Parameters

<i>format</i>	- The OpenWireFormat properties
<i>command</i>	- the object to Un-Marshal
<i>dis</i>	- the DataInputStream to Un-Marshal from

Exceptions

<i>IOException</i>	if an error occurs.
--------------------	---------------------

Reimplemented from **activemq::wireformat::openwire::marshal::generated::Base-CommandMarshaller** (p. 501).

Reimplemented in **activemq::wireformat::openwire::marshal::generated::ActiveMQBlobMessageMarshaller** (p.164), **activemq::wireformat::openwire::marshal::generated::ActiveMQBytesMessageMarshaller** (p.185), **activemq::wireformat::openwire::marshal::generated::ActiveMQMapMessageMarshaller** (p.286), **activemq::wireformat::openwire::marshal::generated::ActiveMQMessageMarshaller** (p.293), **activemq::wireformat::openwire::marshal::generated::ActiveMQObjectMessageMarshaller** (p.306), **activemq::wireformat::openwire::marshal::generated::ActiveMQStreamMessageMarshaller** (p.377), and **activemq::wireformat::openwire::marshal::generated::ActiveMQTextMessageMarshaller** (p.412).

```
6.357.3.3  virtual int activemq::wireformat::openwire::marshal::generated::-
            MessageMarshaller::tightMarshal1 ( OpenWireFormat * format,
            commands::DataStructure * command, utils::BooleanStream * bs )
            [virtual]
```

Tight Marshal to the given stream.

Parameters

<i>format</i>	The OpenWireFormat properties
<i>command</i>	The object to Marshal
<i>bs</i>	The boolean stream to marshal to.

Exceptions

<i>IOException</i>	if an error occurs.
--------------------	---------------------

Reimplemented from **activemq::wireformat::openwire::marshal::generated::BaseCommandMarshaller** (p.503).

Reimplemented in **activemq::wireformat::openwire::marshal::generated::ActiveMQBlobMessageMarshaller** (p.164), **activemq::wireformat::openwire::marshal::generated::ActiveMQBytesMessageMarshaller** (p.186), **activemq::wireformat::openwire::marshal::generated::ActiveMQMapMessageMarshaller** (p.286), **activemq::wireformat::openwire::marshal::generated::ActiveMQMessageMarshaller** (p.293), **activemq::wireformat::openwire::marshal::generated::ActiveMQObjectMessageMarshaller** (p.306), **activemq::wireformat::openwire::marshal::generated::ActiveMQStreamMessageMarshaller** (p.377), and **activemq::wireformat::openwire::marshal::generated::ActiveMQTextMessageMarshaller** (p.413).

```
6.357.3.4  virtual void activemq::wireformat::openwire::marshal::generated-
            ::MessageMarshaller::tightMarshal2 ( OpenWireFormat * format,
            commands::DataStructure * command, decaf::io::DataOutputStream *
            ds, utils::BooleanStream * bs ) [virtual]
```

Tight Marshal to the given stream.

6.357 activemq::wireformat::openwire::marshal::generated::MessageMarshaller Class Reference 1927

Parameters

<i>format</i>	- The OpenWireFormat properties
<i>command</i>	- the object to Marshal
<i>ds</i>	- the DataOutputStream to Marshal to
<i>bs</i>	- boolean stream to marshal to.

Exceptions

<i>IOException</i>	if an error occurs.
--------------------	---------------------

Reimplemented from **activemq::wireformat::openwire::marshal::generated::Base-CommandMarshaller** (p. 504).

Reimplemented in **activemq::wireformat::openwire::marshal::generated::ActiveMQBlobMessageMarshaller** (p. 165), **activemq::wireformat::openwire::marshal::generated::ActiveMQBytesMessageMarshaller** (p. 186), **activemq::wireformat::openwire::marshal::generated::ActiveMQMapMessageMarshaller** (p. 287), **activemq::wireformat::openwire::marshal::generated::ActiveMQMessageMarshaller** (p. 294), **activemq::wireformat::openwire::marshal::generated::ActiveMQObjectMessageMarshaller** (p. 307), **activemq::wireformat::openwire::marshal::generated::ActiveMQStreamMessageMarshaller** (p. 378), and **activemq::wireformat::openwire::marshal::generated::ActiveMQTextMessageMarshaller** (p. 413).

6.357.3.5 virtual void **activemq::wireformat::openwire::marshal::generated::MessageMarshaller::tightUnmarshal** (**OpenWireFormat** * *format*, **commands::DataStructure** * *command*, **decaf::io::DataInputStream** * *dis*, **utils::BooleanStream** * *bs*) [virtual]

Tight Un-marhsal to the given stream.

Parameters

<i>format</i>	- The OpenWireFormat properties
<i>command</i>	- the object to Un-Marshal
<i>dis</i>	- the DataInputStream to Un-Marshal from
<i>bs</i>	- boolean stream to unmarshal from.

Exceptions

<i>IOException</i>	if an error occurs.
--------------------	---------------------

Reimplemented from **activemq::wireformat::openwire::marshal::generated::Base-CommandMarshaller** (p. 505).

Reimplemented in **activemq::wireformat::openwire::marshal::generated::ActiveMQBlobMessageMarshaller** (p. 165), **activemq::wireformat::openwire::marshal::generated::ActiveMQBytesMessageMarshaller** (p. 187), **activemq::wireformat::openwire::marshal::generated::ActiveMQMapMessageMarshaller** (p. 287), **activemq::wireformat::openwire::marshal::generated::ActiveMQMessageMarshaller**

(p. 294), `activemq::wireformat::openwire::marshal::generated::ActiveMQObjectMessageMarshaller` (p. 307), `activemq::wireformat::openwire::marshal::generated::ActiveMQStreamMessageMarshaller` (p. 378), and `activemq::wireformat::openwire::marshal::generated::ActiveMQTextMessageMarshaller` (p. 414).

The documentation for this class was generated from the following file:

- `src/main/activemq/wireformat/openwire/marshal/generated/MessageMarshaller.h`

6.358 cms::MessageNotReadableException Class Reference

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

```
#include <src/main/cms/MessageNotReadableException.h>
```

Inheritance diagram for `cms::MessageNotReadableException`:

Public Member Functions

- `MessageNotReadableException ()`
- `MessageNotReadableException (const MessageNotReadableException &ex)`
- `MessageNotReadableException (const std::string &message)`
- `MessageNotReadableException (const std::string &message, const std::exception *cause)`
- `MessageNotReadableException (const std::string &message, const std::exception *cause, const std::vector< std::pair< std::string, int > > &stackTrace)`
- `virtual ~MessageNotReadableException () throw ()`

6.358.1 Detailed Description

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

Since

1.3

6.358.2 Constructor & Destructor Documentation

6.358.2.1 cms::MessageNotReadableException::MessageNotReadableException ()

- 6.358.2.2 `cms::MessageNotReadableException::MessageNotReadableException (const MessageNotReadableException & ex)`
- 6.358.2.3 `cms::MessageNotReadableException::MessageNotReadableException (const std::string & message)`
- 6.358.2.4 `cms::MessageNotReadableException::MessageNotReadableException (const std::string & message, const std::exception * cause)`
- 6.358.2.5 `cms::MessageNotReadableException::MessageNotReadableException (const std::string & message, const std::exception * cause, const std::vector< std::pair< std::string, int > > & stackTrace)`
- 6.358.2.6 `virtual cms::MessageNotReadableException::~MessageNotReadableException () throw () [virtual]`

The documentation for this class was generated from the following file:

- `src/main/cms/MessageNotReadableException.h`

6.359 cms::MessageNotWriteableException Class Reference

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

```
#include <src/main/cms/MessageNotWriteableException.h>
```

Inheritance diagram for cms::MessageNotWriteableException:

Public Member Functions

- **MessageNotWriteableException** ()
- **MessageNotWriteableException** (const **MessageNotWriteableException** &ex)
- **MessageNotWriteableException** (const std::string &message)
- **MessageNotWriteableException** (const std::string &message, const std::exception *cause)
- **MessageNotWriteableException** (const std::string &message, const std::exception *cause, const std::vector< std::pair< std::string, int > > &stackTrace)
- virtual **~MessageNotWriteableException** () throw ()

6.359.1 Detailed Description

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

Since

1.3

6.359.2 Constructor & Destructor Documentation

6.359.2.1 `cms::MessageNotWriteableException::MessageNotWriteableException ()`

6.359.2.2 `cms::MessageNotWriteableException::MessageNotWriteableException (const MessageNotWriteableException & ex)`

6.359.2.3 `cms::MessageNotWriteableException::MessageNotWriteableException (const std::string & message)`

6.359.2.4 `cms::MessageNotWriteableException::MessageNotWriteableException (const std::string & message, const std::exception * cause)`

6.359.2.5 `cms::MessageNotWriteableException::MessageNotWriteableException (const std::string & message, const std::exception * cause, const std::vector< std::pair< std::string, int > > & stackTrace)`

6.359.2.6 `virtual cms::MessageNotWriteableException::~MessageNotWriteableException () throw () [virtual]`

The documentation for this class was generated from the following file:

- `src/main/cms/MessageNotWriteableException.h`

6.360 cms::MessageProducer Class Reference

A client uses a **MessageProducer** (p.1924) object to send messages to a **Destination** (p.1210).

```
#include <src/main/cms/MessageProducer.h>
```

Inheritance diagram for cms::MessageProducer:

Public Member Functions

- `virtual ~MessageProducer () throw ()`

- virtual void **send** (**Message** *message)=0
Sends the message to the default producer destination, but does not take ownership of the message, caller must still destroy it.
- virtual void **send** (**Message** *message, int deliveryMode, int priority, long long timeToLive)=0
Sends the message to the default producer destination, but does not take ownership of the message, caller must still destroy it.
- virtual void **send** (const **Destination** *destination, **Message** *message)=0
Sends the message to the designated destination, but does not take ownership of the message, caller must still destroy it.
- virtual void **send** (const **Destination** *destination, **Message** *message, int deliveryMode, int priority, long long timeToLive)=0
Sends the message to the designated destination, but does not take ownership of the message, caller must still destroy it.
- virtual void **setDeliveryMode** (int mode)=0
Sets the delivery mode for this Producer.
- virtual int **getDeliveryMode** () const =0
Gets the delivery mode for this Producer.
- virtual void **setDisableMessageID** (bool value)=0
*Sets if **Message** (p. 1839) Ids are disabled for this Producer.*
- virtual bool **getDisableMessageID** () const =0
*Gets if **Message** (p. 1839) Ids are disabled for this Producer.*
- virtual void **setDisableMessageTimeStamp** (bool value)=0
*Sets if **Message** (p. 1839) Time Stamps are disabled for this Producer.*
- virtual bool **getDisableMessageTimeStamp** () const =0
*Gets if **Message** (p. 1839) Time Stamps are disabled for this Producer.*
- virtual void **setPriority** (int priority)=0
Sets the Priority that this Producers sends messages at.
- virtual int **getPriority** () const =0
Gets the Priority level that this producer sends messages at.
- virtual void **setTimeToLive** (long long time)=0
Sets the Time to Live that this Producers sends messages with.
- virtual long long **getTimeToLive** () const =0
Gets the Time to Live that this producer sends messages with.

6.360.1 Detailed Description

A client uses a **MessageProducer** (p.1924) object to send messages to a **Destination** (p.1210).

A **MessageProducer** (p.1924) object is created by passing a **Destination** (p.1210) object to a message-producer creation method supplied by a session.

A client also has the option of creating a message producer without supplying a destination. In this case, a **Destination** (p.1210) must be provided with every send operation.

A typical use for this kind of message producer is to send replies to requests using the request's CMSReplyTo destination.

A client can specify a default delivery mode, priority, and time to live for messages sent by a message producer. It can also specify the delivery mode, priority, and time to live for an individual message.

A client can specify a time-to-live value in milliseconds for each message it sends. This value defines a message expiration time that is the sum of the message's time-to-live and the GMT when it is sent (for transacted sends, this is the time the client sends the message, not the time the transaction is committed).

Since

1.0

6.360.2 Constructor & Destructor Documentation

6.360.2.1 `virtual cms::MessageProducer::~MessageProducer () throw ()`
[virtual]

6.360.3 Member Function Documentation

6.360.3.1 `virtual int cms::MessageProducer::getDeliveryMode () const` [pure virtual]

Gets the delivery mode for this Producer.

Returns

The **DeliveryMode** (p. 1195)

Exceptions

CMSException (p. 826)	- if an internal error occurs.
---------------------------------	--------------------------------

Implemented in **activemq::core::ActiveMQProducer** (p. 310), and **activemq::cmsutil::CachedProducer** (p. 740).

6.360.3.2 `virtual bool cms::MessageProducer::getDisableMessageID () const`
[pure virtual]

Gets if **Message** (p. 1839) Ids are disabled for this Producer.

Returns

boolean indicating enable / disable (true / false)

Exceptions

CMSException (p. 826)	- if an internal error occurs.
---------------------------------	--------------------------------

Implemented in **activemq::core::ActiveMQProducer** (p.310), and **activemq-
::cmsutil::CachedProducer** (p. 740).

6.360.3.3 `virtual bool cms::MessageProducer::getDisableMessageTimeStamp ()
const [pure virtual]`

Gets if **Message** (p. 1839) Time Stamps are disabled for this Producer.

Returns

boolean indicating enable / disable (true / false)

Exceptions

CMSException (p. 826)	- if an internal error occurs.
---------------------------------	--------------------------------

Implemented in **activemq::core::ActiveMQProducer** (p.310), and **activemq-
::cmsutil::CachedProducer** (p. 741).

6.360.3.4 `virtual int cms::MessageProducer::getPriority () const [pure
virtual]`

Gets the Priority level that this producer sends messages at.

Returns

int based priority level

Exceptions

CMSException (p. 826)	- if an internal error occurs.
---------------------------------	--------------------------------

Implemented in **activemq::core::ActiveMQProducer** (p.311), and **activemq-
::cmsutil::CachedProducer** (p. 741).

6.360.3.5 `virtual long long cms::MessageProducer::getTimeToLive () const [pure
virtual]`

Gets the Time to Live that this producer sends messages with.

Returns

Time to live value in milliseconds

Exceptions

CMSException (p. 826)	- if an internal error occurs.
---------------------------------	--------------------------------

Implemented in **activemq::core::ActiveMQProducer** (p. 312), and **activemq::cmsutil::CachedProducer** (p. 741).

6.360.3.6 `virtual void cms::MessageProducer::send (Message * message) [pure virtual]`

Sends the message to the default producer destination, but does not take ownership of the message, caller must still destroy it.

Uses default values for deliveryMode, priority, and time to live.

Parameters

<i>message</i>	The message to be sent.
----------------	-------------------------

Exceptions

CMSException (p. 826)	- if an internal error occurs while sending the message.
MessageFormatException (p. 1906)	- if an Invalid Message (p. 1839) is given.
InvalidDestinationException (p. 1535)	- if a client uses this method with a MessageProducer (p. 1924) with an invalid destination.
UnsupportedOperationException (p. 2827)	- if a client uses this method with a MessageProducer (p. 1924) that did not specify a destination at creation time.

Implemented in **activemq::core::ActiveMQProducer** (p. 312), and **activemq::cmsutil::CachedProducer** (p. 742).

6.360.3.7 `virtual void cms::MessageProducer::send (Message * message, int deliveryMode, int priority, long long timeToLive) [pure virtual]`

Sends the message to the default producer destination, but does not take ownership of the message, caller must still destroy it.

Parameters

<i>message</i>	The message to be sent.
<i>delivery-Mode</i>	The delivery mode to be used.
<i>priority</i>	The priority for this message.
<i>timeToLive</i>	The time to live value for this message in milliseconds.

Exceptions

CMSException (p. 826)	- if an internal error occurs while sending the message.
MessageFormatException (p. 1906)	- if an Invalid Message (p. 1839) is given.
InvalidDestinationException (p. 1535)	- if a client uses this method with a MessageProducer (p. 1924) with an invalid destination.
UnsupportedOperationException (p. 2827)	- if a client uses this method with a MessageProducer (p. 1924) that did not specify a destination at creation time.

Implemented in **activemq::core::ActiveMQProducer** (p.313), and **activemq::cmsutil::CachedProducer** (p. 742).

6.360.3.8 virtual void cms::MessageProducer::send (const Destination * *destination*, Message * *message*) [pure virtual]

Sends the message to the designated destination, but does not take ownership of the message, caller must still destroy it.

Uses default values for deliveryMode, priority, and time to live.

Parameters

<i>destination</i>	The destination on which to send the message
<i>message</i>	the message to be sent.

Exceptions

CMSException (p. 826)	- if an internal error occurs while sending the message.
MessageFormatException (p. 1906)	- if an Invalid Message (p. 1839) is given.
InvalidDestinationException (p. 1535)	- if a client uses this method with a MessageProducer (p. 1924) with an invalid destination.

<i>Unsupported-OperationException</i> (p. 2827)	- if a client uses this method with a MessageProducer (p. 1924) that did not specify a destination at creation time.
---	---

Implemented in **activemq::core::ActiveMQProducer** (p. 313), and **activemq::cmsutil::CachedProducer** (p. 743).

6.360.3.9 **virtual void cms::MessageProducer::send (const Destination * destination, Message * message, int deliveryMode, int priority, long long timeToLive)**
[pure virtual]

Sends the message to the designated destination, but does not take ownership of the message, caller must still destroy it.

Parameters

<i>destination</i>	The destination on which to send the message
<i>message</i>	The message to be sent.
<i>delivery-Mode</i>	The delivery mode to be used.
<i>priority</i>	The priority for this message.
<i>timeToLive</i>	The time to live value for this message in milliseconds.

Exceptions

<i>CMSException</i> (p. 826)	- if an internal error occurs while sending the message.
<i>MessageFormatException</i> (p. 1906)	- if an Invalid Message (p. 1839) is given.
<i>InvalidDestinationException</i> (p. 1535)	- if a client uses this method with a MessageProducer (p. 1924) with an invalid destination.
<i>Unsupported-OperationException</i> (p. 2827)	- if a client uses this method with a MessageProducer (p. 1924) that did not specify a destination at creation time.

Implemented in **activemq::core::ActiveMQProducer** (p. 314), and **activemq::cmsutil::CachedProducer** (p. 743).

6.360.3.10 **virtual void cms::MessageProducer::setDeliveryMode (int mode)**
[pure virtual]

Sets the delivery mode for this Producer.

Parameters

<i>mode</i>	The DeliveryMode (p. 1195)
-------------	-----------------------------------

Exceptions

CMSException (p. 826)	- if an internal error occurs.
---------------------------------	--------------------------------

Implemented in **activemq::core::ActiveMQProducer** (p.314), and **activemq::cmsutil::CachedProducer** (p. 744).

6.360.3.11 virtual void **cms::MessageProducer::setDisableMessageID** (bool *value*)
[pure virtual]

Sets if **Message** (p. 1839) Ids are disabled for this Producer.

Parameters

<i>value</i>	boolean indicating enable / disable (true / false)
--------------	--

Exceptions

CMSException (p. 826)	- if an internal error occurs.
---------------------------------	--------------------------------

Implemented in **activemq::core::ActiveMQProducer** (p.315), and **activemq::cmsutil::CachedProducer** (p. 744).

6.360.3.12 virtual void **cms::MessageProducer::setDisableMessageTimeStamp** (bool *value*) [pure virtual]

Sets if **Message** (p. 1839) Time Stamps are disabled for this Producer.

Parameters

<i>value</i>	- boolean indicating enable / disable (true / false)
--------------	--

Exceptions

CMSException (p. 826)	- if an internal error occurs.
---------------------------------	--------------------------------

Implemented in **activemq::core::ActiveMQProducer** (p.315), and **activemq::cmsutil::CachedProducer** (p. 744).

6.360.3.13 `virtual void cms::MessageProducer::setPriority (int priority)` [pure virtual]

Sets the Priority that this Producers sends messages at.

Parameters

<i>priority</i>	int value for Priority level
-----------------	------------------------------

Exceptions

<i>CMSEException</i> (p. 826)	- if an internal error occurs.
---	--------------------------------

Implemented in `activemq::core::ActiveMQProducer` (p. 315), and `activemq::cmsutil::CachedProducer` (p. 745).

6.360.3.14 `virtual void cms::MessageProducer::setTimeToLive (long long time)` [pure virtual]

Sets the Time to Live that this Producers sends messages with.

This value will be used if the time to live is not specified via the send method.

Parameters

<i>time</i>	default time to live value in milliseconds
-------------	--

Exceptions

<i>CMSEException</i> (p. 826)	- if an internal error occurs.
---	--------------------------------

Implemented in `activemq::core::ActiveMQProducer` (p. 315), and `activemq::cmsutil::CachedProducer` (p. 745).

The documentation for this class was generated from the following file:

- `src/main/cms/MessageProducer.h`

6.361 `activemq::wireformat::openwire::utils::MessageProperty-Interceptor` Class Reference

Used the base `ActiveMQMessage` class to intercept calls to get and set properties in order to capture the calls that use the reserved JMS properties and get and set them in the `OpenWire` Message properties.


```
#include <src/main/activemq/wireformat/openwire/utils/-  
MessagePropertyInterceptor.h>
```

Public Member Functions

- **MessagePropertyInterceptor** (commands::Message *message, util::PrimitiveMap *properties)

Constructor, accepts the Message that will be used to store JMS reserved property values, and the PrimitiveMap to get and set the rest to.

- virtual ~**MessagePropertyInterceptor** () throw ()
- virtual bool **getBooleanProperty** (const std::string &name) const
Gets a boolean property.
- virtual unsigned char **getByteProperty** (const std::string &name) const
Gets a byte property.
- virtual double **getDoubleProperty** (const std::string &name) const
Gets a double property.
- virtual float **getFloatProperty** (const std::string &name) const
Gets a float property.
- virtual int **getIntProperty** (const std::string &name) const
Gets a int property.
- virtual long long **getLongProperty** (const std::string &name) const
Gets a long property.
- virtual short **getShortProperty** (const std::string &name) const
Gets a short property.
- virtual std::string **getStringProperty** (const std::string &name) const
Gets a string property.
- virtual void **setBooleanProperty** (const std::string &name, bool value)
Sets a boolean property.
- virtual void **setByteProperty** (const std::string &name, unsigned char value)
Sets a byte property.
- virtual void **setDoubleProperty** (const std::string &name, double value)
Sets a double property.
- virtual void **setFloatProperty** (const std::string &name, float value)
Sets a float property.
- virtual void **setIntProperty** (const std::string &name, int value)
Sets a int property.
- virtual void **setLongProperty** (const std::string &name, long long value)
Sets a long property.
- virtual void **setShortProperty** (const std::string &name, short value)
Sets a short property.
- virtual void **setStringProperty** (const std::string &name, const std::string &value)
Sets a string property.

6.361.1 Detailed Description

Used the base ActiveMQMessage class to intercept calls to get and set properties in order to capture the calls that use the reserved JMS properties and get and set them in the OpenWire Message properties.

Currently the only properties that are intercepted and handled are:

Name | Conversion Supported ----- JMSXDelivery-
Count | Int, Long, String JMSXGroupID | String JMSXGroupSeq | Int, Long, String

6.361.2 Constructor & Destructor Documentation

6.361.2.1 `activemq::wireformat::openwire::utils::MessagePropertyInterceptor::MessagePropertyInterceptor (commands::Message * message, util::PrimitiveMap * properties)`

Constructor, accepts the Message that will be used to store JMS reserved property values, and the PrimitiveMap to get and set the rest to.

Parameters

<i>message</i>	- The Message to store reserved property data in
<i>properties</i>	- The PrimitiveMap to store the rest of the properties in.

Exceptions

<i>NullPointerException</i>	if either param is NULL
-----------------------------	-------------------------

6.361.2.2 `virtual activemq::wireformat::openwire::utils::MessagePropertyInterceptor::~MessagePropertyInterceptor () throw ()`
[virtual]

6.361.3 Member Function Documentation

6.361.3.1 `virtual bool activemq::wireformat::openwire::utils::MessagePropertyInterceptor::getBooleanProperty (const std::string & name) const`
[virtual]

Gets a boolean property.

Parameters

<i>name</i>	The name of the property to retrieve.
-------------	---------------------------------------

6.361 activemq::wireformat::openwire::utils::MessagePropertyInterceptor Class Reference 1941

Returns

The value for the named property.

6.361.3.2 `virtual unsigned char activemq::wireformat::openwire::utils::MessagePropertyInterceptor::getByteProperty (const std::string & name) const`
`[virtual]`

Gets a byte property.

Parameters

<i>name</i>	The name of the property to retrieve.
-------------	---------------------------------------

Returns

The value for the named property.

6.361.3.3 `virtual double activemq::wireformat::openwire::utils::MessagePropertyInterceptor::getDoubleProperty (const std::string & name) const`
`[virtual]`

Gets a double property.

Parameters

<i>name</i>	The name of the property to retrieve.
-------------	---------------------------------------

Returns

The value for the named property.

6.361.3.4 `virtual float activemq::wireformat::openwire::utils::MessagePropertyInterceptor::getFloatProperty (const std::string & name) const`
`[virtual]`

Gets a float property.

Parameters

<i>name</i>	The name of the property to retrieve.
-------------	---------------------------------------

Returns

The value for the named property.

6.361.3.5 `virtual int activemq::wireformat::openwire::utils::MessageProperty-
Interceptor::getIntProperty (const std::string & name) const`
[virtual]

Gets a int property.

Parameters

<i>name</i>	The name of the property to retrieve.
-------------	---------------------------------------

Returns

The value for the named property.

6.361.3.6 `virtual long long activemq::wireformat::openwire::utils::Message-
PropertyInterceptor::getLongProperty (const std::string & name) const`
[virtual]

Gets a long property.

Parameters

<i>name</i>	The name of the property to retrieve.
-------------	---------------------------------------

Returns

The value for the named property.

6.361.3.7 `virtual short activemq::wireformat::openwire::utils::MessageProperty-
Interceptor::getShortProperty (const std::string & name) const`
[virtual]

Gets a short property.

Parameters

<i>name</i>	The name of the property to retrieve.
-------------	---------------------------------------

6.361 activemq::wireformat::openwire::utils::MessagePropertyInterceptor Class Reference 1943

Returns

The value for the named property.

6.361.3.8 `virtual std::string activemq::wireformat::openwire::utils::MessagePropertyInterceptor::getStringProperty (const std::string & name) const`
[virtual]

Gets a string property.

Parameters

<i>name</i>	The name of the property to retrieve.
-------------	---------------------------------------

Returns

The value for the named property.

6.361.3.9 `virtual void activemq::wireformat::openwire::utils::MessagePropertyInterceptor::setBooleanProperty (const std::string & name, bool value)`
[virtual]

Sets a boolean property.

Parameters

<i>name</i>	The name of the property to retrieve.
<i>value</i>	The value for the named property.

6.361.3.10 `virtual void activemq::wireformat::openwire::utils::MessagePropertyInterceptor::setByteProperty (const std::string & name, unsigned char value)`
[virtual]

Sets a byte property.

Parameters

<i>name</i>	The name of the property to retrieve.
<i>value</i>	The value for the named property.

6.361.3.11 `virtual void activemq::wireformat::openwire::utils::MessagePropertyInterceptor::setDoubleProperty (const std::string & name, double value)`
[virtual]

Sets a double property.

Parameters

<i>name</i>	The name of the property to retrieve.
<i>value</i>	The value for the named property.

6.361.3.12 `virtual void activemq::wireformat::openwire::utils::MessageProperty-
Interceptor::setFloatProperty (const std::string & name, float value)
[virtual]`

Sets a float property.

Parameters

<i>name</i>	The name of the property to retrieve.
<i>value</i>	The value for the named property.

6.361.3.13 `virtual void activemq::wireformat::openwire::utils::MessageProperty-
Interceptor::setIntProperty (const std::string & name, int value)
[virtual]`

Sets a int property.

Parameters

<i>name</i>	The name of the property to retrieve.
<i>value</i>	The value for the named property.

6.361.3.14 `virtual void activemq::wireformat::openwire::utils::MessageProperty-
Interceptor::setLongProperty (const std::string & name, long long value)
[virtual]`

Sets a long property.

Parameters

<i>name</i>	The name of the property to retrieve.
<i>value</i>	The value for the named property.

6.361.3.15 `virtual void activemq::wireformat::openwire::utils::MessageProperty-
Interceptor::setShortProperty (const std::string & name, short value)
[virtual]`

Sets a short property.

Parameters

<i>name</i>	The name of the property to retrieve.
<i>value</i>	The value for the named property.

6.361.3.16 virtual void activemq::wireformat::openwire::utils::MessageProperty-Interceptor::setStringProperty (const std::string & *name*, const std::string & *value*) [virtual]

Sets a string property.

Parameters

<i>name</i>	The name of the property to retrieve.
<i>value</i>	The value for the named property.

The documentation for this class was generated from the following file:

- src/main/activemq/wireformat/openwire/utils/MessagePropertyInterceptor.h

6.362 activemq::commands::MessagePull Class Reference

```
#include <src/main/activemq/commands/MessagePull.h>
```

Inheritance diagram for activemq::commands::MessagePull:

Public Member Functions

- **MessagePull** ()
- virtual ~**MessagePull** ()
- virtual unsigned char **getDataStructureType** () const
*Get the **DataStructure** (p. 1133) Type as defined in CommandTypes.h.*
- virtual **MessagePull** * **cloneDataStructure** () const
Clone this object and return a new instance that the caller now owns, this will be an exact copy of this one.
- virtual void **copyDataStructure** (const **DataStructure** *src)
Copy the contents of the passed object into this objects members, overwriting any existing data.
- virtual std::string **toString** () const
*Returns a string containing the information for this **DataStructure** (p. 1133) such as its type and value of its elements.*
- virtual bool **equals** (const **DataStructure** *value) const
*Compares the **DataStructure** (p. 1133) passed in to this one, and returns if they are equivalent.*

- virtual const **Pointer** < **ConsumerId** > & **getConsumerId** () const
- virtual **Pointer**< **ConsumerId** > & **getConsumerId** ()
- virtual void **setConsumerId** (const **Pointer**< **ConsumerId** > &**consumerId**)
- virtual const **Pointer** < **ActiveMQDestination** > & **getDestination** () const
- virtual **Pointer** < **ActiveMQDestination** > & **getDestination** ()
- virtual void **setDestination** (const **Pointer**< **ActiveMQDestination** > &**destination**)
- virtual long long **getTimeout** () const
- virtual void **setTimeout** (long long **timeout**)
- virtual const std::string & **getCorrelationId** () const
- virtual std::string & **getCorrelationId** ()
- virtual void **setCorrelationId** (const std::string &**correlationId**)
- virtual const **Pointer** < **MessageId** > & **getMessageId** () const
- virtual **Pointer**< **MessageId** > & **getMessageId** ()
- virtual void **setMessageId** (const **Pointer**< **MessageId** > &**messageId**)
- virtual **Pointer**< **Command** > **visit** (activemq::state::CommandVisitor *visitor)

Allows a Visitor to visit this command and return a response to the command based on the command type being visited.

Static Public Attributes

- static const unsigned char **ID_MESSAGEPULL** = 20

Protected Attributes

- **Pointer**< **ConsumerId** > **consumerId**
- **Pointer**< **ActiveMQDestination** > **destination**
- long long **timeout**
- std::string **correlationId**
- **Pointer**< **MessageId** > **messageId**

6.362.1 Constructor & Destructor Documentation

6.362.1.1 **activemq::commands::MessagePull::MessagePull** ()

6.362.1.2 **virtual activemq::commands::MessagePull::~~MessagePull** ()
[virtual]

6.362.2 Member Function Documentation

6.362.2.1 **virtual MessagePull* activemq::commands::MessagePull::cloneDataStructure** () const [virtual]

Clone this object and return a new instance that the caller now owns, this will be an exact copy of this one.

Returns

new copy of this object.

Implements **activemq::commands::DataStructure** (p. 1133).

6.362.2.2 `virtual void activemq::commands::MessagePull::copyDataStructure (const DataStructure * src) [virtual]`

Copy the contents of the passed object into this objects members, overwriting any existing data.

Returns

src - Source Object

Reimplemented from **activemq::commands::BaseCommand** (p. 493).

6.362.2.3 `virtual bool activemq::commands::MessagePull::equals (const DataStructure * value) const [virtual]`

Compares the **DataStructure** (p. 1133) passed in to this one, and returns if they are equivalent.

Equivalent here means that they are of the same type, and that each element of the objects are the same.

Returns

true if **DataStructure** (p. 1133)'s are Equal.

Reimplemented from **activemq::commands::BaseCommand** (p. 494).

6.362.2.4 `virtual const Pointer<ConsumerId>& activemq::commands::MessagePull::getConsumerId () const [virtual]`

6.362.2.5 `virtual Pointer<ConsumerId>& activemq::commands::MessagePull::getConsumerId () [virtual]`

6.362.2.6 `virtual const std::string& activemq::commands::MessagePull::getCorrelationId () const [virtual]`

6.362.2.7 `virtual std::string& activemq::commands::MessagePull::getCorrelationId () [virtual]`

6.362.2.8 `virtual unsigned char activemq::commands::MessagePull::getDataStructureType () const [virtual]`

Get the **DataStructure** (p. 1133) Type as defined in CommandTypes.h.

Returns

The type of the data structure

Implements **activemq::commands::DataStructure** (p. 1137).

- 6.362.2.9 `virtual const Pointer<ActiveMQDestination>& activemq::commands::MessagePull::getDestination () const`
[virtual]
- 6.362.2.10 `virtual Pointer<ActiveMQDestination>& activemq::commands::MessagePull::getDestination ()`
[virtual]
- 6.362.2.11 `virtual const Pointer<MessageId>& activemq::commands::MessagePull::getMessageId () const`
[virtual]
- 6.362.2.12 `virtual Pointer<MessageId>& activemq::commands::MessagePull::getMessageId ()` [virtual]
- 6.362.2.13 `virtual long long activemq::commands::MessagePull::getTimeout () const` [virtual]
- 6.362.2.14 `virtual void activemq::commands::MessagePull::setConsumerId (const Pointer< ConsumerId > & consumerId)` [virtual]
- 6.362.2.15 `virtual void activemq::commands::MessagePull::setCorrelationId (const std::string & correlationId)` [virtual]
- 6.362.2.16 `virtual void activemq::commands::MessagePull::setDestination (const Pointer< ActiveMQDestination > & destination)` [virtual]
- 6.362.2.17 `virtual void activemq::commands::MessagePull::setMessageId (const Pointer< MessageId > & messageId)` [virtual]
- 6.362.2.18 `virtual void activemq::commands::MessagePull::setTimeout (long long timeout)` [virtual]
- 6.362.2.19 `virtual std::string activemq::commands::MessagePull::toString () const`
[virtual]

Returns a string containing the information for this **DataStructure** (p. 1133) such as its type and value of its elements.

Returns

formatted string useful for debugging.

Reimplemented from **activemq::commands::BaseCommand** (p. 498).

6.363

activemq::wireformat::openwire::marshal::generated::MessagePullMarshaller

Class Reference

1949

6.362.2.20 `virtual Pointer<Command> activemq::commands::MessagePull::visit (activemq::state::CommandVisitor * visitor) [virtual]`

Allows a Visitor to visit this command and return a response to the command based on the command type being visited.

The command will call the proper processXXX method in the visitor.

Returns

a **Response** (p. 2298) to the visitor being called or NULL if no response.

Implements **activemq::commands::Command** (p. 871).

6.362.3 Field Documentation

6.362.3.1 `Pointer<ConsumerId> activemq::commands::MessagePull::consumerId [protected]`

6.362.3.2 `std::string activemq::commands::MessagePull::correlationId [protected]`

6.362.3.3 `Pointer<ActiveMQDestination> activemq::commands::MessagePull::destination [protected]`

6.362.3.4 `const unsigned char activemq::commands::MessagePull::ID_MESSAGEPULL = 20 [static]`

6.362.3.5 `Pointer<MessageId> activemq::commands::MessagePull::messageId [protected]`

6.362.3.6 `long long activemq::commands::MessagePull::timeout [protected]`

The documentation for this class was generated from the following file:

- `src/main/activemq/commands/MessagePull.h`

6.363 **activemq::wireformat::openwire::marshal::generated::MessagePullMarshaller Class Reference**

Marshaling code for Open Wire Format for **MessagePullMarshaller** (p. 1944).

```
#include <src/main/activemq/wireformat/openwire/marshal/generated/-  
MessagePullMarshaller.h>
```

Inheritance diagram for **activemq::wireformat::openwire::marshal::generated::MessagePullMarshaller**:

Public Member Functions

- **MessagePullMarshaller** ()
- virtual **~MessagePullMarshaller** ()
- virtual **commands::DataStructure * createObject** () const
Creates a new instance of the class that this class is a marshaling director for.
- virtual unsigned char **getDataStructureType** () const
Gets the DataStructureType that this class marshals/unmarshals.
- virtual void **tightUnmarshal** (**OpenWireFormat *wireFormat**, **commands::DataStructure *dataStructure**, **decaf::io::DataInputStream *dataIn**, **utils::BooleanStream *bs**)
Tight Un-marhsal to the given stream.
- virtual int **tightMarshal1** (**OpenWireFormat *wireFormat**, **commands::DataStructure *dataStructure**, **utils::BooleanStream *bs**)
Tight Marhsal to the given stream.
- virtual void **tightMarshal2** (**OpenWireFormat *wireFormat**, **commands::DataStructure *dataStructure**, **decaf::io::DataOutputStream *dataOut**, **utils::BooleanStream *bs**)
Tight Marhsal to the given stream.
- virtual void **looseUnmarshal** (**OpenWireFormat *wireFormat**, **commands::DataStructure *dataStructure**, **decaf::io::DataInputStream *dataIn**)
Loose Un-marhsal to the given stream.
- virtual void **looseMarshal** (**OpenWireFormat *wireFormat**, **commands::DataStructure *dataStructure**, **decaf::io::DataOutputStream *dataOut**)
Tight Marhsal to the given stream.

6.363.1 Detailed Description

Marshaling code for Open Wire Format for **MessagePullMarshaller** (p. 1944).

NOTE!: This file is auto generated - do not modify! if you need to make a change, please see the Java Classes in the activemq-openwire-generator module

6.363.2 Constructor & Destructor Documentation

- 6.363.2.1 **activemq::wireformat::openwire::marshal::generated::MessagePullMarshaller::MessagePullMarshaller** ()
[inline]
- 6.363.2.2 **virtual activemq::wireformat::openwire::marshal::generated::MessagePullMarshaller::~~MessagePullMarshaller** () [inline, virtual]

6.363.3 Member Function Documentation

6.363

activemq::wireformat::openwire::marshal::generated::MessagePullMarshaller

Class Reference

1951

6.363.3.1 `virtual commands::DataStructure* activemq::wireformat::openwire-
::marshal::generated::MessagePullMarshaller::createObject () const
[virtual]`

Creates a new instance of the class that this class is a marshaling director for.

Returns

newly allocated Command

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller**
(p. 1120).

6.363.3.2 `virtual unsigned char activemq::wireformat::openwire::marshal-
::generated::MessagePullMarshaller::getDataStructureType () const
[virtual]`

Gets the DataStructureType that this class marshals/unmarshals.

Returns

byte Id of this classes DataStructureType

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller**
(p. 1122).

6.363.3.3 `virtual void activemq::wireformat::openwire::marshal::generated-
::MessagePullMarshaller::looseMarshal (OpenWireFormat * format,
commands::DataStructure * command, decaf::io::DataOutputStream * ds
) [virtual]`

Tight Marshal to the given stream.

Parameters

<i>format</i>	- The OpenWireFormat properties
<i>command</i>	- the object to Marshal
<i>ds</i>	- DataOutputStream to marshal to

Exceptions

<i>IOException</i>	if an error occurs.
--------------------	---------------------

Reimplemented from **activemq::wireformat::openwire::marshal::generated::Base-
CommandMarshaller** (p. 500).

6.363.3.4 `virtual void activemq::wireformat::openwire::marshal::generated::-`
MessagePullMarshaller::looseUnmarshal (`OpenWireFormat * format,`
`commands::DataStructure * command, decaf::io::DataInputStream * dis`)
 [virtual]

Loose Un-marhsal to the given stream.

Parameters

<i>format</i>	- The OpenwireFormat properties
<i>command</i>	- the object to Un-Marshal
<i>dis</i>	- the DataInputStream to Un-Marshal from

Exceptions

<i>IOException</i>	if an error occurs.
--------------------	---------------------

Reimplemented from **activemq::wireformat::openwire::marshal::generated::Base-CommandMarshaller** (p. 501).

6.363.3.5 `virtual int activemq::wireformat::openwire::marshal::generated::-`
MessagePullMarshaller::tightMarshal1 (`OpenWireFormat * format,`
`commands::DataStructure * command, utils::BooleanStream * bs`)
 [virtual]

Tisht Marhsal to the given stream.

Parameters

<i>format</i>	The OpenwireFormat properties
<i>command</i>	The object to Marshal
<i>bs</i>	The boolean stream to marshal to.

Exceptions

<i>IOException</i>	if an error occurs.
--------------------	---------------------

Reimplemented from **activemq::wireformat::openwire::marshal::generated::Base-CommandMarshaller** (p. 503).

6.363.3.6 `virtual void activemq::wireformat::openwire::marshal::generated-`
MessagePullMarshaller::tightMarshal2 (`OpenWireFormat * format,`
`commands::DataStructure * command, decaf::io::DataOutputStream * ds,`
`utils::BooleanStream * bs`) [virtual]

Tight Marhsal to the given stream.

Parameters

<i>format</i>	- The OpenWireFormat properties
<i>command</i>	- the object to Marshal
<i>ds</i>	- the DataOutputStream to Marshal to
<i>bs</i>	- boolean stream to marshal to.

Exceptions

<i>IOException</i>	if an error occurs.
--------------------	---------------------

Reimplemented from **activemq::wireformat::openwire::marshal::generated::Base-CommandMarshaller** (p. 504).

6.363.3.7 virtual void **activemq::wireformat::openwire::marshal::generated::MessagePullMarshaller::tightUnmarshal** (**OpenWireFormat** * *format*, **commands::DataStructure** * *command*, **decaf::io::DataInputStream** * *dis*, **utils::BooleanStream** * *bs*) [virtual]

Tight Un-marhsal to the given stream.

Parameters

<i>format</i>	- The OpenWireFormat properties
<i>command</i>	- the object to Un-Marshal
<i>dis</i>	- the DataInputStream to Un-Marshal from
<i>bs</i>	- boolean stream to unmarshal from.

Exceptions

<i>IOException</i>	if an error occurs.
--------------------	---------------------

Reimplemented from **activemq::wireformat::openwire::marshal::generated::Base-CommandMarshaller** (p. 505).

The documentation for this class was generated from the following file:

- src/main/activemq/wireformat/openwire/marshal/generated/**MessagePullMarshaller.h**

6.364 activemq::transport::mock::MockTransport Class Reference

The **MockTransport** (p. 1948) defines a base level **Transport** (p. 2790) class that is intended to be used in place of an a regular protocol **Transport** (p. 2790) such as TCP.

```
#include <src/main/activemq/transport/mock/MockTransport.h>
```

Inheritance diagram for activemq::transport::mock::MockTransport:

Public Member Functions

- **MockTransport** (const **Pointer**< **wireformat::WireFormat** > &wireFormat, const **Pointer**< **ResponseBuilder** > &responseBuilder)
- virtual **~MockTransport** ()
- virtual void **fireCommand** (const **Pointer**< **Command** > &command)

Fires a Command back through this transport to its registered CommandListener if there is one.
- virtual void **fireException** (const **exceptions::ActiveMQException** &ex)

Fires a Exception back through this transport to its registered ExceptionListener if there is one.
- void **setResponseBuilder** (const **Pointer**< **ResponseBuilder** > &responseBuilder)

*Sets the **ResponseBuilder** (p. 2301) that this class uses to create Responses to - Commands sent.*
- virtual void **setOutgoingListener** (**TransportListener** *listener)

Sets a Listener that gets notified for every command that would have been sent by this transport to the Broker, this allows a client to verify that its messages are making it to the wire.
- **Pointer**< **wireformat::WireFormat** > **getWireFormat** () const

Gets the currently set WireFormat.
- virtual void **oneway** (const **Pointer**< **Command** > &command)

Sends a one-way command.
- virtual **Pointer**< **Response** > **request** (const **Pointer**< **Command** > &command)

Sends the given command to the broker and then waits for the response.
- virtual **Pointer**< **Response** > **request** (const **Pointer**< **Command** > &command, unsigned int timeout)

Sends the given command to the broker and then waits for the response.
- virtual void **setWireFormat** (const **Pointer**< **wireformat::WireFormat** > &wireFormat **AMQCPP_UNUSED**)
- virtual void **setTransportListener** (**TransportListener** *listener)

Sets the observer of asynchronous events from this transport.
- virtual **TransportListener** * **getTransportListener** () const

Gets the observer of asynchronous events from this transport.
- virtual void **start** ()

*Starts the **Transport** (p. 2790), the send methods of a **Transport** (p. 2790) will throw an exception if used before the **Transport** (p. 2790) is started.*
- virtual void **stop** ()

*Stops the **Transport** (p. 2790).*
- virtual void **close** ()

Closes this object and deallocates the appropriate resources.

- virtual **Transport * narrow** (const std::type_info &typeid)

*Narrows down a Chain of Transports to a specific **Transport** (p. 2790) to allow a higher level transport to skip intermediate Transports in certain circumstances.*

- virtual bool **isFaultTolerant** () const

*Is this **Transport** (p. 2790) fault tolerant, meaning that it will reconnect to a broker on disconnect.*

- virtual bool **isConnected** () const

*Is the **Transport** (p. 2790) Connected to its Broker.*

- virtual bool **isClosed** () const

*Has the **Transport** (p. 2790) been shutdown and no longer usable.*

- virtual std::string **getRemoteAddress** () const
- virtual void **reconnect** (const **decaf::net::URI** &uri AMQCPP_UNUSED)
- std::string **getName** () const
- void **setName** (const std::string &name)
- bool **isFailOnSendMessage** () const
- void **setFailOnSendMessage** (bool value)
- int **getNumSentMessageBeforeFail** () const
- void **setNumSentMessageBeforeFail** (int value)
- int **getNumSentMessages** () const
- void **setNumSentMessages** (int value)
- bool **isFailOnReceiveMessage** () const
- void **setFailOnReceiveMessage** (bool value)
- int **getNumReceivedMessageBeforeFail** () const
- void **setNumReceivedMessageBeforeFail** (int value)
- int **getNumReceivedMessages** () const
- void **setNumReceivedMessages** (int value)
- bool **isFailOnKeepAliveSends** () const
- void **setFailOnKeepAliveSends** (bool value)
- int **getNumSentKeepAlivesBeforeFail** () const
- void **setNumSentKeepAlivesBeforeFail** (int value)
- int **getNumSentKeepAlives** () const
- void **setNumSentKeepAlives** (int value)
- bool **isFailOnStart** () const
- void **setFailOnStart** (bool value)
- bool **isFailOnStop** () const
- void **setFailOnStop** (bool value)
- bool **isFailOnClose** () const
- void **setFailOnClose** (bool value)
- virtual bool **isReconnectSupported** () const
- virtual bool **isUpdateURIsSupported** () const
- virtual void **updateURIs** (bool rebalance AMQCPP_UNUSED, const **decaf::util::List**< **decaf::net::URI** > &uris AMQCPP_UNUSED)

Static Public Member Functions

- static **MockTransport** * **getInstance** ()

6.364.1 Detailed Description

The **MockTransport** (p. 1948) defines a base level **Transport** (p. 2790) class that is intended to be used in place of an a regular protocol **Transport** (p. 2790) such as TCP.

This **Transport** (p. 2790) assumes that it is the base **Transport** (p. 2790) in the - Transports stack, and destroys any Transports that are passed to it in its constructor.

This **Transport** (p. 2790) defines an Interface **ResponseBuilder** (p. 2301) which must be implemented by any protocol for which the **Transport** (p. 2790) is used to Emulate. The **Transport** (p. 2790) hands off all outbound commands to the **ResponseBuilder** (p. 2301) for processing, it is up to the builder to create appropriate responses and schedule any asynchronous messages that might result from a message sent to the Broker.

6.364.2 Constructor & Destructor Documentation

6.364.2.1 **activemq::transport::mock::MockTransport::MockTransport** (**const** **Pointer**< **wireformat::WireFormat** > & *wireFormat*, **const** **Pointer**< **ResponseBuilder** > & *responseBuilder*)

6.364.2.2 **virtual activemq::transport::mock::MockTransport::~MockTransport** ()
[inline, virtual]

6.364.3 Member Function Documentation

6.364.3.1 **virtual void activemq::transport::mock::MockTransport::close** ()
[virtual]

Closes this object and deallocates the appropriate resources.

The object is generally no longer usable after calling close.

Exceptions

<i>IOException</i>	if an error occurs while closing.
--------------------	-----------------------------------

Implements **decaf::io::Closeable** (p. 817).

6.364.3.2 **virtual void activemq::transport::mock::MockTransport::fireCommand** (**const** **Pointer**< **Command** > & *command*) [inline, virtual]

Fires a Command back through this transport to its registered CommandListener if there is one.

Parameters

<i>command</i>	- Command to send to the Listener.
----------------	------------------------------------

6.364.3.3 `virtual void activemq::transport::mock::MockTransport::fireException (const exceptions::ActiveMQException & ex) [inline, virtual]`

Fires a Exception back through this transport to its registered ExceptionListener if there is one.

Parameters

<i>ex</i>	The Exception that will be passed on the the Transport (p. 2790) listener.
-----------	---

6.364.3.4 `static MockTransport* activemq::transport::mock::MockTransport::getInstance () [inline, static]`

6.364.3.5 `std::string activemq::transport::mock::MockTransport::getName () const [inline]`

6.364.3.6 `int activemq::transport::mock::MockTransport::getNumReceivedMessageBeforeFail () const [inline]`

6.364.3.7 `int activemq::transport::mock::MockTransport::getNumReceivedMessages () const [inline]`

6.364.3.8 `int activemq::transport::mock::MockTransport::getNumSentKeepAlives () const [inline]`

6.364.3.9 `int activemq::transport::mock::MockTransport::getNumSentKeepAlivesBeforeFail () const [inline]`

6.364.3.10 `int activemq::transport::mock::MockTransport::getNumSentMessageBeforeFail () const [inline]`

6.364.3.11 `int activemq::transport::mock::MockTransport::getNumSentMessages () const [inline]`

6.364.3.12 `virtual std::string activemq::transport::mock::MockTransport::getRemoteAddress () const [inline, virtual]`

Returns

the remote address for this connection

Implements **activemq::transport::Transport** (p. 2792).

6.364.3.13 `virtual TransportListener* activemq::transport::mock::MockTransport::getTransportListener () const [inline, virtual]`

Gets the observer of asynchronous events from this transport.

Returns

the listener of transport events.

Implements **activemq::transport::Transport** (p. 2792).

6.364.3.14 `Pointer<wireformat::WireFormat> activemq::transport::mock::MockTransport::getWireFormat () const [inline, virtual]`

Gets the currently set WireFormat.

Returns

the current WireFormat object.

Implements **activemq::transport::Transport** (p. 2792).

6.364.3.15 `virtual bool activemq::transport::mock::MockTransport::isClosed () const [inline, virtual]`

Has the **Transport** (p. 2790) been shutdown and no longer usable.

Returns

true if the **Transport** (p. 2790)

Implements **activemq::transport::Transport** (p. 2793).

6.364.3.16 `virtual bool activemq::transport::mock::MockTransport::isConnected () const [inline, virtual]`

Is the **Transport** (p. 2790) Connected to its Broker.

Returns

true if a connection has been made.

Implements **activemq::transport::Transport** (p. 2793).

- 6.364.3.17 `bool activemq::transport::mock::MockTransport::isFailOnClose () const [inline]`
- 6.364.3.18 `bool activemq::transport::mock::MockTransport::isFailOnKeepAliveSends () const [inline]`
- 6.364.3.19 `bool activemq::transport::mock::MockTransport::isFailOnReceiveMessage () const [inline]`
- 6.364.3.20 `bool activemq::transport::mock::MockTransport::isFailOnSendMessage () const [inline]`
- 6.364.3.21 `bool activemq::transport::mock::MockTransport::isFailOnStart () const [inline]`
- 6.364.3.22 `bool activemq::transport::mock::MockTransport::isFailOnStop () const [inline]`
- 6.364.3.23 `virtual bool activemq::transport::mock::MockTransport::isFaultTolerant () const [inline, virtual]`

Is this **Transport** (p. 2790) fault tolerant, meaning that it will reconnect to a broker on disconnect.

Returns

true if the **Transport** (p. 2790) is fault tolerant.

Implements **activemq::transport::Transport** (p. 2793).

- 6.364.3.24 `virtual bool activemq::transport::mock::MockTransport::isReconnectSupported () const [inline, virtual]`

Returns

true if reconnect is supported.

Implements **activemq::transport::Transport** (p. 2794).

- 6.364.3.25 `virtual bool activemq::transport::mock::MockTransport::isUpdateURIsSupported () const [inline, virtual]`

Returns

true if updating uris is supported.

Implements **activemq::transport::Transport** (p. 2794).

6.364.3.26 `virtual Transport* activemq::transport::mock::MockTransport::narrow (const std::type_info & typeid) [inline, virtual]`

Narrows down a Chain of Transports to a specific **Transport** (p. 2790) to allow a higher level transport to skip intermediate Transports in certain circumstances.

Parameters

<i>typeid</i>	- The type_info of the Object we are searching for.
---------------	---

Returns

the requested Object. or NULL if its not in this chain.

Implements **activemq::transport::Transport** (p. 2794).

6.364.3.27 `virtual void activemq::transport::mock::MockTransport::oneway (const Pointer< Command > & command) [virtual]`

Sends a one-way command.

Does not wait for any response from the broker.

Parameters

<i>command</i>	The command to be sent.
----------------	-------------------------

Exceptions

<i>IOException</i>	if an exception occurs during writing of the command.
<i>Unsupported-OperationException</i>	if this method is not implemented by this transport.

Implements **activemq::transport::Transport** (p. 2795).

6.364.3.28 `virtual void activemq::transport::mock::MockTransport::reconnect (const decaf::net::URI &uri AMQCPP_UNUSED) [inline, virtual]`

6.364.3.29 `virtual Pointer<Response> activemq::transport::mock::MockTransport::request (const Pointer< Command > & command) [virtual]`

Sends the given command to the broker and then waits for the response.

Parameters

<i>command</i>	the command to be sent.
----------------	-------------------------

Returns

the response from the broker.

Exceptions

<i>IOException</i>	if an exception occurs during the read of the command.
<i>Unsupported-Operation</i>	if this method is not implemented by this transport.

Implements **activemq::transport::Transport** (p. 2795).

6.364.3.30 **virtual Pointer<Response> activemq::transport::mock::MockTransport::request (const Pointer< Command > & command, unsigned int timeout)** [virtual]

Sends the given command to the broker and then waits for the response.

Parameters

<i>command</i>	The command to be sent.
<i>timeout</i>	The time to wait for this response.

Returns

the response from the broker.

Exceptions

<i>IOException</i>	if an exception occurs during the read of the command.
<i>Unsupported-Operation</i>	if this method is not implemented by this transport.

Implements **activemq::transport::Transport** (p. 2796).

6.364.3.31 **void activemq::transport::mock::MockTransport::setFailOnClose (bool value)** [inline]

6.364.3.32 **void activemq::transport::mock::MockTransport::setFailOnKeepAliveSends (bool value)** [inline]

6.364.3.33 **void activemq::transport::mock::MockTransport::setFailOnReceiveMessage (bool value)** [inline]

6.364.3.34 **void activemq::transport::mock::MockTransport::setFailOnSendMessage (bool value)** [inline]

- 6.364.3.35 `void activemq::transport::mock::MockTransport::setFailOnStart (bool value) [inline]`
- 6.364.3.36 `void activemq::transport::mock::MockTransport::setFailOnStop (bool value) [inline]`
- 6.364.3.37 `void activemq::transport::mock::MockTransport::setName (const std::string & name) [inline]`
- 6.364.3.38 `void activemq::transport::mock::MockTransport::setNumReceivedMessageBeforeFail (int value) [inline]`
- 6.364.3.39 `void activemq::transport::mock::MockTransport::setNumReceivedMessages (int value) [inline]`
- 6.364.3.40 `void activemq::transport::mock::MockTransport::setNumSentKeepAlives (int value) [inline]`
- 6.364.3.41 `void activemq::transport::mock::MockTransport::setNumSentKeepAlivesBeforeFail (int value) [inline]`
- 6.364.3.42 `void activemq::transport::mock::MockTransport::setNumSentMessageBeforeFail (int value) [inline]`
- 6.364.3.43 `void activemq::transport::mock::MockTransport::setNumSentMessages (int value) [inline]`
- 6.364.3.44 `virtual void activemq::transport::mock::MockTransport::setOutgoingListener (TransportListener * listener) [inline, virtual]`

Sets a Listener that gets notified for every command that would have been sent by this transport to the Broker, this allows a client to verify that its messages are making it to the wire.

Parameters

<i>listener</i>	- The CommandListener to notify for each message
-----------------	--

- 6.364.3.45 `void activemq::transport::mock::MockTransport::setResponseBuilder (const Pointer< ResponseBuilder > & responseBuilder) [inline]`

Sets the **ResponseBuilder** (p.2301) that this class uses to create Responses to - Commands sent.

These are either real Response Objects, or Commands that would have been sent -

Asynchronously be the Broker.

Parameters

<i>response-Builder</i>	- The ResponseBuilder (p. 2301) to use from now on.
-------------------------	--

6.364.3.46 `virtual void activemq::transport::mock::MockTransport::setTransportListener (TransportListener * listener) [inline, virtual]`

Sets the observer of asynchronous events from this transport.

Parameters

<i>listener</i>	the listener of transport events.
-----------------	-----------------------------------

Implements **activemq::transport::Transport** (p. 2797).

6.364.3.47 `virtual void activemq::transport::mock::MockTransport::setWireFormat (const Pointer< wireformat::WireFormat > &wireFormat AMQCPP_UNUSED) [inline, virtual]`

6.364.3.48 `virtual void activemq::transport::mock::MockTransport::start () [virtual]`

Starts the **Transport** (p. 2790), the send methods of a **Transport** (p. 2790) will throw an exception if used before the **Transport** (p. 2790) is started.

Exceptions

<i>IOException</i>	if an error occurs while starting the Transport (p. 2790).
--------------------	---

Implements **activemq::transport::Transport** (p. 2797).

6.364.3.49 `virtual void activemq::transport::mock::MockTransport::stop () [virtual]`

Stops the **Transport** (p. 2790).

Exceptions

<i>IOException</i>	if an error occurs while stopping the transport.
--------------------	--

Implements **activemq::transport::Transport** (p. 2797).

```
6.364.3.50 virtual void activemq::transport::mock::MockTransport::updateURIs (
    bool rebalance AMQCPP_UNUSED, const decaf::util::List< decaf::net::URI >
    &uris AMQCPP_UNUSED ) [inline, virtual]
```

The documentation for this class was generated from the following file:

- src/main/activemq/transport/mock/**MockTransport.h**

6.365 activemq::transport::mock::MockTransportFactory Class - Reference

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

```
#include <src/main/activemq/transport/mock/MockTransport-
Factory.h>
```

Inheritance diagram for activemq::transport::mock::MockTransportFactory:

Public Member Functions

- virtual **~MockTransportFactory** ()
- virtual **Pointer< Transport > create** (const **decaf::net::URI** &location)
*Creates a fully configured **Transport** (p. 2790) instance which could be a chain of filters and transports.*
- virtual **Pointer< Transport > createComposite** (const **decaf::net::URI** &location)
*Creates a slimmed down **Transport** (p. 2790) instance which can be used in composite transport instances.*

Protected Member Functions

- virtual **Pointer< Transport > doCreateComposite** (const **decaf::net::URI** &location, const **Pointer< wireformat::WireFormat >** &wireFormat, const **decaf::util::Properties** &properties)
*Creates a slimmed down **Transport** (p. 2790) instance which can be used in composite transport instances.*

6.365.1 Detailed Description

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

6.365.2 Constructor & Destructor Documentation

6.365.2.1 virtual **activemq::transport::mock::MockTransportFactory::~MockTransportFactory** () [inline, virtual]

6.365.3 Member Function Documentation

6.365.3.1 virtual **Pointer<Transport>** **activemq::transport::mock::MockTransportFactory::create** (const decaf::net::URI & *location*) [virtual]

Creates a fully configured **Transport** (p. 2790) instance which could be a chain of filters and transports.

Parameters

<i>location</i>	- URI location to connect to plus any properties to assign.
-----------------	---

Exceptions

<i>ActiveMQException</i>	if an error occurs
--------------------------	--------------------

Implements **activemq::transport::TransportFactory** (p. 2799).

6.365.3.2 virtual **Pointer<Transport>** **activemq::transport::mock::MockTransportFactory::createComposite** (const decaf::net::URI & *location*) [virtual]

Creates a slimed down **Transport** (p. 2790) instance which can be used in composite transport instances.

Parameters

<i>location</i>	- URI location to connect to plus any properties to assign.
-----------------	---

Exceptions

<i>ActiveMQException</i>	if an error occurs
--------------------------	--------------------

Implements **activemq::transport::TransportFactory** (p. 2800).

6.365.3.3 `virtual Pointer<Transport> activemq::transport::mock::MockTransportFactory::doCreateComposite (const decaf::net::URI & location, const Pointer< wireformat::WireFormat > & wireFormat, const decaf::util::Properties & properties) [protected, virtual]`

Creates a slimmed down **Transport** (p. 2790) instance which can be used in composite transport instances.

Parameters

<i>location</i>	- URI location to connect to.
<i>wireFormat</i>	- the assigned WireFormat for the new Transport (p. 2790).
<i>properties</i>	- Properties to apply to the transport.

Returns

Pointer to a new **Transport** (p. 2790) instance.

Exceptions

<i>ActiveMQException</i>	if an error occurs
--------------------------	--------------------

The documentation for this class was generated from the following file:

- `src/main/activemq/transport/mock/MockTransportFactory.h`

6.366 decaf::util::concurrent::Mutex Class Reference

Mutex (p. 1960) object that offers recursive support on all platforms as well as providing the ability to use the standard wait / notify pattern used in languages like Java.

```
#include <src/main/decaf/util/concurrent/Mutex.h>
```

Inheritance diagram for `decaf::util::concurrent::Mutex`:

Public Member Functions

- **Mutex** ()
- **Mutex** (const std::string &name)
- virtual **~Mutex** ()
- std::string **getName** () const
- std::string **toString** () const
- virtual void **lock** ()
- *Locks the object.*
- virtual bool **tryLock** ()

Attempts to **Lock** (p. 1682) the object, if the lock is already held by another thread than this method returns false.

- virtual void **unlock** ()

Unlocks the object.

- virtual void **wait** ()

Waits on a signal from this object, which is generated by a call to Notify.

- virtual void **wait** (long long millisecs)

Waits on a signal from this object, which is generated by a call to Notify.

- virtual void **wait** (long long millisecs, int nanos)

Waits on a signal from this object, which is generated by a call to Notify.

- virtual void **notify** ()

Signals a waiter on this object that it can now wake up and continue.

- virtual void **notifyAll** ()

Signals the waiters on this object that it can now wake up and continue.

6.366.1 Detailed Description

Mutex (p. 1960) object that offers recursive support on all platforms as well as providing the ability to use the standard wait / notify pattern used in languages like Java.

Since

1.0

6.366.2 Constructor & Destructor Documentation

6.366.2.1 decaf::util::concurrent::Mutex::Mutex ()

6.366.2.2 decaf::util::concurrent::Mutex::Mutex (const std::string & name)

6.366.2.3 virtual decaf::util::concurrent::Mutex::~~Mutex () [virtual]

6.366.3 Member Function Documentation

6.366.3.1 std::string decaf::util::concurrent::Mutex::getName () const

6.366.3.2 virtual void decaf::util::concurrent::Mutex::lock () [virtual]

Locks the object.

Exceptions

<i>RuntimeException</i>	if an error occurs while locking the object.
-------------------------	--

Implements decaf::util::concurrent::Synchronizable (p. 2640).

Referenced by `decaf::util::StlQueue< T >::lock()`, `decaf::util::StlMap< std::string, cms::Topic * >::lock()`, `decaf::util::AbstractCollection< cms::Connection * >::lock()`, `decaf::util::concurrent::ConcurrentStlMap< Pointer< ProducerId >, Pointer< ProducerState >, ProducerId::COMPARATOR >::lock()`, and `decaf::util::concurrent::CopyOnWriteArrayList< ServiceListener * >::lock()`.

6.366.3.3 `virtual void decaf::util::concurrent::Mutex::notify ()` [virtual]

Signals a waiter on this object that it can now wake up and continue.

Must have this object locked before calling.

Exceptions

<i>IllegalMonitorStateException</i>	- if the current thread is not the owner of the the Synchronizable (p. 2639) Object.
<i>RuntimeException</i>	if an error occurs while notifying one of the waiting threads.

Implements **decaf::util::concurrent::Synchronizable** (p. 2641).

Referenced by `decaf::util::concurrent::LinkedBlockingQueue< E >::clear()`, `decaf::util::StlQueue< T >::notify()`, `decaf::util::StlMap< std::string, cms::Topic * >::notify()`, `decaf::util::AbstractCollection< cms::Connection * >::notify()`, `decaf::util::concurrent::ConcurrentStlMap< Pointer< ProducerId >, Pointer< ProducerState >, ProducerId::COMPARATOR >::notify()`, `decaf::util::concurrent::CopyOnWriteArrayList< ServiceListener * >::notify()`, `decaf::util::concurrent::LinkedBlockingQueue< E >::offer()`, `decaf::util::concurrent::LinkedBlockingQueue< E >::poll()`, `decaf::util::concurrent::LinkedBlockingQueue< E >::put()`, and `decaf::util::concurrent::LinkedBlockingQueue< E >::take()`.

6.366.3.4 `virtual void decaf::util::concurrent::Mutex::notifyAll ()` [virtual]

Signals the waiters on this object that it can now wake up and continue.

Must have this object locked before calling.

Exceptions

<i>IllegalMonitorStateException</i>	- if the current thread is not the owner of the the Synchronizable (p. 2639) Object.
<i>RuntimeException</i>	if an error occurs while notifying the waiting threads.

Implements **decaf::util::concurrent::Synchronizable** (p. 2642).

Referenced by `decaf::util::StlQueue< T >::notifyAll()`, `decaf::util::StlMap< std::string, cms::Topic * >::notifyAll()`, `decaf::util::AbstractCollection< cms::Connection * >::notifyAll()`, `decaf::util::concurrent::ConcurrentStlMap< Pointer< ProducerId >, Pointer< ProducerState >, ProducerId::COMPARATOR >::notifyAll()`, and `decaf::util::concurrent::CopyOnWriteArrayList< ServiceListener * >::notifyAll()`.

6.366.3.5 `std::string decaf::util::concurrent::Mutex::toString () const`

6.366.3.6 `virtual bool decaf::util::concurrent::Mutex::tryLock () [virtual]`

Attempts to **Lock** (p. 1682) the object, if the lock is already held by another thread than this method returns false.

Returns

true if the lock was acquired, false if it is already held by another thread.

Exceptions

<i>RuntimeException</i>	if an error occurs while locking the object.
-------------------------	--

Implements **decaf::util::concurrent::Synchronizable** (p. 2643).

Referenced by `decaf::util::StlQueue< T >::tryLock()`, `decaf::util::StlMap< std::string, cms::Topic * >::tryLock()`, `decaf::util::AbstractCollection< cms::Connection * >::tryLock()`, `decaf::util::concurrent::ConcurrentStlMap< Pointer< ProducerId >, Pointer< ProducerState >, ProducerId::COMPARATOR >::tryLock()`, and `decaf::util::concurrent::CopyOnWriteArrayList< ServiceListener * >::tryLock()`.

6.366.3.7 `virtual void decaf::util::concurrent::Mutex::unlock () [virtual]`

Unlocks the object.

Exceptions

<i>RuntimeException</i>	if an error occurs while unlocking the object.
-------------------------	--

Implements **decaf::util::concurrent::Synchronizable** (p. 2645).

Referenced by `decaf::util::StlQueue< T >::unlock()`, `decaf::util::StlMap< std::string, cms::Topic * >::unlock()`, `decaf::util::AbstractCollection< cms::Connection * >::unlock()`, `decaf::util::concurrent::ConcurrentStlMap< Pointer< ProducerId >, Pointer< ProducerState >, ProducerId::COMPARATOR >::unlock()`, and `decaf::util::concurrent::CopyOnWriteArrayList< ServiceListener * >::unlock()`.

6.366.3.8 `virtual void decaf::util::concurrent::Mutex::wait () [virtual]`

Waits on a signal from this object, which is generated by a call to `Notify`.

Must have this object locked before calling.

Exceptions

<i>RuntimeException</i>	if an error occurs while waiting on the object.
<i>InterruptedException</i>	if the wait is interrupted before it completes.

<i>IllegalMonitorStateException</i>	- if the current thread is not the owner of the the Synchronizable (p. 2639) Object.
-------------------------------------	---

Implements **decaf::util::concurrent::Synchronizable** (p. 2646).

Referenced by `decaf::util::concurrent::LinkedBlockingQueue< E >::offer()`, `decaf::util::concurrent::LinkedBlockingQueue< E >::poll()`, `decaf::util::concurrent::LinkedBlockingQueue< E >::put()`, `decaf::util::concurrent::LinkedBlockingQueue< E >::take()`, `decaf::util::StlQueue< T >::wait()`, `decaf::util::StlMap< std::string, cms::Topic * >::wait()`, `decaf::util::AbstractCollection< cms::Connection * >::wait()`, `decaf::util::concurrent::ConcurrentStlMap< Pointer< ProducerId >, Pointer< ProducerState >, ProducerId::COMPARATOR >::wait()`, and `decaf::util::concurrent::CopyOnWriteArrayList< ServiceListener * >::wait()`.

6.366.3.9 `virtual void decaf::util::concurrent::Mutex::wait (long long millisecs)`
[virtual]

Waits on a signal from this object, which is generated by a call to Notify.

Must have this object locked before calling. This wait will timeout after the specified time interval.

Parameters

<i>millisecs</i>	the time in milliseconds to wait, or WAIT_INFINITE
------------------	--

Exceptions

<i>RuntimeException</i>	if an error occurs while waiting on the object.
<i>InterruptedException</i>	if the wait is interrupted before it completes.
<i>IllegalMonitorStateException</i>	- if the current thread is not the owner of the the Synchronizable (p. 2639) Object.

Implements **decaf::util::concurrent::Synchronizable** (p. 2647).

6.366.3.10 `virtual void decaf::util::concurrent::Mutex::wait (long long millisecs, int nanos)` [virtual]

Waits on a signal from this object, which is generated by a call to Notify.

Must have this object locked before calling. This wait will timeout after the specified time interval. This method is similar to the one argument wait function except that it add a finer grained control over the amount of time that it waits by adding in the additional nanosecond argument.

NOTE: The ability to wait accurately at a nanosecond scale depends on the platform and OS that the Decaf API is running on, some systems do not provide an accurate enough clock to provide this level of granularity.

Parameters

<i>milliseconds</i>	the time in milliseconds to wait, or WAIT_INFINITE
<i>nanos</i>	additional time in nanoseconds with a range of 0-999999

Exceptions

<i>IllegalArgumentException</i>	if an error occurs or the nanos argument is not in the range of [0-999999]
<i>RuntimeException</i>	if an error occurs while waiting on the object.
<i>InterruptedException</i>	if the wait is interrupted before it completes.
<i>IllegalMonitorStateException</i>	- if the current thread is not the owner of the the Synchronizable (p. 2639) Object.

Implements **decaf::util::concurrent::Synchronizable** (p. 2648).

The documentation for this class was generated from the following file:

- src/main/decaf/util/concurrent/**Mutex.h**

6.367 decaf::util::concurrent::MutexHandle Class Reference

```
#include <src/main/decaf/internal/util/concurrent/unix/-
MutexHandle.h>
```

Public Member Functions

- **MutexHandle** ()
- **~MutexHandle** ()
- **MutexHandle** ()
- **~MutexHandle** ()

Data Fields

- pthread_mutex_t **mutex**
- volatile long long **lock_owner**
- volatile long long **lock_count**
- CRITICAL_SECTION **mutex**

6.367.1 Constructor & Destructor Documentation

6.367.1.1 decaf::util::concurrent::MutexHandle::MutexHandle () [inline]

6.367.1.2 decaf::util::concurrent::MutexHandle::~~MutexHandle () [inline]

6.367.1.3 **decaf::util::concurrent::MutexHandle::MutexHandle** () [inline]

6.367.1.4 **decaf::util::concurrent::MutexHandle::~~MutexHandle** () [inline]

6.367.2 Field Documentation

6.367.2.1 volatile long long **decaf::util::concurrent::MutexHandle::lock_count**

6.367.2.2 volatile long long **decaf::util::concurrent::MutexHandle::lock_owner**

6.367.2.3 CRITICAL_SECTION **decaf::util::concurrent::MutexHandle::mutex**

6.367.2.4 pthread_mutex_t **decaf::util::concurrent::MutexHandle::mutex**

The documentation for this class was generated from the following files:

- src/main/decaf/internal/util/concurrent/unix/**MutexHandle.h**
- src/main/decaf/internal/util/concurrent/windows/**MutexHandle.h**

6.368 decaf::internal::util::concurrent::MutexImpl Class Reference

```
#include <src/main/decaf/internal/util/concurrent/Mutex-Impl.h>
```

Static Public Member Functions

- static **decaf::util::concurrent::MutexHandle * create** ()
Creates a Reentrant Mutex and returns the handle, throws a Runtime Exception if the Mutex cannot be created for some reason.
- static void **destroy** (**decaf::util::concurrent::MutexHandle *handle**)
Destroy a previously create Mutex instance.
- static void **lock** (**decaf::util::concurrent::MutexHandle *handle**)
Locks the Mutex.
- static bool **trylock** (**decaf::util::concurrent::MutexHandle *handle**)
Tries to lock the Mutex.
- static void **unlock** (**decaf::util::concurrent::MutexHandle *handle**)
Unlocks the Mutex allowing other Thread to then acquire the Lock on it.

6.368.1 Member Function Documentation

6.368.1.1 static **decaf::util::concurrent::MutexHandle***
decaf::internal::util::concurrent::MutexImpl::create () [static]

Creates a Reentrant Mutex and returns the handle, throws a Runtime Exception if the Mutex cannot be created for some reason.

Returns

handle to a newly created Mutex.

6.368.1.2 `static void decaf::internal::util::concurrent::MutexImpl::destroy (decaf::util::concurrent::MutexHandle * handle) [static]`

Destroy a previously create Mutex instance.

Parameters

<i>mutex</i>	The Mutex instance to be destroyed.
--------------	-------------------------------------

6.368.1.3 `static void decaf::internal::util::concurrent::MutexImpl::lock (decaf::util::concurrent::MutexHandle * handle) [static]`

Locks the Mutex.

If the Mutex is already locked by another thread this method blocks until the Mutex becomes unlocked and this thread acquires the lock.

Parameters

<i>handle</i>	the handle to the Mutex to Lock.
---------------	----------------------------------

6.368.1.4 `static bool decaf::internal::util::concurrent::MutexImpl::trylock (decaf::util::concurrent::MutexHandle * handle) [static]`

Tries to lock the Mutex.

If the Mutex is unlocked this Thread acquires the lock on the Mutex and this method returns true, if the Mutex is already locked then the lock is not acquired and this method returns false.

Parameters

<i>handle</i>	the handle to the Mutex to attempt to Lock.
---------------	---

Returns

true if the lock was acquired false otherwise.

6.368.1.5 `static void decaf::internal::util::concurrent::MutexImpl::unlock (decaf::util::concurrent::MutexHandle * handle) [static]`

Unlocks the Mutex allowing other Thread to then acquire the Lock on it.

Parameters

<i>handle</i>	the handle to the Mutex to attempt to Lock.
---------------	---

The documentation for this class was generated from the following file:

- `src/main/decaf/internal/util/concurrent/MutexImpl.h`

6.369 decaf::internal::net::Network Class Reference

Internal class used to manage Networking related resources and hide platform dependent calls from the higher level API.

```
#include <src/main/decaf/internal/net/Network.h>
```

Public Member Functions

- virtual `~Network ()`
- `decaf::util::concurrent::Mutex * getRuntimeLock ()`
*Gets a pointer to the **Network** (p. 1968) Runtime's Lock object, this can be used by **Network** (p. 1968) layer APIs to synchronize around certain actions such as adding a resource to the **Network** (p. 1968) layer, etc.*
- void `addNetworkResource (decaf::internal::util::Resource *value)`
*Adds a Resource to the **Network** (p. 1968) Runtime, this resource will be held by the runtime until the Library shutdown method is called at which time all the Resources held by the **Network** (p. 1968) Runtime are destroyed.*
- `template<typename T >`
void `addAsResource (T *value)`
- void `addShutdownTask (decaf::lang::Runnable *task)`
*Register a Runnable to be called when the **Network** (p. 1968) Runtime is shutdown to provide a chance to cleanup any data or references that could cause problems should the **Network** (p. 1968) Runtime be re-initialized.*

Static Public Member Functions

- static `Network * getNetworkRuntime ()`
*Gets the one and only instance of the **Network** (p. 1968) class, if this is called before the **Network** (p. 1968) layer has been initialized or after it has been shutdown then an `IllegalStateException` is thrown.*
- static void `initializeNetworking ()`
Initialize the Networking layer.
- static void `shutdownNetworking ()`
*Shutdown the **Network** (p. 1968) layer and free any associated resources, classes in the Decaf library that use the networking layer will now fail if used after calling the shutdown method.*

Protected Member Functions

- **Network** ()

6.369.1 Detailed Description

Internal class used to manage Networking related resources and hide platform dependent calls from the higher level API.

Since

1.0

6.369.2 Constructor & Destructor Documentation

6.369.2.1 `decaf::internal::net::Network::Network ()` [protected]

6.369.2.2 `virtual decaf::internal::net::Network::~~Network ()` [virtual]

6.369.3 Member Function Documentation

6.369.3.1 `template<typename T> void decaf::internal::net::Network::addAsResource (T * value)` [inline]

6.369.3.2 `void decaf::internal::net::Network::addNetworkResource (decaf::internal::util::Resource * value)`

Adds a Resource to the **Network** (p. 1968) Runtime, this resource will be held by the runtime until the Library shutdown method is called at which time all the Resources held by the **Network** (p. 1968) Runtime are destroyed.

Parameters

<i>value</i>	The Resource to add to the Network (p. 1968) Runtime.
--------------	--

Exceptions

<i>NullPointerException</i>	if the Resource value passed is null.
-----------------------------	---------------------------------------

6.369.3.3 `void decaf::internal::net::Network::addShutdownTask (decaf::lang::Runnable * task)`

Register a Runnable to be called when the **Network** (p. 1968) Runtime is shutdown to provide a chance to cleanup any data or references that could cause problems should the **Network** (p. 1968) Runtime be re-initialized.

The Runnable pointer ownership is transferred to the NetworkRuntime to guarantee the

timing of resource cleanup.

The cleanup tasks are run at a critical time in the Shutdown process and should be as simple as possible and make every attempt to not throw any exceptions. If an exception is thrown it is ignored and processing of the next task is started.

The tasks should not assume that any **Network** (p. 1968) resources are still available and should execute as quickly as possible.

Parameters

<i>task</i>	Pointer to a Runnable object that will now be owned by the Network (p. 1968) Runtime.
-------------	--

6.369.3.4 `static Network* decaf::internal::net::Network::getNetworkRuntime ()`
[static]

Gets the one and only instance of the **Network** (p. 1968) class, if this is called before the **Network** (p. 1968) layer has been initialized or after it has been shutdown then an `IllegalStateException` is thrown.

Returns

pointer to the **Network** (p. 1968) runtime for the Decaf library.

6.369.3.5 `decaf::util::concurrent::Mutex* decaf::internal::net::Network::getRuntimeLock ()`

Gets a pointer to the **Network** (p. 1968) Runtime's Lock object, this can be used by **Network** (p. 1968) layer APIs to synchronize around certain actions such as adding a resource to the **Network** (p. 1968) layer, etc.

The pointer returned is owned by the **Network** (p. 1968) runtime and should not be deleted or copied by the caller.

Returns

a pointer to the **Network** (p. 1968) Runtime's single Lock instance.

6.369.3.6 `static void decaf::internal::net::Network::initializeNetworking ()`
[static]

Initialize the Networking layer.

6.369.3.7 static void decaf::internal::net::Network::shutdownNetworking ()
[static]

Shutdown the **Network** (p. 1968) layer and free any associated resources, classes in the Decaf library that use the networking layer will now fail if used after calling the shutdown method.

The documentation for this class was generated from the following file:

- src/main/decaf/internal/net/**Network.h**

6.370 activemq::commands::NetworkBridgeFilter Class Reference

```
#include <src/main/activemq/commands/NetworkBridgeFilter.h>
```

Inheritance diagram for activemq::commands::NetworkBridgeFilter:

Public Member Functions

- **NetworkBridgeFilter** ()
- virtual ~**NetworkBridgeFilter** ()
- virtual unsigned char **getDataStructureType** () const
*Get the **DataStructure** (p. 1133) Type as defined in CommandTypes.h.*
- virtual **NetworkBridgeFilter** * **cloneDataStructure** () const
Clone this object and return a new instance that the caller now owns, this will be an exact copy of this one.
- virtual void **copyDataStructure** (const **DataStructure** *src)
Copy the contents of the passed object into this objects members, overwriting any existing data.
- virtual std::string **toString** () const
*Returns a string containing the information for this **DataStructure** (p. 1133) such as its type and value of its elements.*
- virtual bool **equals** (const **DataStructure** *value) const
*Compares the **DataStructure** (p. 1133) passed in to this one, and returns if they are equivalent.*
- virtual int **getNetworkTTL** () const
- virtual void **setNetworkTTL** (int networkTTL)
- virtual const **Pointer**< **BrokerId** > & **getNetworkBrokerId** () const
- virtual **Pointer**< **BrokerId** > & **getNetworkBrokerId** ()
- virtual void **setNetworkBrokerId** (const **Pointer**< **BrokerId** > &networkBrokerId)

Static Public Attributes

- static const unsigned char **ID_NETWORKBRIDGEFILTER** = 91

Protected Attributes

- int **networkTTL**
- **Pointer**< **BrokerId** > **networkBrokerId**

6.370.1 Constructor & Destructor Documentation

6.370.1.1 **activemq::commands::NetworkBridgeFilter::NetworkBridgeFilter** ()

6.370.1.2 **virtual activemq::commands::NetworkBridgeFilter::~~NetworkBridgeFilter** () [virtual]

6.370.2 Member Function Documentation

6.370.2.1 **virtual NetworkBridgeFilter*** **activemq::commands::NetworkBridgeFilter::cloneDataStructure** () const [virtual]

Clone this obbect and return a new instance that the caller now owns, this will be an exact copy of this one.

Returns

new copy of this object.

Implements **activemq::commands::DataStructure** (p. 1133).

6.370.2.2 **virtual void** **activemq::commands::NetworkBridgeFilter::copyDataStructure** (const **DataStructure** * *src*) [virtual]

Copy the contents of the passed object into this objects members, overwriting any existing data.

Returns

src - Source Object

Implements **activemq::commands::DataStructure** (p. 1134).

6.370.2.3 `virtual bool activemq::commands::NetworkBridgeFilter::equals (const DataStructure * value) const [virtual]`

Compares the **DataStructure** (p. 1133) passed in to this one, and returns if they are equivalent.

Equivalent here means that they are of the same type, and that each element of the objects are the same.

Returns

true if **DataStructure** (p. 1133)'s are Equal.

Implements **activemq::commands::DataStructure** (p. 1135).

6.370.2.4 `virtual unsigned char activemq::commands::NetworkBridgeFilter::getDataStructureType () const [virtual]`

Get the **DataStructure** (p. 1133) Type as defined in CommandTypes.h.

Returns

The type of the data structure

Implements **activemq::commands::DataStructure** (p. 1137).

6.370.2.5 `virtual const Pointer<BrokerId>& activemq::commands::NetworkBridgeFilter::getNetworkBrokerId () const [virtual]`

6.370.2.6 `virtual Pointer<BrokerId>& activemq::commands::NetworkBridgeFilter::getNetworkBrokerId () [virtual]`

6.370.2.7 `virtual int activemq::commands::NetworkBridgeFilter::getNetworkTTL () const [virtual]`

6.370.2.8 `virtual void activemq::commands::NetworkBridgeFilter::setNetworkBrokerId (const Pointer< BrokerId > & networkBrokerId) [virtual]`

6.370.2.9 `virtual void activemq::commands::NetworkBridgeFilter::setNetworkTTL (int networkTTL) [virtual]`

6.370.2.10 `virtual std::string activemq::commands::NetworkBridgeFilter::toString () const [virtual]`

Returns a string containing the information for this **DataStructure** (p. 1133) such as its type and value of its elements.

Returns

formatted string useful for debugging.

Reimplemented from **activemq::commands::BaseDataStructure** (p. 531).

6.370.3 Field Documentation

6.370.3.1 `const unsigned char activemq::commands::NetworkBridgeFilter::ID_NETWORKBRIDGEFILTER = 91`
[static]

6.370.3.2 `Pointer<BrokerId> activemq::commands::NetworkBridgeFilter::networkBrokerId` [protected]

6.370.3.3 `int activemq::commands::NetworkBridgeFilter::networkTTL`
[protected]

The documentation for this class was generated from the following file:

- `src/main/activemq/commands/NetworkBridgeFilter.h`

6.371 activemq::wireformat::openwire::marshal::generated::NetworkBridgeFilterMarshaller Class Reference

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

```
#include <src/main/activemq/wireformat/openwire/marshal/generated/-  
NetworkBridgeFilterMarshaller.h>
```

Inheritance diagram for `activemq::wireformat::openwire::marshal::generated::NetworkBridgeFilterMarshaller`:

Public Member Functions

- **NetworkBridgeFilterMarshaller** ()
- virtual **~NetworkBridgeFilterMarshaller** ()
- virtual **commands::DataStructure * createObject** () const
Creates a new instance of the class that this class is a marshaling director for.
- virtual unsigned char **getDataStructureType** () const
Gets the DataStructureType that this class marshals/unmarshals.
- virtual void **tightUnmarshal** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataInputStream** *dataIn, **utils::BooleanStream** *bs)

6.371 activemq::wireformat::openwire::marshal::generated::NetworkBridgeFilterMarshaller Class

Reference

1981

Tight Un-marhsal to the given stream.

- virtual int **tightMarshal1** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **utils::BooleanStream** *bs)

Tight Marhsal to the given stream.

- virtual void **tightMarshal2** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataOutputStream** *dataOut, **utils::BooleanStream** *bs)

Tight Marhsal to the given stream.

- virtual void **looseUnmarshal** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataInputStream** *dataIn)

Loose Un-marhsal to the given stream.

- virtual void **looseMarshal** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataOutputStream** *dataOut)

Tight Marhsal to the given stream.

6.371.1 Detailed Description

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

NOTE!: This file is auto generated - do not modify! if you need to make a change, please see the Java Classes in the activemq-openwire-generator module

6.371.2 Constructor & Destructor Documentation

6.371.2.1 **activemq::wireformat::openwire::marshal::generated::NetworkBridgeFilterMarshaller::NetworkBridgeFilterMarshaller** ()
[inline]

6.371.2.2 **virtual activemq::wireformat::openwire::marshal::generated::NetworkBridgeFilterMarshaller::~~NetworkBridgeFilterMarshaller** ()
[inline, virtual]

6.371.3 Member Function Documentation

6.371.3.1 **virtual commands::DataStructure* activemq::wireformat::openwire::marshal::generated::NetworkBridgeFilterMarshaller::createObject** ()
const [virtual]

Creates a new instance of the class that this class is a marshaling director for.

Returns

newly allocated Command

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1120).

6.371.3.2 virtual unsigned char **activemq::wireformat::openwire::marshal::generated-
::NetworkBridgeFilterMarshaller::getDataStructureType ()** const
[virtual]

Gets the DataStructureType that this class marshals/unmarshals.

Returns

byte Id of this classes DataStructureType

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller**
(p. 1122).

6.371.3.3 virtual void **activemq::wireformat::openwire::marshal-
::generated::NetworkBridgeFilterMarshaller::looseMarshal (**
OpenWireFormat * format, commands::DataStructure * command,
decaf::io::DataOutputStream * ds) [virtual]

Tight Marshal to the given stream.

Parameters

<i>format</i>	- The OpenWireFormat properties
<i>command</i>	- the object to Marshal
<i>ds</i>	- DataOutputStream to marshal to

Exceptions

<i>IOException</i>	if an error occurs.
--------------------	---------------------

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller**
(p. 1123).

6.371.3.4 virtual void **activemq::wireformat::openwire::marshal-
::generated::NetworkBridgeFilterMarshaller::looseUnmarshal (**
OpenWireFormat * format, commands::DataStructure * command,
decaf::io::DataInputStream * dis) [virtual]

Loose Un-marshal to the given stream.

Parameters

<i>format</i>	- The OpenWireFormat properties
<i>command</i>	- the object to Un-Marshal
<i>dis</i>	- the DataInputStream to Un-Marshal from

6.371 activemq::wireformat::openwire::marshal::generated::NetworkBridgeFilterMarshaller Class

Reference

1983

Exceptions

<i>IOException</i>	if an error occurs.
--------------------	---------------------

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1125).

6.371.3.5 **virtual int activemq::wireformat::openwire::marshal::generated::NetworkBridgeFilterMarshaller::tightMarshal1 (OpenWireFormat * format, commands::DataStructure * command, utils::BooleanStream * bs) [virtual]**

Tight Marshal to the given stream.

Parameters

<i>format</i>	The OpenWireFormat properties
<i>command</i>	The object to Marshal
<i>bs</i>	The boolean stream to marshal to.

Exceptions

<i>IOException</i>	if an error occurs.
--------------------	---------------------

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1127).

6.371.3.6 **virtual void activemq::wireformat::openwire::marshal::generated::NetworkBridgeFilterMarshaller::tightMarshal2 (OpenWireFormat * format, commands::DataStructure * command, decaf::io::DataOutputStream * ds, utils::BooleanStream * bs) [virtual]**

Tight Marshal to the given stream.

Parameters

<i>format</i>	- The OpenWireFormat properties
<i>command</i>	- the object to Marshal
<i>ds</i>	- the DataOutputStream to Marshal to
<i>bs</i>	- boolean stream to marshal to.

Exceptions

<i>IOException</i>	if an error occurs.
--------------------	---------------------

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1129).

```

6.371.3.7 virtual void activemq::wireformat::openwire::marshal-
::generated::NetworkBridgeFilterMarshaller::tightUnmarshal (
OpenWireFormat * format, commands::DataStructure * command,
decaf::io::DataInputStream * dis, utils::BooleanStream * bs )
[virtual]

```

Tight Un-marhsal to the given stream.

Parameters

<i>format</i>	- The OpenwireFormat properties
<i>command</i>	- the object to Un-Marshal
<i>dis</i>	- the DataInputStream to Un-Marshal from
<i>bs</i>	- boolean stream to unmarshal from.

Exceptions

<i>IOException</i>	if an error occurs.
--------------------	---------------------

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1131).

The documentation for this class was generated from the following file:

- src/main/activemq/wireformat/openwire/marshal/generated/**NetworkBridgeFilterMarshaller.h**

6.372 decaf::net::NoRouteToHostException Class Reference

```
#include <src/main/decaf/net/NoRouteToHostException.h>
```

Inheritance diagram for decaf::net::NoRouteToHostException:

Public Member Functions

- **NoRouteToHostException** () throw ()
Default Constructor.
- **NoRouteToHostException** (const **Exception** &ex) throw ()
Conversion Constructor from some other Exception.
- **NoRouteToHostException** (const **NoRouteToHostException** &ex) throw ()
Copy Constructor.
- **NoRouteToHostException** (const char *file, const int lineNumber, const std::exception *cause, const char *msg,...) throw ()
Constructor - Initializes the file name and line number where this message occurred.

- **NoRouteToHostException** (const std::exception ***cause**) throw ()
Constructor.
- **NoRouteToHostException** (const char *file, const int lineNumber, const char *msg,...) throw ()
Constructor - Initializes the file name and line number where this message occurred.
- virtual **NoRouteToHostException** * **clone** () const
Clones this exception.
- virtual ~**NoRouteToHostException** () throw ()

6.372.1 Constructor & Destructor Documentation

6.372.1.1 **decaf::net::NoRouteToHostException::NoRouteToHostException ()**
throw () [inline]

Default Constructor.

6.372.1.2 **decaf::net::NoRouteToHostException::NoRouteToHostException (const Exception & ex)** throw () [inline]

Conversion Constructor from some other Exception.

Parameters

ex	An exception that should become this type of Exception
-----------	--

6.372.1.3 **decaf::net::NoRouteToHostException::NoRouteToHostException (const NoRouteToHostException & ex)** throw () [inline]

Copy Constructor.

Parameters

ex	An exception that should become this type of Exception
-----------	--

6.372.1.4 **decaf::net::NoRouteToHostException::NoRouteToHostException (const char * file, const int lineNumber, const std::exception * cause, const char * msg, ...)** throw () [inline]

Constructor - Initializes the file name and line number where this message occurred.

Sets the message to report, using an optional list of arguments to parse into the message

Parameters

<i>file</i>	The file name where exception occurs
<i>lineNumber</i>	The line number where the exception occurred.
<i>cause</i>	The exception that was the cause for this one to be thrown.
<i>msg</i>	The message to report
...	list of primitives that are formatted into the message

6.372.1.5 `decaf::net::NoRouteToHostException::NoRouteToHostException (const std::exception * cause) throw () [inline]`

Constructor.

Parameters

<i>cause</i>	Pointer to the exception that caused this one to be thrown, the object is cloned caller retains ownership.
--------------	--

6.372.1.6 `decaf::net::NoRouteToHostException::NoRouteToHostException (const char * file, const int lineNumber, const char * msg, ...) throw () [inline]`

Constructor - Initializes the file name and line number where this message occurred.

Sets the message to report, using an optional list of arguments to parse into the message

Parameters

<i>file</i>	The file name where exception occurs
<i>lineNumber</i>	The line number where the exception occurred.
<i>msg</i>	The message to report
...	list of primitives that are formatted into the message

6.372.1.7 `virtual decaf::net::NoRouteToHostException::~~NoRouteToHostException () throw () [inline, virtual]`

6.372.2 Member Function Documentation

6.372.2.1 `virtual NoRouteToHostException* decaf::net::NoRouteToHostException::clone () const [inline, virtual]`

Clones this exception.

This is useful for cases where you need to preserve the type of the original exception as well as the message. All subclasses should override.

Returns

a new Exception instance that is a copy of this Exception object.

Reimplemented from **decaf::net::SocketException** (p. 2473).

The documentation for this class was generated from the following file:

- src/main/decaf/net/**NoRouteToHostException.h**

6.373 decaf::security::NoSuchAlgorithmException Class Reference

```
#include <src/main/decaf/security/NoSuchAlgorithmException.h>
```

Inheritance diagram for decaf::security::NoSuchAlgorithmException:

Public Member Functions

- **NoSuchAlgorithmException** () throw ()
Default Constructor.
- **NoSuchAlgorithmException** (const **Exception** &ex) throw ()
Conversion Constructor from some other Exception.
- **NoSuchAlgorithmException** (const **NoSuchAlgorithmException** &ex) throw ()
Copy Constructor.
- **NoSuchAlgorithmException** (const char *file, const int lineNumber, const std::exception *cause, const char *msg,...) throw ()
Constructor - Initializes the file name and line number where this message occurred.
- **NoSuchAlgorithmException** (const std::exception *cause) throw ()
Constructor.
- **NoSuchAlgorithmException** (const char *file, const int lineNumber, const char *msg,...) throw ()
Constructor - Initializes the file name and line number where this message occurred.
- virtual **NoSuchAlgorithmException** * **clone** () const
Clones this exception.
- virtual ~**NoSuchAlgorithmException** () throw ()

6.373.1 Constructor & Destructor Documentation

6.373.1.1 decaf::security::NoSuchAlgorithmException::NoSuchAlgorithmException () throw () [inline]

Default Constructor.

6.373.1.2 **decaf::security::NoSuchAlgorithmException::NoSuchAlgorithmException** (const Exception & *ex*) throw ()
[inline]

Conversion Constructor from some other Exception.

Parameters

<i>ex</i>	An exception that should become this type of Exception
-----------	--

6.373.1.3 **decaf::security::NoSuchAlgorithmException::NoSuchAlgorithmException** (const NoSuchAlgorithmException & *ex*) throw ()
[inline]

Copy Constructor.

Parameters

<i>ex</i>	An exception that should become this type of Exception
-----------	--

6.373.1.4 **decaf::security::NoSuchAlgorithmException::NoSuchAlgorithmException** (const char * *file*, const int *lineNumber*, const std::exception * *cause*, const char * *msg*, ...) throw () [inline]

Constructor - Initializes the file name and line number where this message occurred.

Sets the message to report, using an optional list of arguments to parse into the message

Parameters

<i>file</i>	The file name where exception occurs
<i>lineNumber</i>	The line number where the exception occurred.
<i>cause</i>	The exception that was the cause for this one to be thrown.
<i>msg</i>	The message to report
...	list of primitives that are formatted into the message

6.373.1.5 **decaf::security::NoSuchAlgorithmException::NoSuchAlgorithmException** (const std::exception * *cause*) throw ()
[inline]

Constructor.

Parameters

<i>cause</i>	Pointer to the exception that caused this one to be thrown, the object is cloned caller retains ownership.
--------------	--

6.373.1.6 **decaf::security::NoSuchAlgorithmException::NoSuchAlgorithmException** (const char * *file*, const int *lineNumber*, const char * *msg*, ...) throw ()
[inline]

Constructor - Initializes the file name and line number where this message occurred.

Sets the message to report, using an optional list of arguments to parse into the message

Parameters

<i>file</i>	name where exception occurs
<i>lineNumber</i>	line number where the exception occurred.
<i>msg</i>	message to report
...	list of primitives that are formatted into the message

6.373.1.7 **virtual decaf::security::NoSuchAlgorithmException::~~NoSuchAlgorithmException** () throw () [inline, virtual]

6.373.2 Member Function Documentation

6.373.2.1 **virtual NoSuchAlgorithmException* decaf::security::NoSuchAlgorithmException::clone** () const [inline, virtual]

Clones this exception.

This is useful for cases where you need to preserve the type of the original exception as well as the message. All subclasses should override.

Returns

A deep copy of this exception.

Reimplemented from **decaf::security::GeneralSecurityException** (p. 1397).

The documentation for this class was generated from the following file:

- src/main/decaf/security/**NoSuchAlgorithmException.h**

6.374 decaf::util::NoSuchElementException Class Reference

```
#include <src/main/decaf/util/NoSuchElementException.h>
```

Inheritance diagram for decaf::util::NoSuchElementException:

Public Member Functions

- **NoSuchElementException** () throw ()
Default Constructor.
- **NoSuchElementException** (const **decaf::lang::exceptions::RuntimeException** &ex) throw ()
Conversion Constructor from some other Exception.
- **NoSuchElementException** (const **NoSuchElementException** &ex) throw ()
Copy Constructor.
- **NoSuchElementException** (const char *file, const int lineNumber, const std::exception *cause, const char *msg,...) throw ()
Constructor - Initializes the file name and line number where this message occurred.
- **NoSuchElementException** (const std::exception *cause) throw ()
Constructor.
- **NoSuchElementException** (const char *file, const int lineNumber, const char *msg,...) throw ()
Constructor - Initializes the file name and line number where this message occurred.
- virtual **NoSuchElementException** * **clone** () const
Clones this exception.
- virtual ~**NoSuchElementException** () throw ()

6.374.1 Constructor & Destructor Documentation

6.374.1.1 **decaf::util::NoSuchElementException::NoSuchElementException** () throw ()

Default Constructor.

6.374.1.2 **decaf::util::NoSuchElementException::NoSuchElementException** (const **decaf::lang::exceptions::RuntimeException** & ex) throw ()
[inline]

Conversion Constructor from some other Exception.

Parameters

ex	The Exception whose data is to be copied into this one.
----	---

6.374.1.3 **decaf::util::NoSuchElementException::NoSuchElementException** (const **NoSuchElementException** & ex) throw () [inline]

Copy Constructor.

Parameters

<i>ex</i>	The Exception whose data is to be copied into this one.
-----------	---

6.374.1.4 decaf::util::NoSuchElementException::NoSuchElementException (
const char * *file*, const int *lineNumber*, const std::exception * *cause*, const char * *msg*, ...) throw () `[inline]`

Constructor - Initializes the file name and line number where this message occurred.

Sets the message to report, using an optional list of arguments to parse into the message

Parameters

<i>file</i>	The file name where exception occurs
<i>lineNumber</i>	The line number where the exception occurred.
<i>cause</i>	The exception that was the cause for this one to be thrown.
<i>msg</i>	The message to report
...	list of primitives that are formatted into the message

6.374.1.5 decaf::util::NoSuchElementException::NoSuchElementException (
const std::exception * *cause*) throw () `[inline]`

Constructor.

Parameters

<i>cause</i>	Pointer to the exception that caused this one to be thrown, the object is cloned caller retains ownership.
--------------	--

6.374.1.6 decaf::util::NoSuchElementException::NoSuchElementException
(const char * *file*, const int *lineNumber*, const char * *msg*, ...) throw ()
`[inline]`

Constructor - Initializes the file name and line number where this message occurred.

Sets the message to report, using an optional list of arguments to parse into the message

Parameters

<i>file</i>	The file name where exception occurs
<i>lineNumber</i>	The line number where the exception occurred.
<i>msg</i>	The message to report
...	list of primitives that are formatted into the message

6.374.1.7 `virtual decaf::util::NoSuchElementException::~~NoSuchElementException () throw ()` [virtual]

6.374.2 Member Function Documentation

6.374.2.1 `virtual NoSuchElementException* decaf::util::NoSuchElementException::clone () const` [inline, virtual]

Clones this exception.

This is useful for cases where you need to preserve the type of the original exception as well as the message. All subclasses should override.

Returns

an new Exception instance that is a copy of this one.

Reimplemented from `decaf::lang::exceptions::RuntimeException` (p. 2317).

The documentation for this class was generated from the following file:

- `src/main/decaf/util/NoSuchElementException.h`

6.375 decaf::security::NoSuchProviderException Class Reference

```
#include <src/main/decaf/security/NoSuchProviderException.h>
```

Inheritance diagram for `decaf::security::NoSuchProviderException`:

Public Member Functions

- `NoSuchProviderException () throw ()`
Default Constructor.
- `NoSuchProviderException (const Exception &ex) throw ()`
Conversion Constructor from some other Exception.
- `NoSuchProviderException (const NoSuchElementException &ex) throw ()`
Copy Constructor.
- `NoSuchProviderException (const char *file, const int lineNumber, const std::exception *cause, const char *msg,...) throw ()`
Constructor - Initializes the file name and line number where this message occurred.
- `NoSuchProviderException (const std::exception *cause) throw ()`
Constructor.

- **NoSuchProviderException** (const char *file, const int lineNumber, const char *msg,...) throw ()
Constructor - Initializes the file name and line number where this message occurred.
- virtual **NoSuchProviderException** * clone () const
Clones this exception.
- virtual ~**NoSuchProviderException** () throw ()

6.375.1 Constructor & Destructor Documentation

6.375.1.1 **decaf::security::NoSuchProviderException::NoSuchProviderException**
() throw () [inline]

Default Constructor.

6.375.1.2 **decaf::security::NoSuchProviderException::NoSuchProviderException**
(const Exception & ex) throw () [inline]

Conversion Constructor from some other Exception.

Parameters

<i>ex</i>	An exception that should become this type of Exception
-----------	--

6.375.1.3 **decaf::security::NoSuchProviderException::NoSuchProviderException**
(const NoSuchProviderException & ex) throw () [inline]

Copy Constructor.

Parameters

<i>ex</i>	An exception that should become this type of Exception
-----------	--

6.375.1.4 **decaf::security::NoSuchProviderException::NoSuchProviderException**
(const char * file, const int lineNumber, const std::exception * cause, const char * msg, ...) throw () [inline]

Constructor - Initializes the file name and line number where this message occurred.

Sets the message to report, using an optional list of arguments to parse into the message

Parameters

<i>file</i>	The file name where exception occurs
<i>lineNumber</i>	The line number where the exception occurred.
<i>cause</i>	The exception that was the cause for this one to be thrown.

<i>msg</i>	The message to report
...	list of primitives that are formatted into the message

6.375.1.5 `decaf::security::NoSuchProviderException::NoSuchProviderException`
`(const std::exception * cause) throw () [inline]`

Constructor.

Parameters

<i>cause</i>	Pointer to the exception that caused this one to be thrown, the object is cloned caller retains ownership.
--------------	--

6.375.1.6 `decaf::security::NoSuchProviderException::NoSuchProviderException`
`(const char * file, const int lineNumber, const char * msg, ...) throw ()`
`[inline]`

Constructor - Initializes the file name and line number where this message occurred.

Sets the message to report, using an optional list of arguments to parse into the message

Parameters

<i>file</i>	name where exception occurs
<i>lineNumber</i>	line number where the exception occurred.
<i>msg</i>	message to report
...	list of primitives that are formatted into the message

6.375.1.7 `virtual decaf::security::NoSuchProviderException::~~NoSuchProviderException`
`() throw () [inline, virtual]`

6.375.2 Member Function Documentation

6.375.2.1 `virtual NoSuchProviderException* decaf::security::NoSuchProviderException::clone`
`() const [inline, virtual]`

Clones this exception.

This is useful for cases where you need to preserve the type of the original exception as well as the message. All subclasses should override.

Returns

A deep copy of this exception.

Reimplemented from **decaf::security::GeneralSecurityException** (p. 1397).

The documentation for this class was generated from the following file:

- src/main/decaf/security/**NoSuchProviderException.h**

6.376 decaf::lang::exceptions::NullPointerException Class - Reference

```
#include <src/main/decaf/lang/exceptions/NullPointerException-  
Exception.h>
```

Inheritance diagram for decaf::lang::exceptions::NullPointerException:

Public Member Functions

- **NullPointerException** () throw ()
Default Constructor.
- **NullPointerException** (const **Exception** &ex) throw ()
*Conversion Constructor from some other **Exception** (p. 1279).*
- **NullPointerException** (const **NullPointerException** &ex) throw ()
Copy Constructor.
- **NullPointerException** (const char *file, const int lineNumber, const std::exception *cause, const char *msg,...) throw ()
Constructor - Initializes the file name and line number where this message occurred.
- **NullPointerException** (const std::exception *cause) throw ()
Constructor.
- **NullPointerException** (const char *file, const int lineNumber, const char *msg,...) throw ()
Constructor - Initializes the file name and line number where this message occurred.
- virtual **NullPointerException** * **clone** () const
Clones this exception.
- virtual ~**NullPointerException** () throw ()

6.376.1 Constructor & Destructor Documentation

6.376.1.1 decaf::lang::exceptions::NullPointerException::NullPointerException () throw () [inline]

Default Constructor.

6.376.1.2 `decaf::lang::exceptions::NullPointerException::NullPointerException (const Exception & ex) throw () [inline]`

Conversion Constructor from some other **Exception** (p. 1279).

Parameters

<i>ex</i>	The Exception (p. 1279) whose data is to be copied into this one.
-----------	--

6.376.1.3 `decaf::lang::exceptions::NullPointerException::NullPointerException (const NullPointerException & ex) throw () [inline]`

Copy Constructor.

Parameters

<i>ex</i>	The Exception (p. 1279) whose data is to be copied into this one.
-----------	--

6.376.1.4 `decaf::lang::exceptions::NullPointerException::NullPointerException (const char * file, const int lineNumber, const std::exception * cause, const char * msg, ...) throw () [inline]`

Constructor - Initializes the file name and line number where this message occurred.

Sets the message to report, using an optional list of arguments to parse into the message

Parameters

<i>file</i>	The file name where exception occurs
<i>lineNumber</i>	The line number where the exception occurred.
<i>cause</i>	The exception that was the cause for this one to be thrown.
<i>msg</i>	The message to report
<i>...</i>	list of primitives that are formatted into the message

6.376.1.5 `decaf::lang::exceptions::NullPointerException::NullPointerException (const std::exception * cause) throw () [inline]`

Constructor.

Parameters

<i>cause</i>	Pointer (p.2083) to the exception that caused this one to be thrown, the object is cloned caller retains ownership.
--------------	--

6.376.1.6 `decaf::lang::exceptions::NullPointerException::NullPointerException`
 (`const char * file`, `const int lineNumber`, `const char * msg`, ...) `throw ()`
`[inline]`

Constructor - Initializes the file name and line number where this message occurred.

Sets the message to report, using an optional list of arguments to parse into the message

Parameters

<i>file</i>	The file name where exception occurs
<i>lineNumber</i>	The line number where the exception occurred.
<i>msg</i>	The message to report
...	list of primitives that are formatted into the message

6.376.1.7 `virtual decaf::lang::exceptions::NullPointerException-`
`::~NullPointerException () throw () [inline,`
`virtual]`

6.376.2 Member Function Documentation

6.376.2.1 `virtual NullPointerException* decaf::lang::exceptions-`
`::NullPointerException::clone () const [inline,`
`virtual]`

Clones this exception.

This is useful for cases where you need to preserve the type of the original exception as well as the message. All subclasses should override.

Returns

an new **Exception** (p. 1279) instance that is a copy of this one.

Reimplemented from **decaf::lang::Exception** (p. 1282).

The documentation for this class was generated from the following file:

- `src/main/decaf/lang/exceptions/NullPointerException.h`

6.377 decaf::lang::Number Class Reference

The abstract class **Number** (p. 1992) is the superclass of classes **Byte** (p. 614), **Double** (p. 1235), **Float** (p. 1344), **Integer** (p. 1500), **Long** (p. 1726), and **Short** (p. 2398).

```
#include <src/main/decaf/lang/Number.h>
```

Inheritance diagram for decaf::lang::Number:

Public Member Functions

- virtual **~Number** ()
- virtual unsigned char **byteValue** () const
Answers the byte value which the receiver represents.
- virtual double **doubleValue** () const =0
Answers the double value which the receiver represents.
- virtual float **floatValue** () const =0
Answers the float value which the receiver represents.
- virtual int **intValue** () const =0
Answers the int value which the receiver represents.
- virtual long long **longValue** () const =0
Answers the long value which the receiver represents.
- virtual short **shortValue** () const
Answers the short value which the receiver represents.

6.377.1 Detailed Description

The abstract class **Number** (p. 1992) is the superclass of classes **Byte** (p. 614), **Double** (p. 1235), **Float** (p. 1344), **Integer** (p. 1500), **Long** (p. 1726), and **Short** (p. 2398).

Subclasses of **Number** (p. 1992) must provide methods to convert the represented numeric value to byte, double, float, int, long, and short.

Since

1.0

6.377.2 Constructor & Destructor Documentation

6.377.2.1 virtual decaf::lang::Number::~~Number () [inline, virtual]

6.377.3 Member Function Documentation

6.377.3.1 virtual unsigned char decaf::lang::Number::byteValue () const [inline, virtual]

Answers the byte value which the receiver represents.

Returns

byte the value of the receiver.

Reimplemented in **decaf::lang::Double** (p. 1238), **decaf::lang::Float** (p. 1347), **decaf::lang::Integer** (p. 1503), **decaf::lang::Byte** (p. 616), **decaf::lang::Long** (p. 1729), **decaf::lang::Short** (p. 2400), and **decaf::lang::Character** (p. 767).

6.377.3.2 `virtual double decaf::lang::Number::doubleValue () const [pure virtual]`

Answers the double value which the receiver represents.

Returns

double the value of the receiver.

Implemented in **decaf::util::concurrent::atomic::AtomicInteger** (p. 478), **decaf::lang::Double** (p. 1241), **decaf::lang::Float** (p. 1348), **decaf::lang::Integer** (p. 1504), **decaf::lang::Byte** (p. 618), **decaf::lang::Long** (p. 1731), **decaf::lang::Short** (p. 2402), and **decaf::lang::Character** (p. 768).

6.377.3.3 `virtual float decaf::lang::Number::floatValue () const [pure virtual]`

Answers the float value which the receiver represents.

Returns

float the value of the receiver.

Implemented in **decaf::lang::Double** (p. 1242), **decaf::lang::Float** (p. 1350), **decaf::lang::Integer** (p. 1505), **decaf::lang::Byte** (p. 618), **decaf::lang::Long** (p. 1731), **decaf::util::concurrent::atomic::AtomicInteger** (p. 478), **decaf::lang::Short** (p. 2402), and **decaf::lang::Character** (p. 769).

6.377.3.4 `virtual int decaf::lang::Number::intValue () const [pure virtual]`

Answers the int value which the receiver represents.

Returns

int the value of the receiver.

Implemented in **decaf::lang::Double** (p. 1242), **decaf::lang::Float** (p. 1351), **decaf::lang::Integer** (p. 1506), **decaf::lang::Byte** (p. 618), **decaf::lang::Long** (p. 1732), **decaf::lang::Short** (p. 2403), **decaf::lang::Character** (p. 769), and **decaf::util::concurrent::atomic::AtomicInteger** (p. 480).

6.377.3.5 `virtual long long decaf::lang::Number::longValue () const [pure virtual]`

Answers the long value which the receiver represents.

Returns

long long the value of the receiver.

Implemented in **decaf::lang::Double** (p. 1243), **decaf::lang::Float** (p. 1352), **decaf::lang::Integer** (p. 1506), **decaf::lang::Byte** (p. 619), **decaf::lang::Long** (p. 1732), **decaf::lang::Short** (p. 2403), **decaf::lang::Character** (p. 770), and **decaf::util::concurrent::atomic::AtomicInteger** (p. 480).

6.377.3.6 `virtual short decaf::lang::Number::shortValue () const [inline, virtual]`

Answers the short value which the receiver represents.

Returns

short the value of the receiver.

Reimplemented in **decaf::lang::Double** (p. 1245), **decaf::lang::Float** (p. 1354), **decaf::lang::Integer** (p. 1511), **decaf::lang::Byte** (p. 621), **decaf::lang::Long** (p. 1738), **decaf::lang::Short** (p. 2406), and **decaf::lang::Character** (p. 772).

The documentation for this class was generated from the following file:

- `src/main/decaf/lang/Number.h`

6.378 decaf::lang::exceptions::NumberFormatException Class - Reference

```
#include <src/main/decaf/lang/exceptions/NumberFormatException.h>
```

Inheritance diagram for `decaf::lang::exceptions::NumberFormatException`:

Public Member Functions

- **NumberFormatException** ()
Default Constructor.
- **NumberFormatException** (const **Exception** &ex) throw ()
*Conversion Constructor from some other **Exception** (p. 1279).*

- **NumberFormatException** (const **NumberFormatException** &ex) throw ()
Copy Constructor.
- **NumberFormatException** (const char *file, const int lineNumber, const std::exception *cause, const char *msg,...) throw ()
Constructor - Initializes the file name and line number where this message occurred.
- **NumberFormatException** (const std::exception *cause) throw ()
Constructor.
- **NumberFormatException** (const char *file, const int lineNumber, const char *msg,...) throw ()
Constructor - Initializes the file name and line number where this message occurred.
- virtual **NumberFormatException** * clone () const
Clones this exception.
- virtual ~**NumberFormatException** () throw ()

6.378.1 Constructor & Destructor Documentation

6.378.1.1 decaf::lang::exceptions::NumberFormatException::NumberFormatException () [inline]

Default Constructor.

Referenced by clone().

6.378.1.2 decaf::lang::exceptions::NumberFormatException::NumberFormatException (const Exception & ex) throw () [inline]

Conversion Constructor from some other **Exception** (p. 1279).

Parameters

<i>ex</i>	The Exception (p. 1279) whose data is to be copied into this one.
-----------	--

6.378.1.3 decaf::lang::exceptions::NumberFormatException::NumberFormatException (const NumberFormatException & ex) throw () [inline]

Copy Constructor.

Parameters

<i>ex</i>	The Exception (p. 1279) whose data is to be copied into this one.
-----------	--

6.378.1.4 decaf::lang::exceptions::NumberFormatException::NumberFormatException (const char * *file*, const int *lineNumber*, const std::exception * *cause*, const char * *msg*, ...) throw () [inline]

Constructor - Initializes the file name and line number where this message occurred.

Sets the message to report, using an optional list of arguments to parse into the message

Parameters

<i>file</i>	The file name where exception occurs
<i>lineNumber</i>	The line number where the exception occurred.
<i>cause</i>	The exception that was the cause for this one to be thrown.
<i>msg</i>	The message to report
...	list of primitives that are formatted into the message

References decaf::lang::Exception::buildMessage(), and decaf::lang::Exception::set-Mark().

6.378.1.5 decaf::lang::exceptions::NumberFormatException::Number-FormatException (const std::exception * *cause*) throw () [inline]

Constructor.

Parameters

<i>cause</i>	Pointer (p. 2083) to the exception that caused this one to be thrown, the object is cloned caller retains ownership.
--------------	---

6.378.1.6 decaf::lang::exceptions::NumberFormatException::NumberFormatException (const char * *file*, const int *lineNumber*, const char * *msg*, ...) throw () [inline]

Constructor - Initializes the file name and line number where this message occurred.

Sets the message to report, using an optional list of arguments to parse into the message

Parameters

<i>file</i>	The file name where exception occurs
<i>lineNumber</i>	The line number where the exception occurred.
<i>msg</i>	The message to report
...	list of primitives that are formatted into the message

References decaf::lang::Exception::buildMessage(), and decaf::lang::Exception::set-Mark().

6.378.1.7 virtual decaf::lang::exceptions::NumberFormatException-
::~NumberFormatException () throw () [inline,
virtual]

6.378.2 Member Function Documentation

6.378.2.1 virtual NumberFormatException* decaf::lang::exceptions-
::NumberFormatException::clone () const [inline,
virtual]

Clones this exception.

This is useful for cases where you need to preserve the type of the original exception as well as the message. All subclasses should override.

Returns

an new **Exception** (p. 1279) instance that is a copy of this one.

Reimplemented from **decaf::lang::Exception** (p. 1282).

References NumberFormatException().

The documentation for this class was generated from the following file:

- src/main/decaf/lang/exceptions/**NumberFormatException.h**

6.379 cms::ObjectMessage Class Reference

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

```
#include <src/main/cms/ObjectMessage.h>
```

Inheritance diagram for cms::ObjectMessage:

Public Member Functions

- virtual ~**ObjectMessage** () throw ()

6.379.1 Detailed Description

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

serialized **ObjectMessage** (p. 1997) s.

Since

1.0

6.379.2 Constructor & Destructor Documentation

6.379.2.1 `virtual cms::ObjectMessage::~ObjectMessage () throw () [virtual]`

The documentation for this class was generated from the following file:

- `src/main/cms/ObjectMessage.h`

6.380 decaf::internal::net::ssl::openssl::OpenSSLContextSpi - Class Reference

Provides an SSLContext that wraps the OpenSSL API.

```
#include <src/main/decaf/internal/net/ssl/openssl/OpenSSLContextSpi.h>
```

Inheritance diagram for `decaf::internal::net::ssl::openssl::OpenSSLContextSpi`:

Public Member Functions

- `OpenSSLContextSpi ()`
- `virtual ~OpenSSLContextSpi ()`
- `virtual void providerInit (security::SecureRandom *random)`

Perform the initialization of this Context.

Parameters

random	Pointer to an instance of a secure random number generator.
--------	---

Exceptions

NullPointerException	if the SecureRandom instance is NULL.
KeyManagement-Exception	if an error occurs while initializing the context.

- `virtual decaf::net::SocketFactory * providerGetSocketFactory ()`

*Returns a **SocketFactory** (p. 2473) instance that can be used to create new **SSL-Socket** (p. 2513) objects.*

*The **SocketFactory** (p. 2473) is owned by the Service Provider and should not be destroyed by the caller.*

Returns

SocketFactory (p. 2473) instance that can be used to create new SSL Sockets.

6.380 decaf::internal::net::ssl::openssl::OpenSSLContextSpi Class Reference 2005

Exceptions

IllegalStateException	if the SSLContextSpi (p. 2498) object requires initialization but has not been initialized yet.
-----------------------	--

- virtual **decaf::net::ServerSocketFactory * providerGetServerSocketFactory**
()

Returns a **ServerSocketFactory** (p. 2352) instance that can be used to create new **SSLServerSocket** (p. 2504) objects.

The **ServerSocketFactory** (p. 2352) is owned by the Service Provider and should not be destroyed by the caller.

Returns

SocketFactory (p. 2473) instance that can be used to create new **SSLServerSockets**.

Exceptions

IllegalStateException	if the SSLContextSpi (p. 2498) object requires initialization but has not been initialized yet.
-----------------------	--

Friends

- class **OpenSSLSocket**
- class **OpenSSLSocketFactory**

6.380.1 Detailed Description

Provides an SSLContext that wraps the OpenSSL API.

Since

1.0

6.380.2 Constructor & Destructor Documentation

6.380.2.1 **decaf::internal::net::ssl::openssl::OpenSSLContextSpi::OpenSSLContextSpi** ()

6.380.2.2 **virtual decaf::internal::net::ssl::openssl::OpenSSLContextSpi::~~OpenSSLContextSpi** () [virtual]

6.380.3 Member Function Documentation

6.380.3.1 **virtual decaf::net::ServerSocketFactory* decaf::internal::net::ssl::openssl::OpenSSLContextSpi::providerGetServerSocketFactory** () [virtual]

Returns a **ServerSocketFactory** (p. 2352) instance that can be used to create new **SSLServerSocket** (p. 2504) objects.

The **ServerSocketFactory** (p. 2352) is owned by the Service Provider and should not be destroyed by the caller.

Returns

SocketFactory (p. 2473) instance that can be used to create new SSLServerSockets.

Exceptions

<i>IllegalStateException</i>	if the SSLContextSpi (p. 2498) object requires initialization but has not been initialized yet.
------------------------------	--

Implements **decaf::net::ssl::SSLContextSpi** (p. 2500).

```
6.380.3.2  virtual decaf::net::SocketFactory* decaf::internal::net::ssl-
           ::openssl::OpenSSLContextSpi::providerGetSocketFactory ( )
           [virtual]
```

Returns a **SocketFactory** (p. 2473) instance that can be used to create new **SSLSocket** (p. 2513) objects.

The **SocketFactory** (p. 2473) is owned by the Service Provider and should not be destroyed by the caller.

Returns

SocketFactory (p. 2473) instance that can be used to create new SSLSockets.

Exceptions

<i>IllegalStateException</i>	if the SSLContextSpi (p. 2498) object requires initialization but has not been initialized yet.
------------------------------	--

Implements **decaf::net::ssl::SSLContextSpi** (p. 2500).

```
6.380.3.3  virtual void decaf::internal::net::ssl::openssl::OpenSSL-
           ContextSpi::providerInit ( security::SecureRandom * random )
           [virtual]
```

Perform the initialization of this Context.

Parameters

<i>random</i>	Pointer to an instance of a secure random number generator.
---------------	---

6.381 decaf::internal::net::ssl::openssl::OpenSSLParameters Class Reference 2007

Exceptions

<i>NullPointerException</i>	if the SecureRandom instance is NULL.
<i>KeyManagementException</i>	if an error occurs while initializing the context.

Implements **decaf::net::ssl::SSLContextSpi** (p. 2501).

6.380.4 Friends And Related Function Documentation

6.380.4.1 friend class **OpenSSLSocket** [friend]

6.380.4.2 friend class **OpenSSLSocketFactory** [friend]

The documentation for this class was generated from the following file:

- src/main/decaf/internal/net/ssl/openssl/**OpenSSLContextSpi.h**

6.381 decaf::internal::net::ssl::openssl::OpenSSLParameters - Class Reference

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

```
#include <src/main/decaf/internal/net/ssl/openssl/OpenSSLParameters.h>
```

Public Member Functions

- virtual **~OpenSSLParameters** ()
- bool **getNeedClientAuth** () const
- void **setNeedClientAuth** (bool value)
- bool **getWantClientAuth** () const
- void **setWantClientAuth** (bool value)
- bool **getUseClientMode** () const
- void **setUseClientMode** (bool value)
- std::vector< std::string > **getSupportedCipherSuites** () const
- std::vector< std::string > **getSupportedProtocols** () const
- std::vector< std::string > **getEnabledCipherSuites** () const
- void **setEnabledCipherSuites** (const std::vector< std::string > &suites)
- std::vector< std::string > **getEnabledProtocols** () const
- void **setEnabledProtocols** (const std::vector< std::string > &protocols)
- **OpenSSLParameters * clone** () const

Creates a clone of this object such that all settings are transferred to a new instance of an SSL object whose parent is the same SSL_CTX as this object's.

6.381.1 Detailed Description

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

Since

1.0

6.381.2 Constructor & Destructor Documentation

6.381.2.1 `virtual decaf::internal::net::ssl::openssl::OpenSSLParameters::~~OpenSSLParameters () [virtual]`

6.381.3 Member Function Documentation

6.381.3.1 `OpenSSLParameters* decaf::internal::net::ssl::openssl::OpenSSLParameters::clone () const`

Creates a clone of this object such that all settings are transferred to a new instance of an SSL object whose parent is the same SSL_CTX as this object's.

6.381.3.2 `std::vector<std::string> decaf::internal::net::ssl::openssl::OpenSSLParameters::getEnabledCipherSuites () const`

6.381.3.3 `std::vector<std::string> decaf::internal::net::ssl::openssl::OpenSSLParameters::getEnabledProtocols () const`

6.381.3.4 `bool decaf::internal::net::ssl::openssl::OpenSSLParameters::getNeedClientAuth () const [inline]`

6.381.3.5 `std::vector<std::string> decaf::internal::net::ssl::openssl::OpenSSLParameters::getSupportedCipherSuites () const`

6.381.3.6 `std::vector<std::string> decaf::internal::net::ssl::openssl::OpenSSLParameters::getSupportedProtocols () const`

6.381.3.7 `bool decaf::internal::net::ssl::openssl::OpenSSLParameters::getUseClientMode () const [inline]`

6.381.3.8 `bool decaf::internal::net::ssl::openssl::OpenSSLParameters::getWantClientAuth () const [inline]`

6.382 `decaf::internal::net::ssl::openssl::OpenSSLServerSocket` Class Reference 2009

- 6.381.3.9 `void decaf::internal::net::ssl::openssl::OpenSSLParameters::setEnabledCipherSuites (const std::vector< std::string > & suites)`
- 6.381.3.10 `void decaf::internal::net::ssl::openssl::OpenSSLParameters::setEnabledProtocols (const std::vector< std::string > & protocols)`
- 6.381.3.11 `void decaf::internal::net::ssl::openssl::OpenSSLParameters::setNeedClientAuth (bool value)`
[inline]
- 6.381.3.12 `void decaf::internal::net::ssl::openssl::OpenSSLParameters::setUseClientMode (bool value)`
[inline]
- 6.381.3.13 `void decaf::internal::net::ssl::openssl::OpenSSLParameters::setWantClientAuth (bool value)`
[inline]

The documentation for this class was generated from the following file:

- `src/main/decaf/internal/net/ssl/openssl/OpenSSLParameters.h`

6.382 `decaf::internal::net::ssl::openssl::OpenSSLServerSocket` - Class Reference

SSLServerSocket based on OpenSSL library code.

```
#include <src/main/decaf/internal/net/ssl/openssl/OpenSSLServerSocket.h>
```

Inheritance diagram for `decaf::internal::net::ssl::openssl::OpenSSLServerSocket`:

Public Member Functions

- **OpenSSLServerSocket** (**OpenSSLParameters** *parameters)
- virtual **~OpenSSLServerSocket** ()
- virtual std::vector< std::string > **getSupportedCipherSuites** () const
*Gets a vector containing the names of all the cipher suites that are supported by this **SSLServerSocket** (p. 2504).
Normally not all of these cipher suites will be enabled on the **Socket** (p. 2452).*

Returns

a vector containing the names of all the supported cipher suites.

- virtual std::vector< std::string > **getSupportedProtocols** () const

*Gets a vector containing the names of all the protocols that could be enabled for this **SSLServerSocket** (p. 2504) instance.*

Returns

a vector containing the names of all the supported protocols.

- virtual std::vector< std::string > **getEnabledCipherSuites** () const

*Returns a vector containing the names of all the currently enabled Cipher Suites for this **SSLServerSocket** (p. 2504).*

Returns

vector of the names of all enabled Cipher Suites.

- virtual void **setEnabledCipherSuites** (const std::vector< std::string > &suites)

*Sets the Cipher Suites that are to be enabled on the **SSLServerSocket** (p. 2504) connection.*

Each of the named Cipher Suites must appear in the list of supported cipher suites for this connection or an exception will be thrown.

Parameters

suites	An Vector of names for all the Cipher Suites that are to be enabled.
--------	--

Exceptions

IllegalArgument-Exception	if the vector is empty or one of the names is invalid.
---------------------------	--

- virtual std::vector< std::string > **getEnabledProtocols** () const

*Returns a vector containing the names of all the currently enabled Protocols for this **SSLServerSocket** (p. 2504).*

Returns

vector of the names of all enabled Protocols.

- virtual void **setEnabledProtocols** (const std::vector< std::string > &protocols)

*Sets the Protocols that are to be enabled on the **SSLServerSocket** (p. 2504) connection.*

Each of the named Protocols must appear in the list of supported protocols suites for this connection or an exception will be thrown.

Parameters

protocols	An Vector of names for all the Protocols that are to be enabled.
-----------	--

Exceptions

IllegalArgument-Exception	if the vector is empty or one of the names is invalid.
---------------------------	--

- virtual bool **getWantClientAuth** () const

Returns

*true if the **Socket** (p. 2452) request client Authentication.*

- virtual void **setWantClientAuth** (bool value)

*Sets whether or not this **Socket** (p. 2452) will request Client Authentication.*

*If set to true the **Socket** (p. 2452) (when used in server mode) will request that the client authenticate itself, if the client doesn't send authentication the socket will still allow negotiation to continue.*

Parameters

value	<i>Whether the server socket should request client authentication.</i>
-------	--

- virtual bool **getNeedClientAuth** () const

Returns

*true if the **Socket** (p. 2452) requires client Authentication.*

- virtual void **setNeedClientAuth** (bool value)

*Sets whether or not this **Socket** (p. 2452) will require Client Authentication.*

*If set to true the **Socket** (p. 2452) (when used in server mode) will require that the client authenticate itself, if the client doesn't send authentication the socket will not allow negotiation to continue.*

Parameters

value	<i>Whether the server socket should require client authentication.</i>
-------	--

- virtual **decaf::net::Socket** * **accept** ()

*Listens for a connection request on the bound IPAddress and Port for this **ServerSocket** (p. 2342), the caller blocks until a connection is made.*

*If the **SO_TIMEOUT** option is set this method could throw a **SocketTimeoutException** (p. 2493) if the operation times out.*

Returns

*a new **Socket** (p. 2452) object pointer. Never returns NULL, the returned pointer is owned by the caller and must be explicitly freed by them.*

Exceptions

IOException	<i>if an I/O error occurs while binding the socket.</i>
SocketException (p. 2471)	<i>if an error occurs while blocking on the accept call.</i>
SocketTimeoutException (p. 2493)	<i>if the SO_TIMEOUT option was used and the accept timed out.</i>

6.382.1 Detailed Description

SSLServerSocket based on OpenSSL library code.

Since

1.0

6.382.2 Constructor & Destructor Documentation

6.382.2.1 `decaf::internal::net::ssl::openssl::OpenSSLServerSocket::OpenSSLServerSocket (OpenSSLParameters * parameters)`

6.382.2.2 `virtual decaf::internal::net::ssl::openssl::OpenSSLServerSocket::~~OpenSSLServerSocket ()`
[virtual]

6.382.3 Member Function Documentation

6.382.3.1 `virtual decaf::net::Socket* decaf::internal::net::ssl::openssl::OpenSSLServerSocket::accept ()`
[virtual]

Listens for a connection request on the bound IPAddress and Port for this **ServerSocket** (p. 2342), the caller blocks until a connection is made.

If the SO_TIMEOUT option is set this method could throw a **SocketTimeoutException** (p. 2493) if the operation times out.

Returns

a new **Socket** (p. 2452) object pointer. Never returns NULL, the returned pointer is owned by the caller and must be explicitly freed by them.

Exceptions

<i>IOException</i>	if an I/O error occurs while binding the socket.
SocketException (p. 2471)	if an error occurs while blocking on the accept call.
SocketTimeoutException (p. 2493)	if the SO_TIMEOUT option was used and the accept timed out.

Reimplemented from **decaf::net::ServerSocket** (p. 2346).

6.382.3.2 `virtual std::vector<std::string> decaf::internal::net::ssl::openssl::OpenSSLServerSocket::getEnabledCipherSuites () const`
[virtual]

Returns a vector containing the names of all the currently enabled Cipher Suites for this **SSLServerSocket** (p. 2504).

Returns

vector of the names of all enabled Cipher Suites.

Implements **decaf::net::ssl::SSLServerSocket** (p. 2508).

6.382 decaf::internal::net::ssl::openssl::OpenSSLServerSocket Class Reference 2013

6.382.3.3 `virtual std::vector<std::string> decaf::internal::net::ssl::openssl::OpenSSLServerSocket::getEnabledProtocols () const`
[virtual]

Returns a vector containing the names of all the currently enabled Protocols for this **SSLServerSocket** (p. 2504).

Returns

vector of the names of all enabled Protocols.

Implements **decaf::net::ssl::SSLServerSocket** (p. 2508).

6.382.3.4 `virtual bool decaf::internal::net::ssl::openssl::OpenSSLServerSocket::getNeedClientAuth () const`
[virtual]

Returns

true if the **Socket** (p. 2452) requires client Authentication.

Implements **decaf::net::ssl::SSLServerSocket** (p. 2508).

6.382.3.5 `virtual std::vector<std::string> decaf::internal::net::ssl::openssl::OpenSSLServerSocket::getSupportedCipherSuites () const`
[virtual]

Gets a vector containing the names of all the cipher suites that are supported by this **SSLServerSocket** (p. 2504).

Normally not all of these cipher suites will be enabled on the **Socket** (p. 2452).

Returns

a vector containing the names of all the supported cipher suites.

Implements **decaf::net::ssl::SSLServerSocket** (p. 2508).

6.382.3.6 `virtual std::vector<std::string> decaf::internal::net::ssl::openssl::OpenSSLServerSocket::getSupportedProtocols () const`
[virtual]

Gets a vector containing the names of all the protocols that could be enabled for this **SSLServerSocket** (p. 2504) instance.

Returns

a vector containing the names of all the supported protocols.

Implements **decaf::net::ssl::SSLServerSocket** (p. 2509).

6.382.3.7 virtual bool **decaf::internal::net::ssl::openssl::OpenSSLServerSocket::getWantClientAuth** () const
[virtual]

Returns

true if the **Socket** (p. 2452) request client Authentication.

Implements **decaf::net::ssl::SSLServerSocket** (p. 2509).

6.382.3.8 virtual void **decaf::internal::net::ssl::openssl::OpenSSLServerSocket::setEnabledCipherSuites** (const std::vector< std::string > & *suites*)
[virtual]

Sets the Cipher Suites that are to be enabled on the **SSLServerSocket** (p. 2504) connection.

Each of the named Cipher Suites must appear in the list of supported cipher suites for this connection or an exception will be thrown.

Parameters

<i>suites</i>	An Vector of names for all the Cipher Suites that are to be enabled.
---------------	--

Exceptions

<i>IllegalArgument-Exception</i>	if the vector is empty or one of the names is invalid.
----------------------------------	--

Implements **decaf::net::ssl::SSLServerSocket** (p. 2509).

6.382.3.9 virtual void **decaf::internal::net::ssl::openssl::OpenSSLServerSocket::setEnabledProtocols** (const std::vector< std::string > & *protocols*)
[virtual]

Sets the Protocols that are to be enabled on the **SSLServerSocket** (p. 2504) connection.

Each of the named Protocols must appear in the list of supported protocols suites for this connection or an exception will be thrown.

Parameters

<i>protocols</i>	An Vector of names for all the Protocols that are to be enabled.
------------------	--

Exceptions

<i>IllegalArgument-Exception</i>	if the vector is empty or one of the names is invalid.
----------------------------------	--

6.383 decaf::internal::net::ssl::openssl::OpenSSLServerSocketFactory Class Reference 2015

Implements **decaf::net::ssl::SSLServerSocket** (p. 2509).

6.382.3.10 virtual void decaf::internal::net::ssl::openssl::OpenSSLServerSocket::setNeedClientAuth (bool *value*)
[virtual]

Sets whether or not this **Socket** (p. 2452) will require Client Authentication.

If set to true the **Socket** (p. 2452) (when used in server mode) will require that the client authenticate itself, if the client doesn't send authentication the socket will not allow negotiation to continue.

Parameters

<i>value</i>	Whether the server socket should require client authentication.
--------------	---

Implements **decaf::net::ssl::SSLServerSocket** (p. 2510).

6.382.3.11 virtual void decaf::internal::net::ssl::openssl::OpenSSLServerSocket::setWantClientAuth (bool *value*)
[virtual]

Sets whether or not this **Socket** (p. 2452) will request Client Authentication.

If set to true the **Socket** (p. 2452) (when used in server mode) will request that the client authenticate itself, if the client doesn't send authentication the socket will still allow negotiation to continue.

Parameters

<i>value</i>	Whether the server socket should request client authentication.
--------------	---

Implements **decaf::net::ssl::SSLServerSocket** (p. 2510).

The documentation for this class was generated from the following file:

- src/main/decaf/internal/net/ssl/openssl/**OpenSSLServerSocket.h**

6.383 decaf::internal::net::ssl::openssl::OpenSSLServerSocketFactory Class Reference

SSLServerSocketFactory that creates Server Sockets that use OpenSSL.

```
#include <src/main/decaf/internal/net/ssl/openssl/OpenSSLServerSocketFactory.h>
```

Inheritance diagram for decaf::internal::net::ssl::openssl::OpenSSLServerSocket-

Factory:

Public Member Functions

- **OpenSSLServerSocketFactory** (**OpenSSLContextSpi** *parent)
- virtual ~**OpenSSLServerSocketFactory** ()
- virtual **decaf::net::ServerSocket** * **createServerSocket** ()

Create a new **ServerSocket** (p. 2342) that is unbound.

The **ServerSocket** (p. 2342) will have been configured with the defaults from the factory.

Returns

new **ServerSocket** (p. 2342) pointer that is owned by the caller.

Exceptions

IOException	if the ServerSocket (p. 2342) cannot be created for some reason.
-------------	---

- virtual **decaf::net::ServerSocket** * **createServerSocket** (int port)

Create a new **ServerSocket** (p. 2342) that is bound to the given port.

The **ServerSocket** (p. 2342) will have been configured with the defaults from the factory.

Parameters

port	The port to bind the ServerSocket (p. 2342) to.
------	--

Returns

new **ServerSocket** (p. 2342) pointer that is owned by the caller.

Exceptions

IOException	if the ServerSocket (p. 2342) cannot be created for some reason.
-------------	---

- virtual **decaf::net::ServerSocket** * **createServerSocket** (int port, int backlog)

Create a new **ServerSocket** (p. 2342) that is bound to the given port.

The **ServerSocket** (p. 2342) will have been configured with the defaults from the factory. The **ServerSocket** (p. 2342) will use the specified connection backlog setting.

Parameters

port	The port to bind the ServerSocket (p. 2342) to.
backlog	The number of pending connect request the ServerSocket (p. 2342) can queue.

Returns

new **ServerSocket** (p. 2342) pointer that is owned by the caller.

Exceptions

IOException	if the ServerSocket (p. 2342) cannot be created for some reason.
-------------	---

- virtual **decaf::net::ServerSocket** * **createServerSocket** (int port, int backlog, const **decaf::net::InetAddress** *address)

Create a new **ServerSocket** (p. 2342) that is bound to the given port.

6.383 decaf::internal::net::ssl::openssl::OpenSSLServerSocketFactory Class Reference 2017

The **ServerSocket** (p. 2342) will have been configured with the defaults from the factory. The **ServerSocket** (p. 2342) will bind to the specified interface on the local host, and accept connections only on that interface. If the address parameter is NULL than the **ServerSocket** (p. 2342) will listen on all interfaces.

Parameters

port	The port to bind the ServerSocket (p. 2342) to.
backlog	The number of pending connect request the ServerSocket (p. 2342) can queue.
address	The address of the interface on the local machine to bind to.

Returns

new **ServerSocket** (p. 2342) pointer that is owned by the caller.

Exceptions

IOException	if the ServerSocket (p. 2342) cannot be created for some reason.
-------------	---

- virtual std::vector< std::string > **getDefaultCipherSuites** ()

Returns the list of cipher suites which are enabled by default.
Unless a different list is enabled, handshaking on an SSL connection will use one of these cipher suites. The minimum quality of service for these defaults requires confidentiality protection and server authentication (that is, no anonymous cipher suites).

Returns

an STL vector containing the list of cipher suites enabled by default.

See also

getSupportedCipherSuites() (p. 2512)

- virtual std::vector< std::string > **getSupportedCipherSuites** ()

Returns the names of the cipher suites which could be enabled for use on an SSL connection.

Normally, only a subset of these will actually be enabled by default, since this list may include cipher suites which do not meet quality of service requirements for those defaults. Such cipher suites are useful in specialized applications.

Returns

an STL vector containing the list of supported cipher suites.

See also

getDefaultCipherSuites() (p. 2512)

6.383.1 Detailed Description

SSLServerSocketFactory that creates Server Sockets that use OpenSSL.

Since

1.0

6.383.2 Constructor & Destructor Documentation

6.383.2.1 `decaf::internal::net::ssl::openssl::OpenSSLServerSocketFactory::-`
`OpenSSLServerSocketFactory (OpenSSLContextSpi * parent`
`)`

6.383.2.2 `virtual decaf::internal::net::ssl::openssl::OpenSSLServer-`
`SocketFactory::~~OpenSSLServerSocketFactory ()`
`[virtual]`

6.383.3 Member Function Documentation

6.383.3.1 `virtual decaf::net::ServerSocket* decaf::internal::net::ssl::openssl-`
`::OpenSSLServerSocketFactory::createServerSocket ()`
`[virtual]`

Create a new **ServerSocket** (p. 2342) that is unbound.

The **ServerSocket** (p. 2342) will have been configured with the defaults from the factory.

Returns

new **ServerSocket** (p. 2342) pointer that is owned by the caller.

Exceptions

<i>IOException</i>	if the ServerSocket (p. 2342) cannot be created for some reason.
--------------------	---

Reimplemented from **decaf::net::ServerSocketFactory** (p. 2353).

6.383.3.2 `virtual decaf::net::ServerSocket* decaf::internal::net::ssl::openssl-`
`::OpenSSLServerSocketFactory::createServerSocket (int port)`
`[virtual]`

Create a new **ServerSocket** (p. 2342) that is bound to the given port.

The **ServerSocket** (p. 2342) will have been configured with the defaults from the factory.

Parameters

<i>port</i>	The port to bind the ServerSocket (p. 2342) to.
-------------	--

Returns

new **ServerSocket** (p. 2342) pointer that is owned by the caller.

Exceptions

<i>IOException</i>	if the ServerSocket (p. 2342) cannot be created for some reason.
--------------------	---

Implements **decaf::net::ServerSocketFactory** (p. 2353).

6.383 decaf::internal::net::ssl::openssl::OpenSSLServerSocketFactory Class Reference 2019

6.383.3.3 virtual **decaf::net::ServerSocket*** **decaf::internal::net::ssl::openssl::OpenSSLServerSocketFactory::createServerSocket** (int *port*, int *backlog*) [virtual]

Create a new **ServerSocket** (p. 2342) that is bound to the given port.

The **ServerSocket** (p. 2342) will have been configured with the defaults from the factory.
The **ServerSocket** (p. 2342) will use the specified connection backlog setting.

Parameters

<i>port</i>	The port to bind the ServerSocket (p. 2342) to.
<i>backlog</i>	The number of pending connect request the ServerSocket (p. 2342) can queue.

Returns

new **ServerSocket** (p. 2342) pointer that is owned by the caller.

Exceptions

<i>IOException</i>	if the ServerSocket (p. 2342) cannot be created for some reason.
--------------------	---

Implements **decaf::net::ServerSocketFactory** (p. 2354).

6.383.3.4 virtual **decaf::net::ServerSocket*** **decaf::internal::net::ssl::openssl::OpenSSLServerSocketFactory::createServerSocket** (int *port*, int *backlog*, const **decaf::net::InetAddress** * *address*) [virtual]

Create a new **ServerSocket** (p. 2342) that is bound to the given port.

The **ServerSocket** (p. 2342) will have been configured with the defaults from the factory.
The **ServerSocket** (p. 2342) will bind to the specified interface on the local host, and accept connections only on that interface. If the address parameter is NULL than the **ServerSocket** (p. 2342) will listen on all interfaces.

Parameters

<i>port</i>	The port to bind the ServerSocket (p. 2342) to.
<i>backlog</i>	The number of pending connect request the ServerSocket (p. 2342) can queue.
<i>address</i>	The address of the interface on the local machine to bind to.

Returns

new **ServerSocket** (p. 2342) pointer that is owned by the caller.

Exceptions

<i>IOException</i> if the ServerSocket (p. 2342) cannot be created for some reason.
--

Implements **decaf::net::ServerSocketFactory** (p. 2354).

6.383.3.5 `virtual std::vector<std::string> decaf::internal::net::ssl::openssl::OpenSSLServerSocketFactory::getDefaultCipherSuites ()`
`[virtual]`

Returns the list of cipher suites which are enabled by default.

Unless a different list is enabled, handshaking on an SSL connection will use one of these cipher suites. The minimum quality of service for these defaults requires confidentiality protection and server authentication (that is, no anonymous cipher suites).

Returns

an STL vector containing the list of cipher suites enabled by default.

See also

getSupportedCipherSuites() (p. 2512)

Implements **decaf::net::ssl::SSLServerSocketFactory** (p. 2512).

6.383.3.6 `virtual std::vector<std::string> decaf::internal::net::ssl::openssl::OpenSSLServerSocketFactory::getSupportedCipherSuites ()`
`[virtual]`

Returns the names of the cipher suites which could be enabled for use on an SSL connection.

Normally, only a subset of these will actually be enabled by default, since this list may include cipher suites which do not meet quality of service requirements for those defaults. Such cipher suites are useful in specialized applications.

Returns

an STL vector containing the list of supported cipher suites.

See also

getDefaultCipherSuites() (p. 2512)

Implements **decaf::net::ssl::SSLServerSocketFactory** (p. 2512).

The documentation for this class was generated from the following file:

- `src/main/decaf/internal/net/ssl/openssl/OpenSSLServerSocketFactory.h`

6.384 decaf::internal::net::ssl::openssl::OpenSSLSocket Class - Reference

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

```
#include <src/main/decaf/internal/net/ssl/openssl/OpenSSL-
LSocket.h>
```

Inheritance diagram for decaf::internal::net::ssl::openssl::OpenSSLSocket:

Public Member Functions

- **OpenSSLSocket** (**OpenSSLParameters** *parameters)
- **OpenSSLSocket** (**OpenSSLParameters** *parameters, const **decaf::net::InetAddress** *address, int port)
- **OpenSSLSocket** (**OpenSSLParameters** *parameters, const **decaf::net::InetAddress** *address, int port, const **decaf::net::InetAddress** *localAddress, int localPort)
- **OpenSSLSocket** (**OpenSSLParameters** *parameters, const std::string &host, int port)
- **OpenSSLSocket** (**OpenSSLParameters** *parameters, const std::string &host, int port, const **decaf::net::InetAddress** *localAddress, int localPort)
- virtual ~**OpenSSLSocket** ()
- virtual void **connect** (const std::string &host, int port, int timeout)

Connects to the specified destination, with a specified timeout value.

*If a connection to the remote host is not established within the specified timeout interval than an **SocketTimeoutException** (p. 2493) is thrown. A timeout value of zero is treated as an infinite timeout.*

Parameters

host	<i>The host name or IP address of the remote host to connect to.</i>
port	<i>The port on the remote host to connect to.</i>
timeout	<i>The number of Milliseconds to wait before treating the connection as failed.</i>

Exceptions

IOException	<i>Thrown if a failure occurred in the connect.</i>
SocketTimeoutException (p. 2493)	<i>if the timeout for connection is exceeded.</i>
IllegalArgument-Exception	<i>if the timeout value is negative or the endpoint is invalid.</i>

- virtual void **close** ()

*Closes the **Socket** (p. 2452).*

*Once closed a **Socket** (p. 2452) cannot be connected or otherwise operated upon, a new **Socket** (p. 2452) instance must be created.*

Exceptions

IOException	if an I/O error occurs while closing the Socket (p. 2452).
-------------	---

- virtual **decaf::io::InputStream * getInputStream ()**

Gets the InputStream for this socket if its connected.

*The pointer returned is the property of the associated **Socket** (p. 2452) and should not be deleted by the caller.*

When the returned InputStream is performing a blocking operation and the underlying connection is closed or otherwise broker the read calls will normally throw an exception to indicate the failure.

*Closing the InputStream will also close the underlying **Socket** (p. 2452).*

Returns

The InputStream for this socket.

Exceptions

IOException	if an error occurs during creation of the InputStream, also if the Socket (p. 2452) is not connected or the input has been shutdown previously.
-------------	--

- virtual **decaf::io::OutputStream * getOutputStream ()**

Gets the OutputStream for this socket if it is connected.

*The pointer returned is the property of the **Socket** (p. 2452) instance and should not be deleted by the caller.*

*Closing the returned **Socket** (p. 2452) will also close the underlying **Socket** (p. 2452).*

Returns

the OutputStream for this socket.

Exceptions

IOException	if an error occurs during the creation of this OutputStream, or if the Socket (p. 2452) is closed or the output has been shutdown previously.
-------------	--

- virtual void **shutdownInput ()**

Shuts down the InputStream for this socket essentially marking it as EOF.

The stream returns EOF for any calls to read after this method has been called.

Exceptions

IOException	if an I/O error occurs while performing this operation.
-------------	---

- virtual void **shutdownOutput ()**

Shuts down the OutputStream for this socket, any data already written to the socket will be sent, any further calls to `OutputStream::write` will throw an IOException.

Exceptions

IOException	if an I/O error occurs while performing this operation.
-------------	---

- virtual void **setOOBInline** (bool value)

Sets the value of the OOBINLINE for this socket, by default this option is disabled.

*If enabled the urgent data is read inline on the **Socket** (p. 2452)'s InputStream, no notification is give.*

Returns

true if OOBINLINE is enabled, false otherwise.

6.384 decaf::internal::net::ssl::openssl::OpenSSLSocket Class Reference 2023

Exceptions

SocketException (p. 2471)	<i>if an error is encountered while performing this operation.</i>
-------------------------------------	--

- virtual void **sendUrgentData** (int data)

*Sends on byte of urgent data to the **Socket** (p. 2452).*

Parameters

data	<i>The value to write as urgent data, only the lower eight bits are sent.</i>
------	---

Exceptions

IOException	<i>if an I/O error occurs while performing this operation.</i>
-------------	--

- virtual std::vector< std::string > **getSupportedCipherSuites** () const

*Gets a vector containing the names of all the cipher suites that are supported by this **SSLSocket** (p. 2513).*

*Normally not all of these cipher suites will be enabled on the **Socket** (p. 2452).*

Returns

a vector containing the names of all the supported cipher suites.

- virtual std::vector< std::string > **getSupportedProtocols** () const

*Gets a vector containing the names of all the protocols that could be enabled for this **SSLSocket** (p. 2513) instance.*

Returns

a vector containing the names of all the supported protocols.

- virtual std::vector< std::string > **getEnabledCipherSuites** () const

*Returns a vector containing the names of all the currently enabled Cipher Suites for this **SSL Socket** (p. 2452).*

Returns

vector of the names of all enabled Cipher Suites.

- virtual void **setEnabledCipherSuites** (const std::vector< std::string > &suites)

*Sets the Cipher Suites that are to be enabled on the **SSL Socket** (p. 2452) connection. Each of the named Cipher Suites must appear in the list of supported cipher suites for this connection or an exception will be thrown.*

Parameters

suites	<i>An Vector of names for all the Cipher Suites that are to be enabled.</i>
--------	---

Exceptions

IllegalArgument-Exception	<i>if the vector is empty or one of the names is invalid.</i>
---------------------------	---

- virtual std::vector< std::string > **getEnabledProtocols** () const

*Returns a vector containing the names of all the currently enabled Protocols for this **SSL Socket** (p. 2452).*

Returns

vector of the names of all enabled Protocols.

- virtual void **setEnabledProtocols** (const std::vector< std::string > &protocols)

*Sets the Protocols that are to be enabled on the **SSL Socket** (p. 2452) connection. Each of the named Protocols must appear in the list of supported protocols suites for this connection or an exception will be thrown.*

Parameters

protocols	<i>An Vector of names for all the Protocols that are to be enabled.</i>
-----------	---

Exceptions

IllegalArgument-Exception	<i>if the vector is empty or one of the names is invalid.</i>
---------------------------	---

- virtual void **startHandshake** ()

Initiates a handshake for this SSL Connection, this can be necessary for several reasons such as using new encryption keys, or starting a new session.

When called for the first time after the socket connects this method blocks until the handshake is completed. The provider is not required to support multiple handshakes and can throw an IOException to indicate an error.

Exceptions

IOException	<i>if an I/O error occurs while performing the Handshake</i>
-------------	--

- virtual void **setUseClientMode** (bool value)

Determines the mode that the socket uses when a handshake is initiated, client or server.

*This method must be called prior to any handshake attempts on this **Socket** (p. 2452), once a handshake has been initiated this socket remains the set mode; client or server, for the life of this object.*

Parameters

value	<i>The mode setting, true for client or false for server.</i>
-------	---

Exceptions

IllegalArgument-Exception	<i>if the handshake process has begun and mode is locked.</i>
---------------------------	---

- virtual bool **getUseClientMode** () const

*Gets whether this **Socket** (p. 2452) is in Client or Server mode, true indicates that the mode is set to Client.*

Returns

*true if the **Socket** (p. 2452) is in Client mode, false otherwise.*

- virtual void **setNeedClientAuth** (bool value)

*Sets the **Socket** (p. 2452) to require that a client authenticate itself by sending a valid Certificate that is trusted by this Server mode socket.*

This option only applies to sockets in the Server mode.

If the option is enabled and the client does not provide a certificate then the handshake is considered failed and the connection is refused. Calling this method resets any previous value for this option as well as clears any value set in the setWantClientAuth method.

Parameters

value	<i>The value indicating if a client is required to authenticate itself or not.</i>
-------	--

- virtual bool **getNeedClientAuth** () const

Returns if this socket is configured to require client authentication, true means that it has and that clients that failed to authenticate will be rejected.

This option is only useful when the socket is operating in server mode.

Returns

true if client authentication is required.

- virtual void **setWantClientAuth** (bool value)

*Sets the **Socket** (p. 2452) to request that a client authenticate itself by sending a valid Certificate that is trusted by this Server mode socket.*

This option only applies to sockets in the Server mode.

If the option is enabled an the client does not provide a certificate then the handshake is considered to have succeeded, if it does send a certificate and that certificate is invalid the the handshake will fail. Calling this method resets any previous value for this option as well as clears any value set in the setNeedClientAuth method.

Parameters

value	The value indicating if a client is requested to authenticate itself or not.
-------	--

- virtual bool **getWantClientAuth** () const

Returns if this socket is configured to request client authentication, true means that is has and that clients that failed to authenticate will be rejected but that cleints that do not send a certificate are not considered to have failed authentication.

This option is only useful when the socket is operating in server mode.

Returns

true if client authentication is required.

- int **read** (unsigned char *buffer, int size, int offset, int length)

Reads the requested data from the Socket and write it into the passed in buffer.

- void **write** (const unsigned char *buffer, int size, int offset, int length)

Writes the specified data in the passed in buffer to the Socket.

- int **available** ()

Gets the number of bytes in the Socket buffer that can be read without blocking.

6.384.1 Detailed Description

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

Since

1.0

6.384.2 Constructor & Destructor Documentation

6.384.2.1 decaf::internal::net::ssl::openssl::OpenSSLSocket::OpenSSLSocket (
 OpenSSLParameters * parameters)

6.384.2.2 decaf::internal::net::ssl::openssl::OpenSSLSocket::OpenSSLSocket (
 OpenSSLParameters * parameters, const decaf::net::InetAddress *
 address, int port)

6.384.2.3 `decaf::internal::net::ssl::openssl::OpenSSLSocket::OpenSSLSocket (OpenSSLParameters * parameters, const decaf::net::InetAddress * address, int port, const decaf::net::InetAddress * localAddress, int localPort)`

6.384.2.4 `decaf::internal::net::ssl::openssl::OpenSSLSocket::OpenSSLSocket (OpenSSLParameters * parameters, const std::string & host, int port)`

6.384.2.5 `decaf::internal::net::ssl::openssl::OpenSSLSocket::OpenSSLSocket (OpenSSLParameters * parameters, const std::string & host, int port, const decaf::net::InetAddress * localAddress, int localPort)`

6.384.2.6 `virtual decaf::internal::net::ssl::openssl::OpenSSLSocket::~~OpenSSLSocket () [virtual]`

6.384.3 Member Function Documentation

6.384.3.1 `int decaf::internal::net::ssl::openssl::OpenSSLSocket::available ()`

Gets the number of bytes in the Socket buffer that can be read without blocking.

Returns

the number of bytes that can be read from the Socket without blocking.

Exceptions

<i>IOException</i>	if an I/O error occurs while performing this operation.
--------------------	---

6.384.3.2 `virtual void decaf::internal::net::ssl::openssl::OpenSSLSocket::close () [virtual]`

Closes the **Socket** (p. 2452).

Once closed a **Socket** (p. 2452) cannot be connected or otherwise operated upon, a new **Socket** (p. 2452) instance must be created.

Exceptions

<i>IOException</i>	if an I/O error occurs while closing the Socket (p. 2452).
--------------------	---

Reimplemented from **decaf::net::Socket** (p. 2458).

6.384.3.3 `virtual void decaf::internal::net::ssl::openssl::OpenSSLSocket::connect (const std::string & host, int port, int timeout) [virtual]`

Connects to the specified destination, with a specified timeout value.

If a connection to the remote host is not established within the specified timeout inter-

6.384 decaf::internal::net::ssl::openssl::OpenSSLSocket Class Reference 2027

val than an **SocketTimeoutException** (p. 2493) is thrown. A timeout value of zero is treated as an infinite timeout.

Parameters

<i>host</i>	The host name or IP address of the remote host to connect to.
<i>port</i>	The port on the remote host to connect to.
<i>timeout</i>	The number of Milliseconds to wait before treating the connection as failed.

Exceptions

<i>IOException</i>	Thrown if a failure occurred in the connect.
SocketTimeoutException (p. 2493)	if the timeout for connection is exceeded.
<i>IllegalArgument-Exception</i>	if the timeout value is negative or the endpoint is invalid.

Reimplemented from **decaf::net::Socket** (p. 2459).

6.384.3.4 virtual std::vector<std::string> decaf::internal::net::ssl::openssl::OpenSSLSocket::getEnabledCipherSuites () const
[virtual]

Returns a vector containing the names of all the currently enabled Cipher Suites for this SSL **Socket** (p. 2452).

Returns

vector of the names of all enabled Cipher Suites.

Implements **decaf::net::ssl::SSLSocket** (p. 2517).

6.384.3.5 virtual std::vector<std::string> decaf::internal::net::ssl::openssl::OpenSSLSocket::getEnabledProtocols () const
[virtual]

Returns a vector containing the names of all the currently enabled Protocols for this SSL **Socket** (p. 2452).

Returns

vector of the names of all enabled Protocols.

Implements **decaf::net::ssl::SSLSocket** (p. 2517).

6.384.3.6 `virtual decaf::io::InputStream* decaf::internal::net-
::ssl::openssl::OpenSSLSocket::getInputStream ()
[virtual]`

Gets the InputStream for this socket if its connected.

The pointer returned is the property of the associated **Socket** (p. 2452) and should not be deleted by the caller.

When the returned InputStream is performing a blocking operation and the underlying connection is closed or otherwise broker the read calls will normally throw an exception to indicate the failure.

Closing the InputStream will also close the underlying **Socket** (p. 2452).

Returns

The InputStream for this socket.

Exceptions

<i>IOException</i>	if an error occurs during creation of the InputStream, also if the - Socket (p. 2452) is not connected or the input has been shutdown previously.
--------------------	--

Reimplemented from **decaf::net::Socket** (p. 2459).

6.384.3.7 `virtual bool decaf::internal::net::ssl::openssl::OpenSSLSocket::getNeed-
ClientAuth () const [virtual]`

Returns if this socket is configured to require client authentication, true means that is has and that clients that failed to authenticate will be rejected.

This option is only useful when the socket is operating in server mode.

Returns

true if client authentication is required.

Implements **decaf::net::ssl::SSLSocket** (p. 2517).

6.384.3.8 `virtual decaf::io::OutputStream* decaf::internal::net-
::ssl::openssl::OpenSSLSocket::getOutputStream ()
[virtual]`

Gets the OutputStream for this socket if it is connected.

The pointer returned is the property of the **Socket** (p. 2452) instance and should not be deleted by the caller.

Closing the returned **Socket** (p. 2452) will also close the underlying **Socket** (p. 2452).

6.384 decaf::internal::net::ssl::openssl::OpenSSLSocket Class Reference 2029

Returns

the OutputStream for this socket.

Exceptions

<i>IOException</i>	if an error occurs during the creation of this OutputStream, or if the Socket (p. 2452) is closed or the output has been shutdown previously.
--------------------	--

Reimplemented from **decaf::net::Socket** (p. 2461).

6.384.3.9 `virtual std::vector<std::string> decaf::internal::net::ssl::openssl::OpenSSLSocket::getSupportedCipherSuites () const`
[virtual]

Gets a vector containing the names of all the cipher suites that are supported by this **SSLSocket** (p. 2513).

Normally not all of these cipher suites will be enabled on the **Socket** (p. 2452).

Returns

a vector containing the names of all the supported cipher suites.

Implements **decaf::net::ssl::SSLSocket** (p. 2518).

6.384.3.10 `virtual std::vector<std::string> decaf::internal::net::ssl::openssl::OpenSSLSocket::getSupportedProtocols () const`
[virtual]

Gets a vector containing the names of all the protocols that could be enabled for this **SSLSocket** (p. 2513) instance.

Returns

a vector containing the names of all the supported protocols.

Implements **decaf::net::ssl::SSLSocket** (p. 2518).

6.384.3.11 `virtual bool decaf::internal::net::ssl::openssl::OpenSSLSocket::getUseClientMode () const` [virtual]

Gets whether this **Socket** (p. 2452) is in Client or Server mode, true indicates that the mode is set to Client.

Returns

true if the **Socket** (p. 2452) is in Client mode, false otherwise.

Implements **decaf::net::ssl::SSLSocket** (p. 2518).

6.384.3.12 virtual bool **decaf::internal::net::ssl::openssl::OpenSSLSocket::get-WantClientAuth** () const [virtual]

Returns if this socket is configured to request client authentication, true means that it has and that clients that failed to authenticate will be rejected but that clients that do not send a certificate are not considered to have failed authentication.

This option is only useful when the socket is operating in server mode.

Returns

true if client authentication is required.

Implements **decaf::net::ssl::SSLSocket** (p. 2519).

6.384.3.13 int **decaf::internal::net::ssl::openssl::OpenSSLSocket::read** (unsigned char * *buffer*, int *size*, int *offset*, int *length*)

Reads the requested data from the Socket and write it into the passed in buffer.

Parameters

<i>buffer</i>	The buffer to read into
<i>size</i>	The size of the specified buffer
<i>offset</i>	The offset into the buffer where reading should start filling.
<i>length</i>	The number of bytes past offset to fill with data.

Returns

the actual number of bytes read or -1 if at EOF.

Exceptions

<i>IOException</i>	if an I/O error occurs during the read.
<i>NullPointerException</i>	if buffer is Null.
<i>IndexOutOfBoundsException</i>	if offset + length is greater than buffer size.

6.384.3.14 virtual void **decaf::internal::net::ssl::openssl::OpenSSLSocket::send-UrgentData** (int *data*) [virtual]

Sends on byte of urgent data to the **Socket** (p. 2452).

Parameters

<i>data</i>	The value to write as urgent data, only the lower eight bits are sent.
-------------	--

6.384 decaf::internal::net::ssl::openssl::OpenSSLSocket Class Reference 2031

Exceptions

<i>IOException</i>	if an I/O error occurs while performing this operation.
--------------------	---

Reimplemented from **decaf::net::Socket** (p. 2465).

6.384.3.15 virtual void **decaf::internal::net::ssl::openssl::OpenSSLSocket::setEnabledCipherSuites** (const std::vector< std::string > & *suites*)
[virtual]

Sets the Cipher Suites that are to be enabled on the SSL **Socket** (p. 2452) connection.

Each of the named Cipher Suites must appear in the list of supported cipher suites for this connection or an exception will be thrown.

Parameters

<i>suites</i>	An Vector of names for all the Cipher Suites that are to be enabled.
---------------	--

Exceptions

<i>IllegalArgument-Exception</i>	if the vector is empty or one of the names is invalid.
----------------------------------	--

Implements **decaf::net::ssl::SSLSocket** (p. 2519).

6.384.3.16 virtual void **decaf::internal::net::ssl::openssl::OpenSSLSocket::setEnabledProtocols** (const std::vector< std::string > & *protocols*)
[virtual]

Sets the Protocols that are to be enabled on the SSL **Socket** (p. 2452) connection.

Each of the named Protocols must appear in the list of supported protocols suites for this connection or an exception will be thrown.

Parameters

<i>protocols</i>	An Vector of names for all the Protocols that are to be enabled.
------------------	--

Exceptions

<i>IllegalArgument-Exception</i>	if the vector is empty or one of the names is invalid.
----------------------------------	--

Implements **decaf::net::ssl::SSLSocket** (p. 2519).

6.384.3.17 virtual void **decaf::internal::net::ssl::openssl::-**
OpenSSLSocket::setNeedClientAuth (bool *value*)
 [virtual]

Sets the **Socket** (p. 2452) to require that a client authenticate itself by sending a valid Certificate that is trusted by this Server mode socket.

This option only applies to sockets in the Server mode.

If the option is enabled an the client does not provide a certificate then the handshake is considered failed and the connection is refused. Calling this method resets any previous value for this option as well as clears any value set in the setWantClientAuth method.

Parameters

<i>value</i>	The value indicating if a client is required to authenticate itself or not.
--------------	---

Implements **decaf::net::ssl::SSLSocket** (p. 2520).

6.384.3.18 virtual void **decaf::internal::net::ssl::openssl::OpenSSLSocket::setOOB-**
Inline (bool *value*) [virtual]

Sets the value of the OOBINLINE for this socket, by default this option is disabled.

If enabled the urgent data is read inline on the **Socket** (p. 2452)'s InputStream, no notification is give.

Returns

true if OOBINLINE is enabled, false otherwise.

Exceptions

SocketException (p. 2471)	if an error is encountered while performing this operation.
-------------------------------------	---

Reimplemented from **decaf::net::Socket** (p. 2465).

6.384.3.19 virtual void **decaf::internal::net::ssl::openssl::-**
OpenSSLSocket::setUseClientMode (bool *value*)
 [virtual]

Determines the mode that the socket uses when a handshake is initiated, client or server.

This method must be called prior to any handshake attempts on this **Socket** (p. 2452), once a handshake has be initiated this socket remains the the set mode; client or server, for the life of this object.

6.384 decaf::internal::net::ssl::openssl::OpenSSLSocket Class Reference 2033

Parameters

<i>value</i>	The mode setting, true for client or false for server.
--------------	--

Exceptions

<i>IllegalArgument-Exception</i>	if the handshake process has begun and mode is locked.
----------------------------------	--

Implements **decaf::net::ssl::SSLSocket** (p. 2520).

6.384.3.20 virtual void decaf::internal::net::ssl::openssl::OpenSSLSocket::setWantClientAuth (bool *value*)
[virtual]

Sets the **Socket** (p. 2452) to request that a client authenticate itself by sending a valid Certificate that is trusted by this Server mode socket.

This option only applies to sockets in the Server mode.

If the option is enabled and the client does not provide a certificate then the handshake is considered to have succeeded, if it does send a certificate and that certificate is invalid the the handshake will fail. Calling this method resets any previous value for this option as well as clears any value set in the setNeedClientAuth method.

Parameters

<i>value</i>	The value indicating if a client is requested to authenticate itself or not.
--------------	--

Implements **decaf::net::ssl::SSLSocket** (p. 2521).

6.384.3.21 virtual void decaf::internal::net::ssl::openssl::OpenSSLSocket::shutdownInput () [virtual]

Shuts down the InputStream for this socket essentially marking it as EOF.

The stream returns EOF for any calls to read after this method has been called.

Exceptions

<i>IOException</i>	if an I/O error occurs while performing this operation.
--------------------	---

Reimplemented from **decaf::net::Socket** (p. 2469).

6.384.3.22 virtual void decaf::internal::net::ssl::openssl::OpenSSLSocket::shutdownOutput () [virtual]

Shuts down the OutputStream for this socket, any data already written to the socket will be sent, any further calls to OutputStream::write will throw an IOException.

Exceptions

<i>IOException</i>	if an I/O error occurs while performing this operation.
--------------------	---

Reimplemented from **decaf::net::Socket** (p. 2469).

6.384.3.23 virtual void decaf::internal::net::ssl::openssl::OpenSSLSocket::start-Handshake () [virtual]

Initiates a handshake for this SSL Connection, this can be necessary for several reasons such as using new encryption keys, or starting a new session.

When called for the first time after the socket connects this method blocks until the handshake is completed. The provider is not required to support multiple handshakes and can throw an *IOException* to indicate an error.

Exceptions

<i>IOException</i>	if an I/O error occurs while performing the Handshake
--------------------	---

Implements **decaf::net::ssl::SSLSocket** (p. 2521).

6.384.3.24 void decaf::internal::net::ssl::openssl::OpenSSLSocket::write (const unsigned char * buffer, int size, int offset, int length)

Writes the specified data in the passed in buffer to the Socket.

Parameters

<i>buffer</i>	The buffer to write to the socket.
<i>size</i>	The size of the specified buffer.
<i>offset</i>	The offset into the buffer where the data to write starts at.
<i>length</i>	The number of bytes past offset to write.

Exceptions

<i>IOException</i>	if an I/O error occurs during the write.
<i>NullPointerException</i>	if buffer is Null.
<i>IndexOutOfBoundsException</i>	if offset + length is greater than buffer size.

The documentation for this class was generated from the following file:

- src/main/decaf/internal/net/ssl/openssl/**OpenSSLSocket.h**

6.385 decaf::internal::net::ssl::openssl::OpenSSLSocketException Class Reference

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

```
#include <src/main/decaf/internal/net/ssl/openssl/OpenSSL-  
LSocketException.h>
```

Inheritance diagram for decaf::internal::net::ssl::openssl::OpenSSLSocketException:

Public Member Functions

- **OpenSSLSocketException** () throw ()
*Creates a new **OpenSSLSocketException** (p. 2029) with default values.*
- **OpenSSLSocketException** (const **Exception** &ex) throw ()
Conversion Constructor from some other Exception.
- **OpenSSLSocketException** (const **OpenSSLSocketException** &ex) throw ()
Copy Constructor.
- **OpenSSLSocketException** (const char *file, const int lineNumber, const std::exception *cause, const char *msg,...) throw ()
*Create a new **OpenSSLSocketException** (p. 2029) and initializes the file name and line number where this message occurred.*
- **OpenSSLSocketException** (const std::exception *cause) throw ()
*Creates a new **OpenSSLSocketException** (p. 2029) with the passed exception set as the cause of this exception.*
- **OpenSSLSocketException** (const char *file, const int lineNumber, const char *msg,...) throw ()
*Create a new **OpenSSLSocketException** (p. 2029) and initializes the file name and line number where this message occurred.*
- **OpenSSLSocketException** (const char *file, const int lineNumber) throw ()
*Create a new **OpenSSLSocketException** (p. 2029) and initializes the file name and line number where this message occurred.*
- virtual **OpenSSLSocketException** * clone () const
Clones this exception.
- virtual ~**OpenSSLSocketException** () throw ()

Protected Member Functions

- std::string **getErrorString** () const
Gets and formats an error message string from the OpenSSL error stack.

6.385.1 Detailed Description

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

Since

1.0

6.385.2 Constructor & Destructor Documentation

6.385.2.1 `decaf::internal::net::ssl::openssl::OpenSSLSocketException::OpenSSLSocketException () throw ()`

Creates an new **OpenSSLSocketException** (p. 2029) with default values.

6.385.2.2 `decaf::internal::net::ssl::openssl::OpenSSLSocketException::OpenSSLSocketException (const Exception & ex) throw ()`

Conversion Constructor from some other Exception.

Parameters

<code>ex</code>	An Exception object that should become this type of Exception.
-----------------	--

6.385.2.3 `decaf::internal::net::ssl::openssl::OpenSSLSocketException::OpenSSLSocketException (const OpenSSLSocketException & ex) throw ()`

Copy Constructor.

Parameters

<code>ex</code>	The OpenSSLSocketException (p. 2029) whose values should be copied to this instance.
-----------------	---

6.385.2.4 `decaf::internal::net::ssl::openssl::OpenSSLSocketException::OpenSSLSocketException (const char * file, const int lineNumber, const std::exception * cause, const char * msg, ...) throw ()`

Create a new **OpenSSLSocketException** (p. 2029) and initializes the file name and line number where this message occurred.

Sets the message to report, using an optional list of arguments to parse into the message.

Parameters

<i>file</i>	The file name where exception occurs
<i>lineNumber</i>	The line number where the exception occurred.
<i>cause</i>	The exception that was the cause for this one to be thrown (can be null).
<i>msg</i>	The error message to report.
...	The list of primitives that are formatted into the message.

6.385.2.5 decaf::internal::net::ssl::openssl::OpenSSLSocketException-
::OpenSSLSocketException (const std::exception * *cause*) throw
()

Creates a new **OpenSSLSocketException** (p. 2029) with the passed exception set as the cause of this exception.

Parameters

<i>cause</i>	Pointer to the exception that caused this one to be thrown, the object is cloned caller retains ownership.
--------------	--

6.385.2.6 decaf::internal::net::ssl::openssl::OpenSSLSocketException::OpenSS-
LSocketException (const char * *file*, const int *lineNumber*, const char * *msg*, ...
) throw ()

Create a new **OpenSSLSocketException** (p. 2029) and initializes the file name and line number where this message occurred.

Sets the message to report, using an optional list of arguments to parse into the message.

Parameters

<i>file</i>	The file name where exception occurs.
<i>lineNumber</i>	The line number where the exception occurred.
<i>msg</i>	The error message to report.
...	The list of primitives that are formatted into the message

6.385.2.7 decaf::internal::net::ssl::openssl::OpenSSLSocketException::Open-
SSLSocketException (const char * *file*, const int *lineNumber*) throw
()

Create a new **OpenSSLSocketException** (p. 2029) and initializes the file name and line number where this message occurred.

Sets the message to report by getting the complete set of error messages from the OpenSSL error stack and concatenating them into one string.

Parameters

<i>file</i>	The file name where exception occurs.
<i>lineNumber</i>	The line number where the exception occurred.

6.385.2.8 **virtual decaf::internal::net::ssl::openssl::OpenSSLSocketException::~~OpenSSLSocketException () throw ()**
[virtual]

6.385.3 Member Function Documentation

6.385.3.1 **virtual OpenSSLSocketException* decaf::internal::net::ssl::openssl::OpenSSLSocketException::clone () const** [inline, virtual]

Clones this exception.

This is useful for cases where you need to preserve the type of the original exception as well as the message. All subclasses should override this method.

Returns

a new Exception instance that is a copy of this Exception object.

Reimplemented from **decaf::net::SocketException** (p. 2473).

6.385.3.2 **std::string decaf::internal::net::ssl::openssl::OpenSSLSocketException::getErrorString () const**
[protected]

Gets and formats an error message string from the OpenSSL error stack.

Returns

a string containing the complete OpenSSL error string.

The documentation for this class was generated from the following file:

- src/main/decaf/internal/net/ssl/openssl/**OpenSSLSocketException.h**

6.386 decaf::internal::net::ssl::openssl::OpenSSLSocketFactory Class Reference

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

```
#include <src/main/decaf/internal/net/ssl/openssl/OpenSSLSocketFactory.h>
```

Inheritance diagram for decaf::internal::net::ssl::openssl::OpenSSLSocketFactory:

Public Member Functions

- **OpenSSLSocketFactory** (**OpenSSLContextSpi** *parent)
- virtual **~OpenSSLSocketFactory** ()
- virtual **decaf::net::Socket** * **createSocket** ()

*Creates an unconnected **Socket** (p. 2452) object.*

Returns

*a new **Socket** (p. 2452) object, caller must free this object when done.*

Exceptions

IOException	<i>if the Socket (p. 2452) cannot be created.</i>
-------------	--

- virtual **decaf::net::Socket** * **createSocket** (const **decaf::net::InetAddress** *host, int port)

*Creates a new **Socket** (p. 2452) object and connects it to the specified remote host and port using the configuration of this **SocketFactory** (p. 2473).*

Parameters

host	<i>The host to connect the socket to.</i>
port	<i>The port on the remote host to connect to.</i>

Returns

*a new **Socket** (p. 2452) object, caller must free this object when done.*

Exceptions

IOException	<i>if an I/O error occurs while creating the Socket (p. 2452) object.</i>
UnknownHostException (p. 2816)	<i>if the host name is not known.</i>

- virtual **decaf::net::Socket** * **createSocket** (const **decaf::net::InetAddress** *host, int port, const **decaf::net::InetAddress** *ifAddress, int localPort)

*Creates a new **Socket** (p. 2452) object and connects it to the specified remote host and port using the configuration of this **SocketFactory** (p. 2473).*

*The **Socket** (p. 2452) will be bound to the specified local address and port.*

Parameters

host	<i>The host to connect the socket to.</i>
port	<i>The port on the remote host to connect to.</i>
ifAddress	<i>The address on the local machine to bind the Socket (p. 2452) to.</i>
localPort	<i>The local port to bind the Socket (p. 2452) to.</i>

Returns

*a new **Socket** (p. 2452) object, caller must free this object when done.*

Exceptions

IOException	if an I/O error occurs while creating the Socket (p. 2452) object.
UnknownHostException (p. 2816)	if the host name is not known.

- virtual **decaf::net::Socket** * **createSocket** (const std::string &hostname, int port)

Creates a new **Socket** (p. 2452) object and connects it to the specified remote host and port using the configuration of this **SocketFactory** (p. 2473).

Parameters

host	The host name or IP address to connect the socket to.
port	The port on the remote host to connect to.

Returns

a new **Socket** (p. 2452) object, caller must free this object when done.

Exceptions

IOException	if an I/O error occurs while creating the Socket (p. 2452) object.
UnknownHostException (p. 2816)	if the host name is not known.

- virtual **decaf::net::Socket** * **createSocket** (const std::string &name, int port, const **decaf::net::InetAddress** *ifAddress, int localPort)

Creates a new **Socket** (p. 2452) object and connects it to the specified remote host and port using the configuration of this **SocketFactory** (p. 2473).

Parameters

host	The host name or IP address to connect the socket to.
port	The port on the remote host to connect to.
ifAddress	The address on the local machine to bind the Socket (p. 2452) to.
localPort	The local port to bind the Socket (p. 2452) to.

Returns

a new **Socket** (p. 2452) object, caller must free this object when done.

Exceptions

IOException	if an I/O error occurs while creating the Socket (p. 2452) object.
UnknownHostException (p. 2816)	if the host name is not known.

- virtual std::vector< std::string > **getDefaultCipherSuites** ()

Returns the list of cipher suites which are enabled by default.

Unless a different list is enabled, handshaking on an SSL connection will use one of these cipher suites. The minimum quality of service for these defaults requires confidentiality protection and server authentication (that is, no anonymous cipher suites).

Returns

an STL vector containing the list of cipher suites enabled by default.

See also

getSupportedCipherSuites() (p. 2524)

- virtual std::vector< std::string > **getSupportedCipherSuites** ()

Returns the names of the cipher suites which could be enabled for use on an SSL connection.

Normally, only a subset of these will actually be enabled by default, since this list may include cipher suites which do not meet quality of service requirements for those defaults. Such cipher suites are useful in specialized applications.

Returns

an STL vector containing the list of supported cipher suites.

See also

getDefaultCipherSuites() (p. 2524)

- virtual **decaf::net::Socket** * **createSocket** (**decaf::net::Socket** *socket, std::string host, int port, bool autoClose)

Returns a socket layered over an existing socket connected to the named host, at the given port.

This constructor can be used when tunneling SSL through a proxy or when negotiating the use of SSL over an existing socket. The host and port refer to the logical peer destination. This socket is configured using the socket options established for this factory.

Parameters

socket	The existing socket to layer over.
host	The server host the original Socket (p. 2452) is connected to.
port	The server port the original Socket (p. 2452) is connected to.
autoClose	Should the layered over Socket (p. 2452) be closed when the topmost socket is closed.

Returns

a new **Socket** (p. 2452) instance that wraps the given **Socket** (p. 2452).

Exceptions

IOException	if an I/O exception occurs while performing this operation.
UnknownHostException (p. 2816)	if the host is unknown.

6.386.1 Detailed Description

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

Since

1.0

6.386.2 Constructor & Destructor Documentation

6.386.2.1 `decaf::internal::net::ssl::openssl::OpenSSLSocketFactory::OpenSSLSocketFactory (OpenSSLContextSpi * parent)`

6.386.2.2 `virtual decaf::internal::net::ssl::openssl::OpenSSLSocketFactory::~~OpenSSLSocketFactory ()`
[virtual]

6.386.3 Member Function Documentation

6.386.3.1 `virtual decaf::net::Socket* decaf::internal::net::ssl::openssl::OpenSSLSocketFactory::createSocket ()`
[virtual]

Creates an unconnected **Socket** (p. 2452) object.

Returns

a new **Socket** (p. 2452) object, caller must free this object when done.

Exceptions

<i>IOException</i>	if the Socket (p. 2452) cannot be created.
--------------------	---

Reimplemented from **decaf::net::SocketFactory** (p. 2475).

6.386.3.2 `virtual decaf::net::Socket* decaf::internal::net::ssl::openssl::OpenSSLSocketFactory::createSocket (const decaf::net::InetAddress * host, int port)` [virtual]

Creates a new **Socket** (p. 2452) object and connects it to the specified remote host and port using the configuration of this **SocketFactory** (p. 2473).

Parameters

<i>host</i>	The host to connect the socket to.
<i>port</i>	The port on the remote host to connect to.

Returns

a new **Socket** (p. 2452) object, caller must free this object when done.

Exceptions

<i>IOException</i>	if an I/O error occurs while creating the Socket (p. 2452) object.
UnknownHostException (p. 2816)	if the host name is not known.

Implements **decaf::net::SocketFactory** (p. 2475).

6.386.3.3 virtual **decaf::net::Socket*** **decaf::internal::net::ssl::openssl::OpenSSLSocketFactory::createSocket** (const **decaf::net::InetAddress** * *host*, int *port*, const **decaf::net::InetAddress** * *ifAddress*, int *localPort*) [virtual]

Creates a new **Socket** (p. 2452) object and connects it to the specified remote host and port using the configuration of this **SocketFactory** (p. 2473).

The **Socket** (p. 2452) will be bound to the specified local address and port.

Parameters

<i>host</i>	The host to connect the socket to.
<i>port</i>	The port on the remote host to connect to.
<i>ifAddress</i>	The address on the local machine to bind the Socket (p. 2452) to.
<i>localPort</i>	The local port to bind the Socket (p. 2452) to.

Returns

a new **Socket** (p. 2452) object, caller must free this object when done.

Exceptions

<i>IOException</i>	if an I/O error occurs while creating the Socket (p. 2452) object.
UnknownHostException (p. 2816)	if the host name is not known.

Implements **decaf::net::SocketFactory** (p. 2476).

6.386.3.4 virtual **decaf::net::Socket*** **decaf::internal::net::ssl::openssl::OpenSSLSocketFactory::createSocket** (const std::string & *hostname*, int *port*) [virtual]

Creates a new **Socket** (p. 2452) object and connects it to the specified remote host and port using the configuration of this **SocketFactory** (p. 2473).

Parameters

<i>host</i>	The host name or IP address to connect the socket to.
<i>port</i>	The port on the remote host to connect to.

Returns

a new **Socket** (p. 2452) object, caller must free this object when done.

Exceptions

<i>IOException</i>	if an I/O error occurs while creating the Socket (p. 2452) object.
UnknownHostException (p. 2816)	if the host name is not known.

Implements **decaf::net::SocketFactory** (p. 2476).

6.386.3.5 virtual **decaf::net::Socket*** **decaf::internal::net::ssl::openssl::OpenSSL-LSocketFactory::createSocket** (const std::string & *name*, int *port*, const **decaf::net::InetAddress** * *ifAddress*, int *localPort*) [virtual]

Creates a new **Socket** (p. 2452) object and connects it to the specified remote host and port using the configuration of this **SocketFactory** (p. 2473).

Parameters

<i>host</i>	The host name or IP address to connect the socket to.
<i>port</i>	The port on the remote host to connect to.
<i>ifAddress</i>	The address on the local machine to bind the Socket (p. 2452) to.
<i>localPort</i>	The local port to bind the Socket (p. 2452) to.

Returns

a new **Socket** (p. 2452) object, caller must free this object when done.

Exceptions

<i>IOException</i>	if an I/O error occurs while creating the Socket (p. 2452) object.
UnknownHostException (p. 2816)	if the host name is not known.

Implements **decaf::net::SocketFactory** (p. 2477).

6.386.3.6 virtual **decaf::net::Socket*** **decaf::internal::net::ssl::openssl::OpenSSL-SocketFactory::createSocket** (**decaf::net::Socket** * *socket*, std::string *host*, int *port*, bool *autoClose*) [virtual]

Returns a socket layered over an existing socket connected to the named host, at the given port.

This constructor can be used when tunneling SSL through a proxy or when negotiating

the use of SSL over an existing socket. The host and port refer to the logical peer destination. This socket is configured using the socket options established for this factory.

Parameters

<i>socket</i>	The existing socket to layer over.
<i>host</i>	The server host the original Socket (p. 2452) is connected to.
<i>port</i>	The server port the original Socket (p. 2452) is connected to.
<i>autoClose</i>	Should the layered over Socket (p. 2452) be closed when the topmost socket is closed.

Returns

a new **Socket** (p. 2452) instance that wraps the given **Socket** (p. 2452).

Exceptions

<i>IOException</i>	if an I/O exception occurs while performing this operation.
UnknownHostException (p. 2816)	if the host is unknown.

Implements **decaf::net::ssl::SSLSocketFactory** (p. 2523).

6.386.3.7 `virtual std::vector<std::string> decaf::internal::net::ssl::openssl::OpenSSLSocketFactory::getDefaultCipherSuites ()`
[virtual]

Returns the list of cipher suites which are enabled by default.

Unless a different list is enabled, handshaking on an SSL connection will use one of these cipher suites. The minimum quality of service for these defaults requires confidentiality protection and server authentication (that is, no anonymous cipher suites).

Returns

an STL vector containing the list of cipher suites enabled by default.

See also

getSupportedCipherSuites() (p. 2524)

Implements **decaf::net::ssl::SSLSocketFactory** (p. 2524).

6.386.3.8 `virtual std::vector<std::string> decaf::internal::net::ssl::openssl-
::OpenSSLSocketFactory::getSupportedCipherSuites ()
[virtual]`

Returns the names of the cipher suites which could be enabled for use on an SSL connection.

Normally, only a subset of these will actually be enabled by default, since this list may include cipher suites which do not meet quality of service requirements for those defaults. Such cipher suites are useful in specialized applications.

Returns

an STL vector containing the list of supported cipher suites.

See also

getDefaultCipherSuites() (p. 2524)

Implements **decaf::net::ssl::SSLSocketFactory** (p. 2524).

The documentation for this class was generated from the following file:

- `src/main/decaf/internal/net/ssl/openssl/OpenSSLSocketFactory.h`

6.387 decaf::internal::net::ssl::openssl::OpenSSLSocketInputStream Class Reference

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

```
#include <src/main/decaf/internal/net/ssl/openssl/OpenSSLSocketInputStream.h>
```

Inheritance diagram for `decaf::internal::net::ssl::openssl::OpenSSLSocketInputStream`:

Public Member Functions

- **OpenSSLSocketInputStream (OpenSSLSocket *socket)**
- **virtual ~OpenSSLSocketInputStream ()**
- **virtual int available () const**

Indicates the number of bytes available.

The default implementation of this methods returns 0. Classes that override this method may return the total number of bytes that are currently available for reading and others may simply return a value of one indicating that there is some data available. The caller should view the result of this method as an absolute.

The default implementation of this method returns zero.

Returns

the number of bytes available on this input stream.

Exceptions

IOException (p. 1545)	<i>if an I/O error occurs.</i>
------------------------------	--------------------------------

- virtual void **close** ()
Close - does nothing.
- virtual long long **skip** (long long num)
Not supported.

Protected Member Functions

- virtual int **doReadByte** ()
- virtual int **doReadArrayBounded** (unsigned char *buffer, int size, int offset, int length)

6.387.1 Detailed Description

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

Since

1.0

6.387.2 Constructor & Destructor Documentation

6.387.2.1 decaf::internal::net::ssl::openssl::OpenSSLSocketInputStream-
::OpenSSLSocketInputStream (OpenSSLSocket * socket
)

6.387.2.2 virtual decaf::internal::net::ssl::openssl::OpenSSL-
SocketInputStream::~~OpenSSLSocketInputStream ()
[virtual]

6.387.3 Member Function Documentation

6.387.3.1 virtual int decaf::internal::net::ssl::openssl::OpenSSLSocketInput-
Stream::available () const [virtual]

Indicates the number of bytes available.

The default implementation of this methods returns 0. Classes that override this method may return the total number of bytes that are currently available for reading and others may simply return a value of one indicating that there is some data available. The caller should view the result of this method as an absolute.

The default implementation of this method returns zero.

Returns

the number of bytes available on this input stream.

Exceptions

<i>IOException</i> (p. 1545)	if an I/O error occurs.
--	-------------------------

Reimplemented from **decaf::io::InputStream** (p. 1466).

6.387.3.2 virtual void **decaf::internal::net::ssl::openssl::OpenSSLSocketInputStream::close ()** [virtual]

Close - does nothing.

It is the responsibility of the owner of the socket object to close it.

Closes the **InputStream** (p. 1464) freeing any resources that might have been acquired during the lifetime of this stream.

The default implementation of this method does nothing.

Exceptions

<i>IOException</i> (p. 1545)	if an I/O error occurs while closing the InputStream (p. 1464).
--	--

Reimplemented from **decaf::io::InputStream** (p. 1466).

6.387.3.3 virtual int **decaf::internal::net::ssl::openssl::OpenSSLSocketInputStream::doReadArrayBounded (unsigned char * *buffer*, int *size*, int *offset*, int *length*)** [protected, virtual]

Reimplemented from **decaf::io::InputStream** (p. 1467).

6.387.3.4 virtual int **decaf::internal::net::ssl::openssl::OpenSSLSocketInputStream::doReadByte ()** [protected, virtual]

Implements **decaf::io::InputStream** (p. 1467).

6.387.3.5 virtual long long **decaf::internal::net::ssl::openssl::OpenSSLSocketInputStream::skip (long long *num*)** [virtual]

Not supported.

Skips over and discards n bytes of data from this input stream.

6.388 decaf::internal::net::ssl::openssl::OpenSSLSocketOutputStream Class Reference **2049**

The skip method may, for a variety of reasons, end up skipping over some smaller number of bytes, possibly 0. This may result from any of a number of conditions; reaching end of file before n bytes have been skipped is only one possibility. The actual number of bytes skipped is returned.

The skip method of **InputStream** (p. 1464) creates a byte array and then repeatedly reads into it until num bytes have been read or the end of the stream has been reached. Subclasses are encouraged to provide a more efficient implementation of this method.

Parameters

<i>num</i>	The number of bytes to skip.
------------	------------------------------

Returns

total bytes skipped

Exceptions

<i>IOException</i> (p. 1545)	if an I/O error occurs.
<i>Unsupported- OperationException</i>	if the concrete stream class does not support skipping bytes.

Reimplemented from **decaf::io::InputStream** (p. 1472).

The documentation for this class was generated from the following file:

- src/main/decaf/internal/net/ssl/openssl/**OpenSSLSocketInputStream.h**

**6.388 decaf::internal::net::ssl::openssl::OpenSSLSocketOutputStream-
Stream Class Reference**

OutputStream implementation used to write data to an **OpenSSLSocket** (p. 2014) instance.

```
#include <src/main/decaf/internal/net/ssl/openssl/OpenSSL-  
SocketOutputStream.h>
```

Inheritance diagram for decaf::internal::net::ssl::openssl::OpenSSLSocketOutputStream:

Public Member Functions

- **OpenSSLSocketOutputStream** (**OpenSSLSocket** *socket)
- virtual ~**OpenSSLSocketOutputStream** ()

- virtual void **close** ()

*Closes this object and deallocates the appropriate resources.
The object is generally no longer usable after calling close.*

Exceptions

IOException (p. 1545)	if an error occurs while closing.
------------------------------	-----------------------------------

The default implementation of this method does nothing.

Protected Member Functions

- virtual void **doWriteByte** (unsigned char c)
- virtual void **doWriteArrayBounded** (const unsigned char *buffer, int size, int offset, int length)

6.388.1 Detailed Description

OutputStream implementation used to write data to an **OpenSSLSocket** (p. 2014) instance.

Since

1.0

6.388.2 Constructor & Destructor Documentation

6.388.2.1 **decaf::internal::net::ssl::openssl::OpenSSLSocketOutputStream::OpenSSLSocketOutputStream (OpenSSLSocket * socket)**

6.388.2.2 **virtual decaf::internal::net::ssl::openssl::OpenSSLSocketOutputStream::~~OpenSSLSocketOutputStream ()**
[virtual]

6.388.3 Member Function Documentation

6.388.3.1 **virtual void decaf::internal::net::ssl::openssl::OpenSSLSocketOutputStream::close ()** [virtual]

Closes this object and deallocates the appropriate resources.

The object is generally no longer usable after calling close.

Exceptions

IOException (p. 1545)	if an error occurs while closing.
---------------------------------	-----------------------------------

6.389 activemq::wireformat::openwire::OpenWireFormat Class Reference 2051

The default implementation of this method does nothing.

The default implementation of this method does nothing.

Reimplemented from **decaf::io::OutputStream** (p. 2068).

6.388.3.2 virtual void **decaf::internal::net::ssl::openssl::OpenSSLSocketOutputStream::doWriteArrayBounded** (const unsigned char * *buffer*, int *size*, int *offset*, int *length*) [protected, virtual]

Reimplemented from **decaf::io::OutputStream** (p. 2068).

6.388.3.3 virtual void **decaf::internal::net::ssl::openssl::OpenSSLSocketOutputStream::doWriteByte** (unsigned char *c*) [protected, virtual]

Implements **decaf::io::OutputStream** (p. 2069).

The documentation for this class was generated from the following file:

- src/main/decaf/internal/net/ssl/openssl/**OpenSSLSocketOutputStream.h**

6.389 activemq::wireformat::openwire::OpenWireFormat Class - Reference

```
#include <src/main/activemq/wireformat/openwire/OpenWire-  
Format.h>
```

Inheritance diagram for activemq::wireformat::openwire::OpenWireFormat:

Public Member Functions

- **OpenWireFormat** (const **decaf::util::Properties** &properties)
*Constructs a new **OpenWireFormat** (p. 2045) object.*
- virtual **~OpenWireFormat** ()
- virtual bool **hasNegotiator** () const
*Returns true if this **WireFormat** (p. 2884) has a Negotiator that needs to wrap the Transport that uses it.*
Returns
*true if the **WireFormat** (p. 2884) provides a Negotiator.*
- virtual **Pointer** < **transport::Transport** > **createNegotiator** (const **Pointer**< **transport::Transport** > &transport)
If the Transport Provides a Negotiator this method will create and return a new instance of the Negotiator.

Parameters

transport	<i>The Transport to Wrap the Negotiator around.</i>
-----------	---

Returns

*new instance of a **WireFormatNegotiator** (p. 2904) as a **Pointer<Transport>** (p. 2083).*

Exceptions

UnsupportedOperation-Exception	<i>if the WireFormat (p. 2884) doesn't have a Negotiator.</i>
--------------------------------	--

- void **addMarshaller** (**marshal::DataStreamMarshaller** *marshaller)
Allows an external source to add marshalers to this object for types that may be marshaled or unmarshaled.
- virtual void **marshal** (const **Pointer< commands::Command >** &command, const **activemq::transport::Transport** *transport, **decaf::io::DataOutputStream** *out)
Stream based marshaling of a Command, this method blocks until the entire Command has been written out to the output stream.

Parameters

command	<i>The Command to Marshal</i>
transport	<i>The Transport that called this method.</i>
out	<i>The output stream to write the command to.</i>

Exceptions

IOException	<i>if an I/O error occurs.</i>
-------------	--------------------------------

- virtual **Pointer < commands::Command >** **unmarshal** (const **activemq::transport::Transport** *transport, **decaf::io::DataInputStream** *in)
*Stream based unmarshaling, blocks on reads on the input stream until a complete command has been read and unmarshaled into the correct form.
Returns a Pointer to the newly unmarshaled Command.*

Parameters

transport	<i>Pointer to the transport that is making this request.</i>
in	<i>The input stream to read the command from.</i>

Returns

the newly marshaled Command, caller owns the pointer

Exceptions

IOException	<i>if an I/O error occurs.</i>
-------------	--------------------------------

- virtual int **tightMarshalNestedObject1** (**commands::DataStructure** *object, **utils::BooleanStream** *bs)
Utility method for Tight Marshaling the given object to the boolean stream passed.
- void **tightMarshalNestedObject2** (**commands::DataStructure** *o, **decaf::io::DataOutputStream** *ds, **utils::BooleanStream** *bs)
Utility method that will Tight marshal some internally nested object that implements the DataStructure interface.

- **commands::DataStructure * tightUnmarshalNestedObject** (decaf::io::DataInputStream *dis, utils::BooleanStream *bs)
Utility method used to Unmarshal a Nested DataStructure type object from the given DataInputStream.
- **commands::DataStructure * looseUnmarshalNestedObject** (decaf::io::DataInputStream *dis)
Utility method to unmarshal an DataStructure object from an DataInputStream using the Loose Unmarshaling format.
- **void looseMarshalNestedObject** (commands::DataStructure *o, decaf::io::DataOutputStream *dataOut)
Utility method to loosely Marshal an object that is derived from the DataStrucutre interface.
- **void renegotiateWireFormat** (const commands::WireFormatInfo &info)
Called to re-negotiate the settings for the WireFormatInfo, these determine how the client and broker communicate.
- **void setPreferedWireFormatInfo** (const Pointer< commands::WireFormatInfo > &info)
Configures this object using the provided WireformatInfo object.
- **const Pointer < commands::WireFormatInfo > & getPreferedWireFormatInfo** () const
Gets the Preferred WireFormatInfo object that this class holds.
- **bool isStackTraceEnabled** () const
Checks if the stackTraceEnabled flag is on.
- **void setStackTraceEnabled** (bool stackTraceEnabled)
Sets if the stackTraceEnabled flag is on.
- **bool isTcpNoDelayEnabled** () const
Checks if the tcpNoDelayEnabled flag is on.
- **void setTcpNoDelayEnabled** (bool tcpNoDelayEnabled)
Sets if the tcpNoDelayEnabled flag is on.
- **int getVersion** () const
Get the current Wireformat Version.
- **void setVersion** (int version)
Set the current Wireformat Version.
- **virtual bool inReceive** () const
Is there a Message being unmarshaled?
- **bool isCacheEnabled** () const
Checks if the cacheEnabled flag is on.
- **void setCacheEnabled** (bool cacheEnabled)
Sets if the cacheEnabled flag is on.
- **int getCacheSize** () const
Returns the currently set Cache size.
- **void setCacheSize** (int value)
Sets the current Cache size.
- **bool isTightEncodingEnabled** () const

Checks if the tightEncodingEnabled flag is on.

- void **setTightEncodingEnabled** (bool tightEncodingEnabled)

Sets if the tightEncodingEnabled flag is on.

- bool **isSizePrefixDisabled** () const

Checks if the sizePrefixDisabled flag is on.

- void **setSizePrefixDisabled** (bool sizePrefixDisabled)

Sets if the sizePrefixDisabled flag is on.

- long long **getMaxInactivityDuration** () const

Gets the MaxInactivityDuration setting.

- void **setMaxInactivityDuration** (long long value)

Sets the MaxInactivityDuration setting.

- long long **getMaxInactivityDurationInitialDelay** () const

Gets the MaxInactivityDurationInitialDelay setting.

- void **setMaxInactivityDurationInitialDelay** (long long value)

Sets the MaxInactivityDurationInitialDelay setting.

Protected Member Functions

- **commands::DataStructure** * **doUnmarshal** (**decaf::io::DataInputStream** *dis)

Perform the actual unmarshal of data from the given DataInputStream return the unmarshalled DataStrucutre object once done, caller takes ownership of this object.

- void **destroyMarshallers** ()

Cleans up all registered Marshallers and empties the dataMarshallers vector.

Static Protected Attributes

- static const unsigned char **NULL_TYPE**
- static const int **DEFAULT_VERSION**
- static const int **MAX_SUPPORTED_VERSION**

6.389.1 Constructor & Destructor Documentation

6.389.1.1 **activemq::wireformat::openwire::OpenWireFormat::OpenWireFormat** (const **decaf::util::Properties** & *properties*)

Constructs a new **OpenWireFormat** (p. 2045) object.

Parameters

<i>properties</i>	- can contain optional config params.
-------------------	---------------------------------------

6.389.1.2 virtual **activemq::wireformat::openwire::OpenWireFormat::~OpenWireFormat**() [virtual]

6.389.2 Member Function Documentation

6.389.2.1 void **activemq::wireformat::openwire::OpenWireFormat::addMarshaller**(**marshal::DataStreamMarshaller** * *marshaller*)

Allows an external source to add marshalers to this object for types that may be marshaled or unmarshaled.

Parameters

<i>marshaller</i>	- the Marshaler to add to the collection.
-------------------	---

6.389.2.2 virtual **Pointer<transport::Transport>** **activemq::wireformat::openwire::OpenWireFormat::createNegotiator**(const **Pointer<transport::Transport>** & *transport*) [virtual]

If the Transport Provides a Negotiator this method will create and return a new instance of the Negotiator.

Parameters

<i>transport</i>	The Transport to Wrap the Negotiator around.
------------------	--

Returns

new instance of a **WireFormatNegotiator** (p.2904) as a **Pointer<Transport>** (p.2083).

Exceptions

<i>UnsupportedOperationException</i>	if the WireFormat (p. 2884) doesn't have a Negotiator.
--------------------------------------	---

Implements **activemq::wireformat::WireFormat** (p.2886).

6.389.2.3 void **activemq::wireformat::openwire::OpenWireFormat::destroyMarshalers**() [protected]

Cleans up all registered Marshallers and empties the dataMarshallers vector.

This should be called before a reconfiguration of the version marshallers, or on destruction of this object

6.389.2.4 `commands::DataStructure* activemq::wireformat::openwire::OpenWireFormat::doUnmarshal (decaf::io::DataInputStream * dis)`
`[protected]`

Perform the actual unmarshal of data from the given DataInputStream return the unmarshalled DataStrucutre object once done, caller takes ownership of this object.

This method can return null if the type of the object to unmarshal is NULL, empty data.

Parameters

<i>dis</i>	The DataInputStream to read from.
------------	-----------------------------------

Returns

new DataStructure* that the caller owns.

Exceptions

<i>IOException</i>	if an error occurs during the unmarshal.
--------------------	--

6.389.2.5 `int activemq::wireformat::openwire::OpenWireFormat::getCacheSize () const` `[inline]`

Returns the currently set Cache size.

Returns

the current value of the broker's cache size.

6.389.2.6 `long long activemq::wireformat::openwire::OpenWireFormat::getMaxInactivityDuration () const`
`[inline]`

Gets the MaxInactivityDuration setting.

Returns

maximum inactivity duration value in milliseconds.

6.389.2.7 `long long activemq::wireformat::openwire::OpenWireFormat::getMaxInactivityDurationInitialDelay () const`
`[inline]`

Gets the MaxInactivityDurationInitialDelay setting.

Returns

maximum inactivity duration initial delay value in milliseconds.

6.389.2.8 `const Pointer<commands::WireFormatInfo>& activemq::wireformat::openwire::OpenWireFormat::getPreferedWireFormatInfo () const [inline]`

Gets the Preferred WireFormatInfo object that this class holds.

Returns

pointer to a preferred WireFormatInfo object

6.389.2.9 `int activemq::wireformat::openwire::OpenWireFormat::getVersion () const [inline, virtual]`

Get the current Wireformat Version.

Returns

int that identifies the version

Implements **activemq::wireformat::WireFormat** (p. 2886).

6.389.2.10 `virtual bool activemq::wireformat::openwire::OpenWireFormat::hasNegotiator () const [inline, virtual]`

Returns true if this **WireFormat** (p. 2884) has a Negotiator that needs to wrap the Transport that uses it.

Returns

true if the **WireFormat** (p. 2884) provides a Negotiator.

Implements **activemq::wireformat::WireFormat** (p. 2886).

6.389.2.11 `virtual bool activemq::wireformat::openwire::OpenWireFormat::inReceive () const [inline, virtual]`

Is there a Message being unmarshaled?

Returns

true while in the doUnmarshal method.

Implements **activemq::wireformat::WireFormat** (p. 2887).

6.389.2.12 `bool activemq::wireformat::openwire::OpenWireFormat::isCacheEnabled () const [inline]`

Checks if the cacheEnabled flag is on.

Returns

true if the flag is on.

6.389.2.13 `bool activemq::wireformat::openwire::OpenWireFormat::isSizePrefixDisabled () const [inline]`

Checks if the sizePrefixDisabled flag is on.

Returns

true if the flag is on.

6.389.2.14 `bool activemq::wireformat::openwire::OpenWireFormat::isStackTraceEnabled () const [inline]`

Checks if the stackTraceEnabled flag is on.

Returns

true if the flag is on.

6.389.2.15 `bool activemq::wireformat::openwire::OpenWireFormat::isTcpNoDelayEnabled () const [inline]`

Checks if the tcpNoDelayEnabled flag is on.

Returns

true if the flag is on.

6.389.2.16 `bool activemq::wireformat::openwire::OpenWireFormat::isTightEncodingEnabled () const [inline]`

Checks if the tightEncodingEnabled flag is on.

Returns

true if the flag is on.

6.389 activemq::wireformat::openwire::OpenWireFormat Class Reference 2059

6.389.2.17 **void** **activemq::wireformat::openwire::OpenWireFormat::looseMarshalNestedObject** (**commands::DataStructure** * *o*,
decaf::io::DataOutputStream * *dataOut*)

Utility method to loosely Marshal an object that is derived from the DataStrucutre interface.

The marshaled data is written to the passed in DataOutputStream.

Parameters

<i>o</i>	- DataStructure derived Object to Marshal
<i>dataOut</i>	- DataOutputStream to write the data to

Exceptions

<i>IOException</i>	if an error occurs.
--------------------	---------------------

6.389.2.18 **commands::DataStructure*** **activemq::wireformat::openwire::OpenWireFormat::looseUnmarshalNestedObject** (
decaf::io::DataInputStream * *dis*)

Utility method to unmarshal an DataStructure object from an DataInputStream using the Loose Unmarshaling format.

Will read the Data and construct a new DataStructure based Object, the pointer to the Object returned is now owned by the caller.

Parameters

<i>dis</i>	- the DataInputStream to read the data from
------------	---

Returns

a new DataStructure derived Object pointer

Exceptions

<i>IOException</i>	if an error occurs.
--------------------	---------------------

6.389.2.19 **virtual void** **activemq::wireformat::openwire::OpenWireFormat::marshal**
(**const** **Pointer**< **commands::Command** > & *command*,
const **activemq::transport::Transport** * *transport*,
decaf::io::DataOutputStream * *out*) [virtual]

Stream based marshaling of a Command, this method blocks until the entire Command has been written out to the output stream.

Parameters

<i>command</i>	The Command to Marshal
<i>transport</i>	The Transport that called this method.
<i>out</i>	The output stream to write the command to.

Exceptions

<i>IOException</i>	if an I/O error occurs.
--------------------	-------------------------

Implements **activemq::wireformat::WireFormat** (p. 2887).

```
6.389.2.20 void activemq::wireformat::openwire::OpenWireFormat-
::renegotiateWireFormat ( const commands::WireFormatInfo & info
)
```

Called to re-negotiate the settings for the WireFormatInfo, these determine how the client and broker communicate.

Parameters

<i>info</i>	The new Wireformat Info settings.
-------------	-----------------------------------

Exceptions

<i>IllegalStateException</i>	is wire format can't be negotiated.
------------------------------	-------------------------------------

```
6.389.2.21 void activemq::wireformat::openwire::OpenWire-
Format::setCacheEnabled ( bool cacheEnabled )
[inline]
```

Sets if the cacheEnabled flag is on.

Parameters

<i>cache-Enabled</i>	- true to turn flag is on
----------------------	---------------------------

```
6.389.2.22 void activemq::wireformat::openwire::OpenWireFormat::setCacheSize (
int value ) [inline]
```

Sets the current Cache size.

Parameters

<i>value</i>	- the value to send as the broker's cache size.
--------------	---

6.389 activemq::wireformat::openwire::OpenWireFormat Class Reference 2061

6.389.2.23 `void activemq::wireformat::openwire::OpenWireFormat::setMaxInactivityDuration (long long value)`
[inline]

Sets the MaxInactivityDuration setting.

Parameters

<i>value</i>	- the Max inactivity duration value in milliseconds.
--------------	--

6.389.2.24 `void activemq::wireformat::openwire::OpenWireFormat::setMaxInactivityDurationInitialDelay (long long value)`
[inline]

Sets the MaxInactivityDurationInitialDelay setting.

Parameters

<i>value</i>	- the Max inactivity Initial Delay duration value in milliseconds.
--------------	--

6.389.2.25 `void activemq::wireformat::openwire::OpenWireFormat::setPreferredWireFormatInfo (const Pointer< commands::WireFormatInfo > & info)`

Configures this object using the provided WireformatInfo object.

Parameters

<i>info</i>	A WireFormatInfo object, takes ownership.
-------------	---

Exceptions

<i>IllegalStateException</i>	if the WireFormat (p. 2884) object has not been initialized.
------------------------------	---

6.389.2.26 `void activemq::wireformat::openwire::OpenWireFormat::setSizePrefixDisabled (bool sizePrefixDisabled)`
[inline]

Sets if the sizePrefixDisabled flag is on.

Parameters

<i>sizePrefixDisabled</i>	- true to turn flag is on
---------------------------	---------------------------

6.389.2.27 **void activemq::wireformat::openwire::OpenWireFormat::setStackTraceEnabled (bool *stackTraceEnabled*)**
[inline]

Sets if the stackTraceEnabled flag is on.

Parameters

<i>stackTrace-Enabled</i>	- true to turn flag is on
---------------------------	---------------------------

6.389.2.28 **void activemq::wireformat::openwire::OpenWireFormat::setTcpNoDelayEnabled (bool *tcpNoDelayEnabled*)**
[inline]

Sets if the tcpNoDelayEnabled flag is on.

Parameters

<i>tcpNoDelay-Enabled</i>	- true to turn flag is on
---------------------------	---------------------------

6.389.2.29 **void activemq::wireformat::openwire::OpenWireFormat::setTightEncodingEnabled (bool *tightEncodingEnabled*)**
[inline]

Sets if the tightEncodingEnabled flag is on.

Parameters

<i>tight-Encoding-Enabled</i>	- true to turn flag is on
-------------------------------	---------------------------

6.389.2.30 **void activemq::wireformat::openwire::OpenWireFormat::setVersion (int *version*)** [virtual]

Set the current Wireformat Version.

Parameters

<i>version</i>	An int that identifies the version
----------------	------------------------------------

Exceptions

<i>IllegalArgument-Exception</i>	if the version given is not supported.
----------------------------------	--

6.389 activemq::wireformat::openwire::OpenWireFormat Class Reference 2063

Implements **activemq::wireformat::WireFormat** (p. 2887).

6.389.2.31 **virtual int activemq::wireformat::openwire::OpenWireFormat::tightMarshalNestedObject1 (commands::DataStructure * *object*, utils::BooleanStream * *bs*)** [virtual]

Utility method for Tight Marshaling the given object to the boolean stream passed.

Parameters

<i>object</i>	- The DataStructure to marshal
<i>bs</i>	- the BooleanStream to write to

Returns

size of the data returned.

6.389.2.32 **void activemq::wireformat::openwire::OpenWireFormat::tightMarshalNestedObject2 (commands::DataStructure * *o*, decaf::io::DataOutputStream * *ds*, utils::BooleanStream * *bs*)**

Utility method that will Tight marshal some internally nested object that implements the DataStructure interface.

Writes the data to the Data Output Stream provided.

Parameters

<i>o</i>	- DataStructure object
<i>ds</i>	- DataOutputStream for writing
<i>bs</i>	- BooleanStream

Exceptions

<i>IOException</i>	if an error occurs.
--------------------	---------------------

6.389.2.33 **commands::DataStructure* activemq::wireformat::openwire::OpenWireFormat::tightUnmarshalNestedObject (decaf::io::DataInputStream * *dis*, utils::BooleanStream * *bs*)**

Utility method used to Unmarshal a Nested DataStructure type object from the given DataInputStream.

The DataStructure instance that is returned is now the property of the caller.

Parameters

<i>dis</i>	- DataInputStream to read from
<i>bs</i>	- BooleanStream to read from

Returns

Newly allocated DataStructure Object

Exceptions

<i>IOException</i>	if an error occurs.
--------------------	---------------------

6.389.2.34 **virtual Pointer<commands::Command> activemq-
::wireformat::openwire::OpenWireFormat::unmarshal (const
activemq::transport::Transport * *transport*, decaf::io::DataInputStream
* *in*) [virtual]**

Stream based unmarshaling, blocks on reads on the input stream until a complete command has been read and unmarshaled into the correct form.

Returns a Pointer to the newly unmarshaled Command.

Parameters

<i>transport</i>	Pointer to the transport that is making this request.
<i>in</i>	The input stream to read the command from.

Returns

the newly marshaled Command, caller owns the pointer

Exceptions

<i>IOException</i>	if an I/O error occurs.
--------------------	-------------------------

Implements **activemq::wireformat::WireFormat** (p. 2888).

6.389.3 Field Documentation

6.389.3.1 **const int activemq::wireformat::openwire::Open-
WireFormat::DEFAULT_VERSION [static,
protected]**

6.390 activemq::wireformat::openwire::OpenWireFormatFactory Class Reference

6.389.3.2 `const int activemq::wireformat::openwire::OpenWireFormat::MAX_SUPPORTED_VERSION` [static, protected]

6.389.3.3 `const unsigned char activemq::wireformat::openwire::OpenWireFormat::NULL_TYPE` [static, protected]

The documentation for this class was generated from the following file:

- `src/main/activemq/wireformat/openwire/OpenWireFormat.h`

6.390 activemq::wireformat::openwire::OpenWireFormatFactory Class Reference

```
#include <src/main/activemq/wireformat/openwire/OpenWireFormatFactory.h>
```

Inheritance diagram for `activemq::wireformat::openwire::OpenWireFormatFactory`:

Public Member Functions

- **OpenWireFormatFactory** ()
Constructor - Sets Defaults for all properties, these are all subject to change once the `createWireFormat` method is called.
- virtual `~OpenWireFormatFactory` ()
- virtual `Pointer < wireformat::WireFormat > createWireFormat (const decaf::util::Properties &properties)`
*Creates a new **WireFormat** (p. 2884) Object passing it a set of properties from which it can obtain any optional settings.*

6.390.1 Constructor & Destructor Documentation

6.390.1.1 `activemq::wireformat::openwire::OpenWireFormatFactory::OpenWireFormatFactory ()` [inline]

Constructor - Sets Defaults for all properties, these are all subject to change once the `createWireFormat` method is called.

URL options ----- `wireFormat.stackTraceEnabled` `wireFormat.cacheEnabled`
`wireFormat.tcpNoDelayEnabled` `wireFormat.tightEncodingEnabled` `wireFormat.sizePrefixDisabled` `wireFormat.maxInactivityDuration` `wireFormat.maxInactivityDurationInitialDelay`

6.390.1.2 `virtual activemq::wireformat::openwire::OpenWireFormatFactory::~~OpenWireFormatFactory () [inline, virtual]`

6.390.2 Member Function Documentation

6.390.2.1 `virtual Pointer<wireformat::WireFormat> activemq::wireformat::openwire::OpenWireFormatFactory::createWireFormat (const decaf::util::Properties & properties) [virtual]`

Creates a new **WireFormat** (p. 2884) Object passing it a set of properties from which it can obtain any optional settings.

Parameters

<i>properties</i>	The Properties for this WireFormat (p. 2884).
-------------------	--

Returns

Pointer to a new instance of a **WireFormat** (p. 2884) object.

Exceptions

<i>IllegalStateException</i>	if the factory has not been initialized.
------------------------------	--

Implements **activemq::wireformat::WireFormatFactory** (p. 2889).

The documentation for this class was generated from the following file:

- `src/main/activemq/wireformat/openwire/OpenWireFormatFactory.h`

6.391 `activemq::wireformat::openwire::OpenWireFormatNegotiator` Class Reference

```
#include <src/main/activemq/wireformat/openwire/OpenWireFormatNegotiator.h>
```

Inheritance diagram for `activemq::wireformat::openwire::OpenWireFormatNegotiator`:

Public Member Functions

- **OpenWireFormatNegotiator** (**OpenWireFormat** *wireFormat, const **Pointer**<**transport::Transport** > &next)

Constructor - Initializes this object around another Transport.

- virtual **~OpenWireFormatNegotiator** ()
- virtual void **oneway** (const **Pointer**< **commands::Command** > &command) throw (decaf::io::IOException, decaf::lang::exceptions::UnsupportedOperationException)
Sends a one-way command.
- virtual **Pointer** < **commands::Response** > **request** (const **Pointer**< **commands::Command** > &command) throw (decaf::io::IOException, decaf::lang::exceptions::UnsupportedOperationException)
Sends the given request to the server and waits for the response.
- virtual **Pointer** < **commands::Response** > **request** (const **Pointer**< **commands::Command** > &command, unsigned int timeout) throw (decaf::io::IOException, decaf::lang::exceptions::UnsupportedOperationException)
Sends the given request to the server and waits for the response.
- virtual void **onCommand** (const **Pointer**< **commands::Command** > &command)
This is called in the context of the nested transport's reading thread.
- virtual void **onTransportException** (**transport::Transport** *source, const **decaf::lang::Exception** &ex)
Event handler for an exception from a command transport.
- virtual void **start** () throw (decaf::io::IOException)
Starts this transport object and creates the thread for polling on the input stream for commands.
- virtual void **close** () throw (decaf::io::IOException)
Stops the polling thread and closes the streams.

6.391.1 Constructor & Destructor Documentation

- 6.391.1.1 **activemq::wireformat::openwire::OpenWireFormatNegotiator::OpenWireFormatNegotiator** (**OpenWireFormat** * *wireFormat*, const **Pointer**< **transport::Transport** > & *next*)

Constructor - Initializes this object around another Transport.

Parameters

<i>wireFormat</i>	- The WireFormat (p. 2884) object we use to negotiate
<i>next</i>	- The next transport in the chain

- 6.391.1.2 **virtual activemq::wireformat::openwire::OpenWireFormatNegotiator::~OpenWireFormatNegotiator** ()
[virtual]

6.391.2 Member Function Documentation

6.391.2.1 `virtual void activemq::wireformat::openwire::OpenWireFormatNegotiator::close () throw (decaf::io::IOException)`
`[virtual]`

Stops the polling thread and closes the streams.

This can be called explicitly, but is also called in the destructor. Once this object has been closed, it cannot be restarted.

Exceptions

<i>IOException</i>	if errors occur.
--------------------	------------------

Reimplemented from `activemq::transport::TransportFilter` (p. 2802).

6.391.2.2 `virtual void activemq::wireformat::openwire::OpenWireFormatNegotiator::onCommand (const Pointer< commands::Command > & command)`
`[virtual]`

This is called in the context of the nested transport's reading thread.

In the case of a response object, updates the request map and notifies those waiting on the response. Non-response messages are just delegated to the command listener.

Parameters

<i>command</i>	the received from the nested transport.
----------------	---

Reimplemented from `activemq::transport::TransportFilter` (p. 2805).

6.391.2.3 `virtual void activemq::wireformat::openwire::OpenWireFormatNegotiator::oneway (const Pointer< commands::Command > & command) throw (decaf::io::IOException, decaf::lang::exceptions::UnsupportedOperationException)`
`[virtual]`

Sends a one-way command.

Does not wait for any response from the broker. First waits for the WireFormatInfo exchange to happen so that we know how to encode out-bound data.

Parameters

<i>command</i>	the command to be sent.
----------------	-------------------------

Exceptions

<i>IOException</i>	if an exception occurs during writing of the command.
<i>Unsupported-OperationException</i>	if this method is not implemented by this transport.

Reimplemented from **activemq::transport::TransportFilter** (p. 2806).

6.391.2.4 `virtual void activemq::wireformat::openwire::OpenWireFormatNegotiator::onTransportException (transport::Transport * source, const decaf::lang::Exception & ex) [virtual]`

Event handler for an exception from a command transport.

Parameters

<i>source</i>	The source of the exception
<i>ex</i>	The exception.

6.391.2.5 `virtual Pointer<commands::Response> activemq::wireformat::openwire::OpenWireFormatNegotiator::request (const Pointer< commands::Command > & command) throw (decaf::io::IOException, decaf::lang::exceptions::UnsupportedOperationException) [virtual]`

Sends the given request to the server and waits for the response.

First waits for the WireFormatInfo exchange to happen so that we know how to encode out-bound data.

Parameters

<i>command</i>	The request to send.
----------------	----------------------

Returns

the response from the server.

Exceptions

<i>IOException</i>	if an error occurs with the request.
--------------------	--------------------------------------

Reimplemented from **activemq::transport::TransportFilter** (p. 2807).

6.391.2.6 `virtual Pointer<commands::Response> activemq::wireformat::openwire::OpenWireFormatNegotiator::request (const Pointer< commands::Command > & command, unsigned int timeout) throw (decaf::io::IOException, decaf::lang::exceptions::UnsupportedOperationException) [virtual]`

Sends the given request to the server and waits for the response.

First waits for the WireFormatInfo exchange to happen so that we know how to encode out-bound data.

Parameters

<i>command</i>	The request to send.
<i>timeout</i>	The time to wait for the response.

Returns

the response from the server.

Exceptions

<i>IOException</i>	if an error occurs with the request.
--------------------	--------------------------------------

Reimplemented from **activemq::transport::TransportFilter** (p. 2807).

```
6.391.2.7 virtual void activemq::wireformat::openwire::OpenWire-
FormatNegotiator::start ( ) throw ( decaf::io::IOException )
[virtual]
```

Starts this transport object and creates the thread for polling on the input stream for commands.

If this object has been closed, throws an exception. Before calling start, the caller must set the IO streams and the reader and writer objects.

Exceptions

<i>IOException</i>	if an error occurs or if this transport has already been closed.
--------------------	--

Reimplemented from **activemq::transport::TransportFilter** (p. 2808).

The documentation for this class was generated from the following file:

- src/main/activemq/wireformat/openwire/**OpenWireFormatNegotiator.h**

6.392 activemq::wireformat::openwire::OpenWireResponseBuilder Class Reference

Used to allow a MockTransport to generate response commands to OpenWire - Commands.

```
#include <src/main/activemq/wireformat/openwire/OpenWire-
ResponseBuilder.h>
```

Inheritance diagram for activemq::wireformat::openwire::OpenWireResponseBuilder:

Public Member Functions

- **OpenWireResponseBuilder** ()
- virtual **~OpenWireResponseBuilder** ()
- virtual **Pointer** < **commands::Response** > **buildResponse** (const **Pointer**< **commands::Command** > &command)

Given a Command, check if it requires a response and return the appropriate - Response that the Broker would send for this Command.

- virtual void **buildIncomingCommands** (const **Pointer**< **commands::Command** > &command, **decaf::util::LinkedList**< **Pointer**< **commands::Command** > > &queue)

When called the ResponseBuilder must construct all the Responses or Asynchronous commands that would be sent to this client by the Broker upon receipt of the passed command.

6.392.1 Detailed Description

Used to allow a MockTransport to generate response commands to OpenWire - Commands.

6.392.2 Constructor & Destructor Documentation

6.392.2.1 **activemq::wireformat::openwire::OpenWireResponseBuilder::OpenWireResponseBuilder** () [inline]

6.392.2.2 virtual **activemq::wireformat::openwire::OpenWireResponseBuilder::~~OpenWireResponseBuilder** () [inline, virtual]

6.392.3 Member Function Documentation

6.392.3.1 virtual void **activemq::wireformat::openwire::OpenWireResponseBuilder::buildIncomingCommands** (const **Pointer**< **commands::Command** > &command, **decaf::util::LinkedList**< **Pointer**< **commands::Command** > > &queue) [virtual]

When called the ResponseBuilder must construct all the Responses or Asynchronous commands that would be sent to this client by the Broker upon receipt of the passed command.

Parameters

<i>command</i>	- The Command being sent to the Broker.
<i>queue</i>	- Queue of Command sent back from the broker.

Implements **activemq::transport::mock::ResponseBuilder** (p. 2302).

6.392.3.2 **virtual** **Pointer**<**commands::Response**> **activemq::wireformat::openwire::OpenWireResponseBuilder::buildResponse** (**const** **Pointer**< **commands::Command** > & *command*) [virtual]

Given a Command, check if it requires a response and return the appropriate Response that the Broker would send for this Command.

Parameters

<i>command</i>	- The command to build a response for
----------------	---------------------------------------

Returns

A Response object pointer, or NULL if no response.

Implements **activemq::transport::mock::ResponseBuilder** (p. 2303).

The documentation for this class was generated from the following file:

- src/main/activemq/wireformat/openwire/**OpenWireResponseBuilder.h**

6.393 decaf::io::OutputStream Class Reference

Base interface for any class that wants to represent an output stream of bytes.

```
#include <src/main/decaf/io/OutputStream.h>
```

Inheritance diagram for decaf::io::OutputStream:

Public Member Functions

- **OutputStream** ()
- virtual **~OutputStream** ()
- virtual void **close** ()

*Closes this object and deallocates the appropriate resources.
The object is generally no longer usable after calling close.*

Exceptions

IOException (p. 1545)	<i>if an error occurs while closing.</i>
------------------------------	--

- virtual void **flush** ()

Flushes this stream by writing any buffered output to the underlying stream.

Exceptions

IOException (p. 1545)	<i>if an I/O error occurs.</i>
------------------------------	--------------------------------

- virtual void **write** (unsigned char c)

Writes a single byte to the output stream.

- virtual void **write** (const unsigned char *buffer, int size)

Writes an array of bytes to the output stream.

- virtual void **write** (const unsigned char *buffer, int size, int offset, int length)

Writes an array of bytes to the output stream in order starting at buffer[offset] and proceeding until the number of bytes specified by the length argument are written or an error occurs.

- virtual std::string **toString** () const

Output a String representation of this object.

- virtual void **lock** ()

Locks the object.

- virtual bool **tryLock** ()

Attempts to Lock the object, if the lock is already held by another thread than this method returns false.

- virtual void **unlock** ()

Unlocks the object.

- virtual void **wait** ()

Waits on a signal from this object, which is generated by a call to Notify.

- virtual void **wait** (long long millisecs)

Waits on a signal from this object, which is generated by a call to Notify.

- virtual void **wait** (long long millisecs, int nanos)

Waits on a signal from this object, which is generated by a call to Notify.

- virtual void **notify** ()

Signals a waiter on this object that it can now wake up and continue.

- virtual void **notifyAll** ()

Signals the waiters on this object that it can now wake up and continue.

Protected Member Functions

- virtual void **doWriteByte** (unsigned char value)=0

- virtual void **doWriteArray** (const unsigned char *buffer, int size)

- virtual void **doWriteArrayBounded** (const unsigned char *buffer, int size, int offset, int length)

6.393.1 Detailed Description

Base interface for any class that wants to represent an output stream of bytes.

Since

1.0

6.393.2 Constructor & Destructor Documentation

6.393.2.1 `decaf::io::OutputStream::OutputStream ()`

6.393.2.2 `virtual decaf::io::OutputStream::~~OutputStream ()` [virtual]

6.393.3 Member Function Documentation

6.393.3.1 `virtual void decaf::io::OutputStream::close ()` [virtual]

Closes this object and deallocates the appropriate resources.

The object is generally no longer usable after calling close.

Exceptions

<i>IOException</i> (p. 1545)	if an error occurs while closing.
--	-----------------------------------

The default implementation of this method does nothing.

Implements **decaf::io::Closeable** (p. 817).

Reimplemented in **decaf::util::zip::DeflaterOutputStream** (p. 1192), **decaf::io::FilterOutputStream** (p. 1342), **decaf::internal::net::tcp::TcpSocketOutputStream** (p. 2692), **decaf::internal::net::ssl::openssl::OpenSSLSocketOutputStream** (p. 2044), **decaf::internal::io::StandardErrorOutputStream** (p. 2529), and **decaf::internal::io::StandardOutputStream** (p. 2533).

6.393.3.2 `virtual void decaf::io::OutputStream::doWriteArray (const unsigned char * buffer, int size)` [protected, virtual]

Reimplemented in **decaf::io::FilterOutputStream** (p. 1343), and **decaf::io::BufferedOutputStream** (p. 596).

6.393.3.3 `virtual void decaf::io::OutputStream::doWriteArrayBounded (const unsigned char * buffer, int size, int offset, int length)` [protected, virtual]

Reimplemented in **decaf::util::zip::DeflaterOutputStream** (p. 1193), **decaf::io::DataOutputStream** (p. 1110), **decaf::io::ByteArrayOutputStream** (p. 689), **decaf::**

decaf::io::FilterOutputStream (p. 1343), **decaf::io::BufferedOutputStream** (p. 596), **decaf::util::zip::CheckedOutputStream** (p. 809), **decaf::internal::net::tcp::TcpSocketOutputStream** (p. 2692), **decaf::internal::net::ssl::openssl::OpenSSLSocketOutputStream** (p. 2045), **activemq::io::LoggingOutputStream** (p. 1708), **decaf::internal::io::StandardErrorOutputStream** (p. 2530), and **decaf::internal::io::StandardOutputStream** (p. 2533).

6.393.3.4 **virtual void decaf::io::OutputStream::doWriteByte (unsigned char value)**
[protected, pure virtual]

Implemented in **decaf::util::zip::DeflaterOutputStream** (p. 1193), **decaf::io::DataOutputStream** (p. 1111), **decaf::io::ByteArrayOutputStream** (p. 689), **decaf::io::FilterOutputStream** (p. 1343), **decaf::io::BufferedOutputStream** (p. 597), **decaf::util::zip::CheckedOutputStream** (p. 810), **decaf::internal::net::tcp::TcpSocketOutputStream** (p. 2693), **decaf::internal::net::ssl::openssl::OpenSSLSocketOutputStream** (p. 2045), **activemq::io::LoggingOutputStream** (p. 1708), **decaf::internal::io::StandardErrorOutputStream** (p. 2530), and **decaf::internal::io::StandardOutputStream** (p. 2533).

6.393.3.5 **virtual void decaf::io::OutputStream::flush ()** [virtual]

Flushes this stream by writing any buffered output to the underlying stream.

Exceptions

<i>IOException</i> (p. 1545)	if an I/O error occurs.
--	-------------------------

The default implementation of this method does nothing.

Implements **decaf::io::Flushable** (p. 1380).

Reimplemented in **decaf::io::BufferedOutputStream** (p. 597), **decaf::io::FilterOutputStream** (p. 1343), **decaf::internal::io::StandardErrorOutputStream** (p. 2530), and **decaf::internal::io::StandardOutputStream** (p. 2533).

6.393.3.6 **virtual void decaf::io::OutputStream::lock ()** [inline, virtual]

Locks the object.

Exceptions

<i>RuntimeException</i>	if an error occurs while locking the object.
--------------------------------	--

Implements **decaf::util::concurrent::Synchronizable** (p. 2640).

6.393.3.7 `virtual void decaf::io::OutputStream::notify () [inline, virtual]`

Signals a waiter on this object that it can now wake up and continue.

Must have this object locked before calling.

Exceptions

<i>IllegalMonitorStateException</i>	- if the current thread is not the owner of the the Synchronizable Object.
<i>RuntimeException</i>	if an error occurs while notifying one of the waiting threads.

Implements **decaf::util::concurrent::Synchronizable** (p. 2641).

6.393.3.8 `virtual void decaf::io::OutputStream::notifyAll () [inline, virtual]`

Signals the waiters on this object that it can now wake up and continue.

Must have this object locked before calling.

Exceptions

<i>IllegalMonitorStateException</i>	- if the current thread is not the owner of the the Synchronizable Object.
<i>RuntimeException</i>	if an error occurs while notifying the waiting threads.

Implements **decaf::util::concurrent::Synchronizable** (p. 2642).

6.393.3.9 `virtual std::string decaf::io::OutputStream::toString () const [virtual]`

Output a String representation of this object.

The default version of this method just prints the Class Name.

Returns

a string representation of the object.

Reimplemented in **decaf::io::ByteArrayOutputStream** (p. 690), and **decaf::io::FilterOutputStream** (p. 1344).

6.393.3.10 `virtual bool decaf::io::OutputStream::tryLock () [inline, virtual]`

Attempts to Lock the object, if the lock is already held by another thread than this method returns false.

Returns

true if the lock was acquired, false if it is already held by another thread.

Exceptions

<i>RuntimeException</i>	if an error occurs while locking the object.
-------------------------	--

Implements **decaf::util::concurrent::Synchronizable** (p. 2643).

6.393.3.11 virtual void **decaf::io::OutputStream::unlock** () [*inline*, *virtual*]

Unlocks the object.

Exceptions

<i>RuntimeException</i>	if an error occurs while unlocking the object.
-------------------------	--

Implements **decaf::util::concurrent::Synchronizable** (p. 2645).

6.393.3.12 virtual void **decaf::io::OutputStream::wait** () [*inline*, *virtual*]

Waits on a signal from this object, which is generated by a call to Notify.

Must have this object locked before calling.

Exceptions

<i>RuntimeException</i>	if an error occurs while waiting on the object.
<i>InterruptedException</i>	if the wait is interrupted before it completes.
<i>IllegalMonitorStateException</i>	- if the current thread is not the owner of the the Synchronizable Object.

Implements **decaf::util::concurrent::Synchronizable** (p. 2646).

6.393.3.13 virtual void **decaf::io::OutputStream::wait** (long long *millisecs*) [*inline*, *virtual*]

Waits on a signal from this object, which is generated by a call to Notify.

Must have this object locked before calling. This wait will timeout after the specified time interval.

Parameters

<i>millisecs</i>	the time in milliseconds to wait, or WAIT_INFINITE
------------------	--

Exceptions

<i>RuntimeException</i>	if an error occurs while waiting on the object.
<i>InterruptedException</i>	if the wait is interrupted before it completes.
<i>IllegalMonitorStateException</i>	- if the current thread is not the owner of the the Synchronizable Object.

Implements **decaf::util::concurrent::Synchronizable** (p. 2647).

6.393.3.14 `virtual void decaf::io::OutputStream::wait (long long millisecs, int nanos)`
`[inline, virtual]`

Waits on a signal from this object, which is generated by a call to Notify.

Must have this object locked before calling. This wait will timeout after the specified time interval. This method is similar to the one argument wait function except that it add a finer grained control over the amount of time that it waits by adding in the additional nanosecond argument.

NOTE: The ability to wait accurately at a nanosecond scale depends on the platform and OS that the Decaf API is running on, some systems do not provide an accurate enough clock to provide this level of granularity.

Parameters

<i>millisecs</i>	the time in milliseconds to wait, or WAIT_INFINITE
<i>nanos</i>	additional time in nanoseconds with a range of 0-999999

Exceptions

<i>IllegalArgument-Exception</i>	if an error occurs or the nanos argument is not in the range of [0-999999]
<i>RuntimeException</i>	if an error occurs while waiting on the object.
<i>InterruptedException</i>	if the wait is interrupted before it completes.
<i>IllegalMonitorStateException</i>	- if the current thread is not the owner of the the Synchronizable Object.

Implements **decaf::util::concurrent::Synchronizable** (p. 2648).

6.393.3.15 `virtual void decaf::io::OutputStream::write (unsigned char c)`
`[virtual]`

Writes a single byte to the output stream.

The default implementation of this method calls the pure virtual method doWriteByte which must be implemented by any subclass of the **OutputStream** (p. 2066).

Parameters

<i>c</i>	The byte to write to the sink.
----------	--------------------------------

Exceptions

<i>IOException</i> (p. 1545)	if an I/O error occurs.
--	-------------------------

6.393.3.16 virtual void decaf::io::OutputStream::write (const unsigned char * *buffer*, int *size*) [virtual]

Writes an array of bytes to the output stream.

The entire contents of the given vector are written to the output stream.

The default implementation of this method simply calls the doWriteArray which writes the contents of the array using the doWriteByte method repeatedly. It is recommended that a subclass override doWriteArray to provide more performant means of writing the array.

Parameters

<i>buffer</i>	The vector of bytes to write.
<i>size</i>	The size of the buffer passed.

Exceptions

<i>IOException</i> (p. 1545)	if an I/O error occurs.
<i>NullPointerException</i>	thrown if buffer is Null.
<i>IndexOutOfBoundsException</i>	if size value is negative.

6.393.3.17 virtual void decaf::io::OutputStream::write (const unsigned char * *buffer*, int *size*, int *offset*, int *length*) [virtual]

Writes an array of bytes to the output stream in order starting at buffer[offset] and proceeding until the number of bytes specified by the length argument are written or an error occurs.

The default implementation of this method simply calls the doWriteArrayBounded method which writes the contents of the array using the doWriteByte method repeatedly. It is recommended that a subclass override doWriteArrayBounded to provide more performant means of writing the array.

Parameters

<i>buffer</i>	The array of bytes to write.
<i>size</i>	The size of the buffer array passed.
<i>offset</i>	The position to start writing in buffer.
<i>length</i>	The number of bytes from the buffer to be written.

Exceptions

<i>IOException</i> (p. 1545)	if an I/O error occurs.
<i>NullPointerException</i>	thrown if buffer is Null.
<i>IndexOutOfBoundsException</i>	if the offset + length > size. or one of the parameters is negative.

The documentation for this class was generated from the following file:

- src/main/decaf/io/**OutputStream.h**

6.394 decaf::io::OutputStreamWriter Class Reference

A class for turning a character stream into a byte stream.

```
#include <src/main/decaf/io/OutputStreamWriter.h>
```

Inheritance diagram for decaf::io::OutputStreamWriter:

Public Member Functions

- **OutputStreamWriter** (**OutputStream** *stream, bool own=false)
*Creates a new **OutputStreamWriter** (p. 2074).*
- virtual ~**OutputStreamWriter** ()
- virtual void **close** ()
Closes this object and deallocates the appropriate resources.
- virtual void **flush** ()
Flushes this stream by writing any buffered output to the underlying stream.

Protected Member Functions

- virtual void **doWriteArrayBounded** (const char *buffer, int size, int offset, int length)
Override this method to customize the functionality of the method write(char buffer, int size, int offset, int length).*
- virtual void **checkClosed** () const

6.394.1 Detailed Description

A class for turning a character stream into a byte stream.

See also

InputStreamReader (p. 1475)

Since

1.0

6.394.2 Constructor & Destructor Documentation

6.394.2.1 **decaf::io::OutputStreamWriter::OutputStreamWriter** (**OutputStream** * *stream*, bool *own* = false)

Creates a new **OutputStreamWriter** (p. 2074).

Parameters

<i>stream</i>	The OutputStream (p. 2066) to wrap. (cannot be NULL).
<i>own</i>	Indicates whether this instance own the given OutputStream (p. 2066). If true then the OutputStream (p. 2066) is destroyed when this class is.

Exceptions

<i>NullPointerException</i>	if the stream is NULL.
-----------------------------	------------------------

6.394.2.2 **virtual decaf::io::OutputStreamWriter::~~OutputStreamWriter** ()
[virtual]

6.394.3 Member Function Documentation

6.394.3.1 **virtual void decaf::io::OutputStreamWriter::checkClosed** () const
[protected, virtual]

6.394.3.2 **virtual void decaf::io::OutputStreamWriter::close** () [virtual]

Closes this object and deallocates the appropriate resources.

The object is generally no longer usable after calling close.

Exceptions

IOException (p. 1545)	if an error occurs while closing.
---------------------------------	-----------------------------------

Implements **decaf::io::Closeable** (p. 817).

6.394.3.3 **virtual void decaf::io::OutputStreamWriter::doWriteArrayBounded (const char * *buffer*, int *size*, int *offset*, int *length*)** [protected, virtual]

Override this method to customize the functionality of the method `write(char* buffer, int size, int offset, int length)`.

All subclasses must override this method to provide the basic **Writer** (p. 2908) functionality.

Implements **decaf::io::Writer** (p. 2911).

6.394.3.4 **virtual void decaf::io::OutputStreamWriter::flush ()** [virtual]

Flushes this stream by writing any buffered output to the underlying stream.

Exceptions

<i>IOException</i> (p. 1545)	if an I/O error occurs.
--	-------------------------

Implements **decaf::io::Flushable** (p. 1380).

The documentation for this class was generated from the following file:

- `src/main/decaf/io/OutputStreamWriter.h`

6.395 activemq::commands::PartialCommand Class Reference

```
#include <src/main/activemq/commands/PartialCommand.h>
```

Inheritance diagram for `activemq::commands::PartialCommand`:

Public Member Functions

- **PartialCommand ()**
- virtual **~PartialCommand ()**
- virtual unsigned char **getDataStructureType ()** const
*Get the **DataStructure** (p. 1133) Type as defined in *CommandTypes.h*.*
- virtual **PartialCommand * cloneDataStructure ()** const
Clone this object and return a new instance that the caller now owns, this will be an exact copy of this one.
- virtual void **copyDataStructure (const DataStructure *src)**

Copy the contents of the passed object into this objects members, overwriting any existing data.

- virtual std::string **toString** () const

*Returns a string containing the information for this **DataStructure** (p. 1133) such as its type and value of its elements.*

- virtual bool **equals** (const **DataStructure** *value) const

*Compares the **DataStructure** (p. 1133) passed in to this one, and returns if they are equivalent.*

- virtual int **getCommandId** () const
- virtual void **setCommandId** (int commandId)
- virtual const std::vector< unsigned char > & **getData** () const
- virtual std::vector< unsigned char > & **getData** ()
- virtual void **setData** (const std::vector< unsigned char > &data)

Static Public Attributes

- static const unsigned char **ID_PARTIALCOMMAND** = 60

Protected Attributes

- int **commandId**
- std::vector< unsigned char > **data**

6.395.1 Constructor & Destructor Documentation

6.395.1.1 **activemq::commands::PartialCommand::PartialCommand** ()

6.395.1.2 **virtual activemq::commands::PartialCommand::~~PartialCommand** ()
[virtual]

6.395.2 Member Function Documentation

6.395.2.1 **virtual PartialCommand* activemq::commands::-**
PartialCommand::cloneDataStructure () const
[virtual]

Clone this obbject and return a new instance that the caller now owns, this will be an exact copy of this one.

Returns

new copy of this object.

Implements **activemq::commands::DataStructure** (p. 1133).

Reimplemented in **activemq::commands::LastPartialCommand** (p. 1606).

6.395.2.2 `virtual void activemq::commands::PartialCommand::copyDataStructure (const DataStructure * src) [virtual]`

Copy the contents of the passed object into this objects members, overwriting any existing data.

Returns

src - Source Object

Implements **activemq::commands::DataStructure** (p. 1134).

Reimplemented in **activemq::commands::LastPartialCommand** (p. 1607).

6.395.2.3 `virtual bool activemq::commands::PartialCommand::equals (const DataStructure * value) const [virtual]`

Compares the **DataStructure** (p. 1133) passed in to this one, and returns if they are equivalent.

Equivalent here means that they are of the same type, and that each element of the objects are the same.

Returns

true if **DataStructure** (p. 1133)'s are Equal.

Implements **activemq::commands::DataStructure** (p. 1135).

Reimplemented in **activemq::commands::LastPartialCommand** (p. 1607).

6.395.2.4 `virtual int activemq::commands::PartialCommand::getCommandId () const [virtual]`

6.395.2.5 `virtual const std::vector<unsigned char>& activemq::commands::PartialCommand::getData () const [virtual]`

6.395.2.6 `virtual std::vector<unsigned char>& activemq::commands::PartialCommand::getData () [virtual]`

6.395.2.7 `virtual unsigned char activemq::commands::PartialCommand::getDataStructureType () const [virtual]`

Get the **DataStructure** (p. 1133) Type as defined in CommandTypes.h.

Returns

The type of the data structure

Implements **activemq::commands::DataStructure** (p. 1137).

Reimplemented in **activemq::commands::LastPartialCommand** (p. 1607).

6.396 activemq::wireformat::openwire::marshal::generated::PartialCommandMarshaller Class

Reference

2085

6.395.2.8 virtual void **activemq::commands::PartialCommand::setCommandId** (int *commandId*) [virtual]

6.395.2.9 virtual void **activemq::commands::PartialCommand::setData** (const std::vector< unsigned char > & *data*) [virtual]

6.395.2.10 virtual std::string **activemq::commands::PartialCommand::toString** () const [virtual]

Returns a string containing the information for this **DataStructure** (p. 1133) such as its type and value of its elements.

Returns

formatted string useful for debugging.

Reimplemented from **activemq::commands::BaseDataStructure** (p. 531).

Reimplemented in **activemq::commands::LastPartialCommand** (p. 1608).

6.395.3 Field Documentation

6.395.3.1 int **activemq::commands::PartialCommand::commandId** [protected]

6.395.3.2 std::vector<unsigned char> **activemq::commands::PartialCommand::data** [protected]

6.395.3.3 const unsigned char **activemq::commands::PartialCommand::ID_PARTIALCOMMAND = 60** [static]

The documentation for this class was generated from the following file:

- src/main/activemq/commands/**PartialCommand.h**

6.396 activemq::wireformat::openwire::marshal::generated::PartialCommandMarshaller Class Reference

Marshaling code for Open Wire Format for **PartialCommandMarshaller** (p. 2079).

```
#include <src/main/activemq/wireformat/openwire/marshal/generated/-  
PartialCommandMarshaller.h>
```

Inheritance diagram for **activemq::wireformat::openwire::marshal::generated::PartialCommandMarshaller**:

Public Member Functions

- **PartialCommandMarshaller** ()
- virtual **~PartialCommandMarshaller** ()
- virtual **commands::DataStructure * createObject** () const
Creates a new instance of the class that this class is a marshaling director for.
- virtual unsigned char **getDataStructureType** () const
Gets the DataStructureType that this class marshals/unmarshals.
- virtual void **tightUnmarshal** (**OpenWireFormat *wireFormat**, **commands::DataStructure *dataStructure**, **decaf::io::DataInputStream *dataIn**, **utils::BooleanStream *bs**)
Tight Un-marhsal to the given stream.
- virtual int **tightMarshal1** (**OpenWireFormat *wireFormat**, **commands::DataStructure *dataStructure**, **utils::BooleanStream *bs**)
Tight Marhsal to the given stream.
- virtual void **tightMarshal2** (**OpenWireFormat *wireFormat**, **commands::DataStructure *dataStructure**, **decaf::io::DataOutputStream *dataOut**, **utils::BooleanStream *bs**)
Tight Marhsal to the given stream.
- virtual void **looseUnmarshal** (**OpenWireFormat *wireFormat**, **commands::DataStructure *dataStructure**, **decaf::io::DataInputStream *dataIn**)
Loose Un-marhsal to the given stream.
- virtual void **looseMarshal** (**OpenWireFormat *wireFormat**, **commands::DataStructure *dataStructure**, **decaf::io::DataOutputStream *dataOut**)
Tight Marhsal to the given stream.

6.396.1 Detailed Description

Marshaling code for Open Wire Format for **PartialCommandMarshaller** (p. 2079).

NOTE!: This file is auto generated - do not modify! if you need to make a change, please see the Java Classes in the activemq-openwire-generator module

6.396.2 Constructor & Destructor Documentation

- 6.396.2.1 **activemq::wireformat::openwire::marshal::generated::PartialCommandMarshaller::PartialCommandMarshaller** ()
[inline]
- 6.396.2.2 **virtual activemq::wireformat::openwire::marshal::generated::PartialCommandMarshaller::~PartialCommandMarshaller** () [inline, virtual]

6.396.3 Member Function Documentation

6.396 activemq::wireformat::openwire::marshal::generated::PartialCommandMarshaller Class

Reference

2087

```
6.396.3.1 virtual commands::DataStructure* activemq::wireformat::openwire-  
::marshal::generated::PartialCommandMarshaller::createObject ( )  
const [virtual]
```

Creates a new instance of the class that this class is a marshaling director for.

Returns

newly allocated Command

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1120).

Reimplemented in **activemq::wireformat::openwire::marshal::generated::LastPartialCommandMarshaller** (p. 1609).

```
6.396.3.2 virtual unsigned char activemq::wireformat::openwire::marshal::generated-  
::PartialCommandMarshaller::getDataStructureType ( ) const  
[virtual]
```

Gets the DataStructureType that this class marshals/unmarshals.

Returns

byte Id of this classes DataStructureType

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1122).

Reimplemented in **activemq::wireformat::openwire::marshal::generated::LastPartialCommandMarshaller** (p. 1610).

```
6.396.3.3 virtual void activemq::wireformat::openwire::marshal::generated::-  
PartialCommandMarshaller::looseMarshal ( OpenWireFormat * format,  
commands::DataStructure * command, decaf::io::DataOutputStream * ds  
) [virtual]
```

Tight Marshal to the given stream.

Parameters

<i>format</i>	- The OpenWireFormat properties
<i>command</i>	- the object to Marshal
<i>ds</i>	- DataOutputStream to marshal to

Exceptions

<i>IOException</i>	if an error occurs.
--------------------	---------------------

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1123).

Reimplemented in **activemq::wireformat::openwire::marshal::generated::Last-PartialCommandMarshaller** (p. 1610).

6.396.3.4 `virtual void activemq::wireformat::openwire::marshal::generated::Partial-CommandMarshaller::looseUnmarshal (OpenWireFormat * format, commands::DataStructure * command, decaf::io::DataInputStream * dis)`
`[virtual]`

Loose Un-marhsal to the given stream.

Parameters

<i>format</i>	- The OpenWireFormat properties
<i>command</i>	- the object to Un-Marshal
<i>dis</i>	- the DataInputStream to Un-Marshal from

Exceptions

<i>IOException</i>	if an error occurs.
--------------------	---------------------

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1125).

Reimplemented in **activemq::wireformat::openwire::marshal::generated::Last-PartialCommandMarshaller** (p. 1610).

6.396.3.5 `virtual int activemq::wireformat::openwire::marshal::generated::Partial-CommandMarshaller::tightMarshal1 (OpenWireFormat * format, commands::DataStructure * command, utils::BooleanStream * bs)`
`[virtual]`

Tight Marhsal to the given stream.

Parameters

<i>format</i>	The OpenWireFormat properties
<i>command</i>	The object to Marshal
<i>bs</i>	The boolean stream to marshal to.

Exceptions

<i>IOException</i>	if an error occurs.
--------------------	---------------------

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1127).

6.396 activemq::wireformat::openwire::marshal::generated::PartialCommandMarshaller Class

Reference

2089

Reimplemented in **activemq::wireformat::openwire::marshal::generated::LastPartialCommandMarshaller** (p. 1611).

6.396.3.6 virtual void **activemq::wireformat::openwire::marshal::generated::PartialCommandMarshaller::tightMarshal2** (**OpenWireFormat** * *format*, **commands::DataStructure** * *command*, **decaf::io::DataOutputStream** * *ds*, **utils::BooleanStream** * *bs*) [virtual]

Tight Marshal to the given stream.

Parameters

<i>format</i>	- The OpenWireFormat properties
<i>command</i>	- the object to Marshal
<i>ds</i>	- the DataOutputStream to Marshal to
<i>bs</i>	- boolean stream to marshal to.

Exceptions

<i>IOException</i>	if an error occurs.
--------------------	---------------------

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1129).

Reimplemented in **activemq::wireformat::openwire::marshal::generated::LastPartialCommandMarshaller** (p. 1611).

6.396.3.7 virtual void **activemq::wireformat::openwire::marshal::generated::PartialCommandMarshaller::tightUnmarshal** (**OpenWireFormat** * *format*, **commands::DataStructure** * *command*, **decaf::io::DataInputStream** * *dis*, **utils::BooleanStream** * *bs*) [virtual]

Tight Un-marshal to the given stream.

Parameters

<i>format</i>	- The OpenWireFormat properties
<i>command</i>	- the object to Un-Marshal
<i>dis</i>	- the DataInputStream to Un-Marshal from
<i>bs</i>	- boolean stream to unmarshal from.

Exceptions

<i>IOException</i>	if an error occurs.
--------------------	---------------------

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1131).

Reimplemented in **activemq::wireformat::openwire::marshal::generated::Last-PartialCommandMarshaller** (p. 1612).

The documentation for this class was generated from the following file:

- `src/main/activemq/wireformat/openwire/marshal/generated/PartialCommand-Marshaller.h`

6.397 **decaf::lang::Pointer**< T, REFCOUNTER > Class Template - Reference

Decaf's implementation of a Smart **Pointer** (p. 2083) that is a template on a Type and is **Thread** (p. 2703) Safe if the default Reference Counter is used.

```
#include <src/main/decaf/lang/Pointer.h>
```

Public Types

- `typedef T * PointerType`
- `typedef T & ReferenceType`
- `typedef REFCOUNTER CounterType`

Public Member Functions

- **Pointer** ()
Default Constructor.
- **Pointer** (const **PointerType** value)
*Explicit Constructor, creates a **Pointer** (p. 2083) that contains value with a single reference.*
- **Pointer** (const **Pointer** &value) throw ()
Copy constructor.
- `template<typename T1 , typename R1 >`
Pointer (const **Pointer**< T1, R1 > &value) throw ()
Copy constructor.
- `template<typename T1 , typename R1 >`
Pointer (const **Pointer**< T1, R1 > &value, const **STATIC_CAST_TOKEN** &) throw ()
Static Cast constructor.
- `template<typename T1 , typename R1 >`
Pointer (const **Pointer**< T1, R1 > &value, const **DYNAMIC_CAST_TOKEN** &)
Dynamic Cast constructor.
- `virtual ~Pointer () throw ()`
- `void reset (T *value)`
*Resets the **Pointer** (p. 2083) to hold the new value.*

6.397 decaf::lang::Pointer< T, REFCOUNTER > Class Template Reference 2091

- **T * release ()**
*Releases the **Pointer** (p. 2083) held and resets the internal pointer value to Null.*
- **PointerType get () const**
*Gets the real pointer that is contained within this **Pointer** (p. 2083).*
- **void swap (Pointer &value) throw ()**
***Exception** (p. 1279) Safe Swap Function.*
- **Pointer & operator= (const Pointer &right) throw ()**
*Assigns the value of right to this **Pointer** (p. 2083) and increments the reference Count.*
- **template<typename T1 , typename R1 >**
Pointer & operator= (const Pointer< T1, R1 > &right) throw ()
- **ReferenceType operator* ()**
Dereference Operator, returns a reference to the Contained value.
- **ReferenceType operator* () const**
- **PointerType operator-> ()**
Indirection Operator, returns a pointer to the Contained value.
- **PointerType operator-> () const**
- **bool operator! () const**
- **template<typename T1 , typename R1 >**
bool operator== (const Pointer< T1, R1 > &right) const
- **template<typename T1 , typename R1 >**
bool operator!= (const Pointer< T1, R1 > &right) const
- **template<typename T1 >**
Pointer< T1, CounterType > dynamicCast () const
- **template<typename T1 >**
Pointer< T1, CounterType > staticCast () const

Friends

- **bool operator== (const Pointer &left, const T *right)**
- **bool operator== (const T *left, const Pointer &right)**
- **bool operator!= (const Pointer &left, const T *right)**
- **bool operator!= (const T *left, const Pointer &right)**

6.397.1 Detailed Description

`template<typename T, typename REFCOUNTER = decaf::util::concurrent::atomic::AtomicRefCount> class decaf::lang::Pointer< T, REFCOUNTER >`

Decaf's implementation of a Smart **Pointer** (p. 2083) that is a template on a Type and is **Thread** (p. 2703) Safe if the default Reference Counter is used.

This **Pointer** (p. 2083) type allows for the substitution of different Reference Counter implementations which provide a means of using invasive reference counting if desired using a custom implementation of `ReferenceCounter`.

The Decaf smart pointer provide comparison operators for comparing **Pointer** (p. 2083) instances in the same manner as normal pointer, except that it does not provide an

overload of operators (<, <=, >, >=). To allow use of a **Pointer** (p. 2083) in a STL container that requires it, **Pointer** (p. 2083) provides an implementation of std::less.

Since

1.0

6.397.2 Member Typedef Documentation

6.397.2.1 `template<typename T, typename REFCOUNTER = decaf::util::concurrent::atomic::AtomicRefCounter> typedef REFCOUNTER decaf::lang::Pointer< T, REFCOUNTER >::CounterType`

6.397.2.2 `template<typename T, typename REFCOUNTER = decaf::util::concurrent::atomic::AtomicRefCounter> typedef T* decaf::lang::Pointer< T, REFCOUNTER >::PointerType`

6.397.2.3 `template<typename T, typename REFCOUNTER = decaf::util::concurrent::atomic::AtomicRefCounter> typedef T& decaf::lang::Pointer< T, REFCOUNTER >::ReferenceType`

6.397.3 Constructor & Destructor Documentation

6.397.3.1 `template<typename T, typename REFCOUNTER = decaf::util::concurrent::atomic::AtomicRefCounter> decaf::lang::Pointer< T, REFCOUNTER >::Pointer () [inline]`

Default Constructor.

Initialized the contained pointer to NULL, using the -> operator results in an exception unless reset to contain a real value.

Referenced by decaf::lang::Pointer< TransactionId >::reset().

6.397.3.2 `template<typename T, typename REFCOUNTER = decaf::util::concurrent::atomic::AtomicRefCounter> decaf::lang::Pointer< T, REFCOUNTER >::Pointer (const PointerType value) [inline, explicit]`

Explicit Constructor, creates a **Pointer** (p. 2083) that contains value with a single reference.

This object now has ownership until a call to release.

Parameters

<i>value</i>	- instance of the type we are containing here.
--------------	--

6.397 decaf::lang::Pointer< T, REFCOUNTER > Class Template Reference 2093

6.397.3.3 `template<typename T, typename REFCOUNTER = decaf::util::concurrent-
::AtomicRefCount> decaf::lang::Pointer< T, REFCOUNTER >::Pointer (const
Pointer< T, REFCOUNTER > & value) throw () [inline]`

Copy constructor.

Copies the value contained in the pointer to the new instance and increments the reference counter.

6.397.3.4 `template<typename T, typename REFCOUNTER = decaf::util::concurrent-
::atomic::AtomicRefCount> template<typename T1 , typename R1 >
decaf::lang::Pointer< T, REFCOUNTER >::Pointer (const Pointer< T1, R1 >
& value) throw () [inline]`

Copy constructor.

Copies the value contained in the pointer to the new instance and increments the reference counter.

6.397.3.5 `template<typename T, typename REFCOUNTER = decaf::util::concurrent-
::atomic::AtomicRefCount> template<typename T1 , typename R1 >
decaf::lang::Pointer< T, REFCOUNTER >::Pointer (const Pointer< T1, R1 >
& value, const STATIC_CAST_TOKEN &) throw () [inline]`

Static Cast constructor.

Copies the value contained in the pointer to the new instance and increments the reference counter performing a static cast on the value contained in the source **Pointer** (p.2083) object.

Parameters

<i>value</i>	- Pointer (p.2083) instance to cast to this type.
--------------	--

6.397.3.6 `template<typename T, typename REFCOUNTER = decaf::util::concurrent-
::atomic::AtomicRefCount> template<typename T1 , typename R1 >
decaf::lang::Pointer< T, REFCOUNTER >::Pointer (const Pointer< T1, R1 >
& value, const DYNAMIC_CAST_TOKEN &) [inline]`

Dynamic Cast constructor.

Copies the value contained in the pointer to the new instance and increments the reference counter performing a dynamic cast on the value contained in the source **Pointer** (p.2083) object. If the cast fails and return NULL then this method throws a ClassCastException.

Parameters

<i>value</i>	- Pointer (p.2083) instance to cast to this type.
--------------	--

Exceptions

<i>ClassCastException</i>	if the dynamic cast returns NULL
---------------------------	----------------------------------

```
6.397.3.7  template<typename T, typename REFCOUNTER = decaf::util::concurrent::atomic-
::AtomicRefCount> virtual decaf::lang::Pointer< T, REFCOUNTER
>::~~Pointer ( ) throw () [inline, virtual]
```

6.397.4 Member Function Documentation

```
6.397.4.1  template<typename T, typename REFCOUNTER = decaf::util::concurrent::atomic-
AtomicRefCount> template<typename T1 > Pointer<T1, CounterType>
decaf::lang::Pointer< T, REFCOUNTER >::dynamicCast ( ) const
[inline]
```

```
6.397.4.2  template<typename T, typename REFCOUNTER = decaf::util::concurrent::atomic-
AtomicRefCount> PointerType decaf::lang::Pointer< T, REFCOUNTER
>::get ( ) const [inline]
```

Gets the real pointer that is contained within this **Pointer** (p. 2083).

This is not really safe since the caller could delete or alter the pointer but it mimics the STL `auto_ptr` and gives access in cases where the caller absolutely needs the real **Pointer** (p. 2083). Use at your own risk.

Returns

the contained pointer.

Referenced by `activemq::commands::ActiveMQMessageTemplate< cms::Object-Message >::equals()`, `activemq::state::ConnectionState::getTransactionState()`, `decaf::lang::Pointer< TransactionId >::operator!=()`, `decaf::lang::operator!=()`, `std::less< decaf::lang::Pointer< T > >::operator()`, `decaf::lang::Pointer< TransactionId >::operator==()`, `decaf::lang::operator==()`, `decaf::util::concurrent::LinkedBlockingQueue< E >::peek()`, and `activemq::state::ConnectionState::removeTempDestination()`.

```
6.397.4.3  template<typename T, typename REFCOUNTER = decaf::util::concurrent::atomic-
AtomicRefCount> bool decaf::lang::Pointer< T, REFCOUNTER >::operator! (
) const [inline]
```

```
6.397.4.4  template<typename T, typename REFCOUNTER = decaf::util::concurrent-
::atomic::AtomicRefCount> template<typename T1 , typename R1 > bool
decaf::lang::Pointer< T, REFCOUNTER >::operator!= ( const Pointer< T1, R1
> & right ) const [inline]
```

6.397 decaf::lang::Pointer< T, REFCOUNTER > Class Template Reference 2095

6.397.4.5 `template<typename T, typename REFCOUNTER = decaf::util::concurrent::atomic::AtomicRefCounter> ReferenceType decaf::lang::Pointer< T, REFCOUNTER >::operator*() [inline]`

Dereference Operator, returns a reference to the Contained value.

This method throws an `NullPointerException` if the contained value is `NULL`.

Returns

reference to the contained pointer.

Exceptions

<code>NullPointerException</code>	if the contained value is <code>Null</code>
-----------------------------------	---

6.397.4.6 `template<typename T, typename REFCOUNTER = decaf::util::concurrent::atomic::AtomicRefCounter> ReferenceType decaf::lang::Pointer< T, REFCOUNTER >::operator*() const [inline]`

6.397.4.7 `template<typename T, typename REFCOUNTER = decaf::util::concurrent::atomic::AtomicRefCounter> PointerType decaf::lang::Pointer< T, REFCOUNTER >::operator-> () [inline]`

Indirection Operator, returns a pointer to the Contained value.

This method throws an `NullPointerException` if the contained value is `NULL`.

Returns

reference to the contained pointer.

Exceptions

<code>NullPointerException</code>	if the contained value is <code>Null</code>
-----------------------------------	---

6.397.4.8 `template<typename T, typename REFCOUNTER = decaf::util::concurrent::atomic::AtomicRefCounter> PointerType decaf::lang::Pointer< T, REFCOUNTER >::operator-> () const [inline]`

6.397.4.9 `template<typename T, typename REFCOUNTER = decaf::util::concurrent::atomic::AtomicRefCounter> Pointer& decaf::lang::Pointer< T, REFCOUNTER >::operator=(const Pointer< T, REFCOUNTER > & right) throw () [inline]`

Assigns the value of `right` to this **Pointer** (p. 2083) and increments the reference Count.

Parameters

<i>right</i>	- Pointer (p. 2083) on the right hand side of an operator= call to this.
--------------	---

6.397.4.10 `template<typename T, typename REFCOUNTER = decaf::util::concurrent::atomic::AtomicRefCounter> template<typename T1, typename R1 > Pointer& decaf::lang::Pointer< T, REFCOUNTER >::operator= (const Pointer< T1, R1 > & right) throw () [inline]`

6.397.4.11 `template<typename T, typename REFCOUNTER = decaf::util::concurrent::atomic::AtomicRefCounter> template<typename T1, typename R1 > bool decaf::lang::Pointer< T, REFCOUNTER >::operator== (const Pointer< T1, R1 > & right) const [inline]`

6.397.4.12 `template<typename T, typename REFCOUNTER = decaf::util::concurrent::atomic::AtomicRefCounter> T* decaf::lang::Pointer< T, REFCOUNTER >::release () [inline]`

Releases the **Pointer** (p. 2083) held and resets the internal pointer value to Null.

This method is not guaranteed to be safe if the **Pointer** (p. 2083) is held by more than one object or this method is called from more than one thread.

Parameters

<i>value</i>	- The new value to contain.
--------------	-----------------------------

Returns

The pointer instance that was held by this **Pointer** (p. 2083) object, the pointer is no longer owned by this **Pointer** (p. 2083) and won't be freed when this **Pointer** (p. 2083) goes out of scope.

Referenced by `decaf::lang::Pointer< TransactionId >::Pointer()`.

6.397.4.13 `template<typename T, typename REFCOUNTER = decaf::util::concurrent::atomic::AtomicRefCounter> void decaf::lang::Pointer< T, REFCOUNTER >::reset (T * value) [inline]`

Resets the **Pointer** (p. 2083) to hold the new value.

Before the new value is stored reset checks if the old value should be destroyed and if so calls delete. Call reset with a value of NULL is supported and acts to set this **Pointer** (p. 2083) to a NULL pointer.

Parameters

<i>value</i>	- The new value to contain.
--------------	-----------------------------

6.398 decaf::lang::PointerComparator< T, R > Class Template Reference 2097

- 6.397.4.14 `template<typename T, typename REFCOUNTER = decaf::util::concurrent::atomic::AtomicRefCounter> template<typename T1 > Pointer<T1, CounterType> decaf::lang::Pointer< T, REFCOUNTER >::staticCast () const`
[inline]
- 6.397.4.15 `template<typename T, typename REFCOUNTER = decaf::util::concurrent::atomic::AtomicRefCounter> void decaf::lang::Pointer< T, REFCOUNTER >::swap (Pointer< T, REFCOUNTER > & value) throw ()` [inline]

Exception (p. 1279) Safe Swap Function.

Parameters

<i>value</i>	- the value to swap with this.
--------------	--------------------------------

Referenced by `decaf::lang::Pointer< TransactionId >::operator=()`, and `decaf::lang::Pointer< TransactionId >::swap()`.

6.397.5 Friends And Related Function Documentation

- 6.397.5.1 `template<typename T, typename REFCOUNTER = decaf::util::concurrent::atomic::AtomicRefCounter> bool operator!= (const Pointer< T, REFCOUNTER > & left, const T * right)` [friend]
- 6.397.5.2 `template<typename T, typename REFCOUNTER = decaf::util::concurrent::atomic::AtomicRefCounter> bool operator!= (const T * left, const Pointer< T, REFCOUNTER > & right)` [friend]
- 6.397.5.3 `template<typename T, typename REFCOUNTER = decaf::util::concurrent::atomic::AtomicRefCounter> bool operator== (const Pointer< T, REFCOUNTER > & left, const T * right)` [friend]
- 6.397.5.4 `template<typename T, typename REFCOUNTER = decaf::util::concurrent::atomic::AtomicRefCounter> bool operator== (const T * left, const Pointer< T, REFCOUNTER > & right)` [friend]

The documentation for this class was generated from the following file:

- `src/main/decaf/lang/Pointer.h`

6.398 decaf::lang::PointerComparator< T, R > Class Template - Reference

This implementation of Comparator is designed to allows objects in a Collection to be sorted or tested for equality based on the value of the Object being Pointed to and not the value of the contained pointer in the **Pointer** (p. 2083) instance.

```
#include <src/main/decaf/lang/Pointer.h>
```

Inheritance diagram for decaf::lang::PointerComparator< T, R >:

Public Member Functions

- virtual **~PointerComparator** ()
- virtual bool **operator()** (const **Pointer**< T, R > &left, const **Pointer**< T, R > &right) const
- virtual int **compare** (const **Pointer**< T, R > &left, const **Pointer**< T, R > &right) const

6.398.1 Detailed Description

```
template<typename T, typename R = decaf::util::concurrent::atomic::AtomicRefCounter>class
decaf::lang::PointerComparator< T, R >
```

This implementation of Comparator is designed to allows objects in a Collection to be sorted or tested for equality based on the value of the Object being Pointed to and not the value of the contained pointer in the **Pointer** (p. 2083) instance.

This can be useful in the case where a series of values in a Collection is more efficiently accessed in the Objects Natural Order and not the underlying pointers location in memory.

Also this allows **Pointer** (p. 2083) objects that Point to different instances of the same type to be compared based on the comparison of the object itself and not just the value of the pointer.

6.398.2 Constructor & Destructor Documentation

```
6.398.2.1 template<typename T , typename R = decaf::util::concurrent::atomic::-
AtomicRefCounter> virtual decaf::lang::PointerComparator< T, R
>::~~PointerComparator( ) [inline, virtual]
```

6.398.3 Member Function Documentation

```
6.398.3.1 template<typename T , typename R = decaf::util::concurrent::atomic::AtomicRef-
Counter> virtual int decaf::lang::PointerComparator< T, R >::compare
( const Pointer< T, R > & left, const Pointer< T, R > & right ) const
[inline, virtual]
```



```
6.398.3.2  template<typename T , typename R = decaf::util::concurrent::atomic::AtomicRef-
Counter> virtual bool decaf::lang::PointerComparator< T, R >::operator()
( const Pointer< T, R > & left, const Pointer< T, R > & right ) const
[inline, virtual]
```

The documentation for this class was generated from the following file:

- src/main/decaf/lang/Pointer.h

6.399 activemq::cmsutil::PooledSession Class Reference

A pooled session object that wraps around a delegate session.

```
#include <src/main/activemq/cmsutil/PooledSession.h>
```

Inheritance diagram for activemq::cmsutil::PooledSession:

Public Member Functions

- **PooledSession** (**SessionPool** *pool, **cms::Session** *session)
- virtual **~PooledSession** () throw ()
Does nothing.
- virtual **cms::Session** * **getSession** ()
Returns a non-constant reference to the internal session object.
- virtual const **cms::Session** * **getSession** () const
Returns a constant reference to the internal session object.
- virtual void **close** ()
Returns this session back to the pool, but does not close or destroy the internal session object.
- virtual void **start** ()
Starts the service.
- virtual void **stop** ()
Stops this service.
- virtual void **commit** ()
Commits all messages done in this transaction and releases any locks currently held.
- virtual void **rollback** ()
Rolls back all messages done in this transaction and releases any locks currently held.
- virtual void **recover** ()
Stops message delivery in this session, and restarts message delivery with the oldest unacknowledged message.
- virtual **cms::MessageConsumer** * **createConsumer** (const **cms::Destination** *destination)
Creates a MessageConsumer for the specified destination.

- virtual **cms::MessageConsumer** * **createConsumer** (const **cms::Destination** *destination, const std::string &selector)
Creates a MessageConsumer for the specified destination, using a message selector.
- virtual **cms::MessageConsumer** * **createConsumer** (const **cms::Destination** *destination, const std::string &selector, bool noLocal)
Creates a MessageConsumer for the specified destination, using a message selector.
- virtual **cms::MessageConsumer** * **createDurableConsumer** (const **cms::Topic** *destination, const std::string &name, const std::string &selector, bool noLocal=false)
Creates a durable subscriber to the specified topic, using a Message selector.
- virtual **cms::MessageConsumer** * **createCachedConsumer** (const **cms::Destination** *destination, const std::string &selector, bool noLocal)
First checks the internal consumer cache and creates one if none exist for the given destination, selector, noLocal.
- virtual **cms::MessageProducer** * **createProducer** (const **cms::Destination** *destination)
Creates a MessageProducer to send messages to the specified destination.
- virtual **cms::MessageProducer** * **createCachedProducer** (const **cms::Destination** *destination)
First checks the internal producer cache and creates one if none exist for the given destination.
- virtual **cms::QueueBrowser** * **createBrowser** (const **cms::Queue** *queue)
Creates a new QueueBrowser to peek at Messages on the given Queue.
- virtual **cms::QueueBrowser** * **createBrowser** (const **cms::Queue** *queue, const std::string &selector)
Creates a new QueueBrowser to peek at Messages on the given Queue.
- virtual **cms::Queue** * **createQueue** (const std::string &queueName)
Creates a queue identity given a Queue name.
- virtual **cms::Topic** * **createTopic** (const std::string &topicName)
Creates a topic identity given a Queue name.
- virtual **cms::TemporaryQueue** * **createTemporaryQueue** ()
Creates a TemporaryQueue object.
- virtual **cms::TemporaryTopic** * **createTemporaryTopic** ()
Creates a TemporaryTopic object.
- virtual **cms::Message** * **createMessage** ()
Creates a new Message.
- virtual **cms::BytesMessage** * **createBytesMessage** ()
Creates a BytesMessage.
- virtual **cms::BytesMessage** * **createBytesMessage** (const unsigned char *bytes, int bytesSize)
Creates a BytesMessage and sets the payload to the passed value.
- virtual **cms::StreamMessage** * **createStreamMessage** ()
Creates a new StreamMessage.
- virtual **cms::TextMessage** * **createTextMessage** ()
Creates a new TextMessage.

- virtual **cms::TextMessage** * **createTextMessage** (const std::string &text)
Creates a new TextMessage and set the text to the value given.
- virtual **cms::MapMessage** * **createMapMessage** ()
Creates a new MapMessage.
- virtual **cms::Session::AcknowledgeMode** **getAcknowledgeMode** () const
Returns the acknowledgment mode of the session.
- virtual bool **isTransacted** () const
Gets if the Sessions is a Transacted Session.
- virtual void **unsubscribe** (const std::string &name)
Unsubscribes a durable subscription that has been created by a client.

6.399.1 Detailed Description

A pooled session object that wraps around a delegate session.

Calls to close this session only result in giving the session back to the pool.

6.399.2 Constructor & Destructor Documentation

6.399.2.1 **activemq::cmsutil::PooledSession::PooledSession** (**SessionPool** * *pool*,
cms::Session * *session*)

6.399.2.2 **virtual activemq::cmsutil::PooledSession::~~PooledSession** () throw ()
[virtual]

Does nothing.

6.399.3 Member Function Documentation

6.399.3.1 **virtual void activemq::cmsutil::PooledSession::close** () [virtual]

Returns this session back to the pool, but does not close or destroy the internal session object.

Exceptions

<i>CMSException</i>	if an error occurs while performing this operation.
---------------------	---

Implements **cms::Session** (p. 2365).

6.399.3.2 **virtual void activemq::cmsutil::PooledSession::commit** () [inline,
virtual]

Commits all messages done in this transaction and releases any locks currently held.

Exceptions

<i>CMSEException</i>	- If an internal error occurs.
<i>IllegalStateException</i>	- if the method is not called by a transacted session.

Implements **cms::Session** (p. 2366).

References cms::Session::commit().

6.399.3.3 virtual cms::QueueBrowser* activemq::cmsutil::PooledSession::createBrowser (const cms::Queue * *queue*)
[virtual]

Creates a new QueueBrowser to peek at Messages on the given Queue.

Parameters

<i>queue</i>	the Queue to browse
--------------	---------------------

Returns

New QueueBrowser that is owned by the caller.

Exceptions

<i>CMSEException</i>	- If an internal error occurs.
<i>InvalidDestinationException</i>	- if the destination given is invalid.

Implements **cms::Session** (p. 2366).

6.399.3.4 virtual cms::QueueBrowser* activemq::cmsutil::PooledSession::createBrowser (const cms::Queue * *queue*, const std::string & *selector*)
[virtual]

Creates a new QueueBrowser to peek at Messages on the given Queue.

Parameters

<i>queue</i>	the Queue to browse
<i>selector</i>	the Message selector to filter which messages are browsed.

Returns

New QueueBrowser that is owned by the caller.

Exceptions

<i>CMSEException</i>	- If an internal error occurs.
<i>InvalidDestinationException</i>	- if the destination given is invalid.

Implements **cms::Session** (p. 2367).

6.399.3.5 **virtual cms::BytesMessage* activemq::cmsutil::PooledSession::createBytesMessage ()** [*inline*, *virtual*]

Creates a BytesMessage.

Exceptions

<i>CMSEException</i>	- If an internal error occurs.
----------------------	--------------------------------

Implements **cms::Session** (p. 2367).

References cms::Session::createBytesMessage().

6.399.3.6 **virtual cms::BytesMessage* activemq::cmsutil::PooledSession::createBytesMessage (const unsigned char * bytes, int bytesSize)** [*inline*, *virtual*]

Creates a BytesMessage and sets the payload to the passed value.

Parameters

<i>bytes</i>	an array of bytes to set in the message
<i>bytesSize</i>	the size of the bytes array, or number of bytes to use

Exceptions

<i>CMSEException</i>	- If an internal error occurs.
----------------------	--------------------------------

Implements **cms::Session** (p. 2367).

References cms::Session::createBytesMessage().

6.399.3.7 **virtual cms::MessageConsumer* activemq::cmsutil::PooledSession::createCachedConsumer (const cms::Destination * destination, const std::string & selector, bool noLocal)** [*virtual*]

First checks the internal consumer cache and creates one if none exist for the given destination, selector, noLocal.

If created, the consumer is added to the pool's lifecycle manager.

Parameters

<i>destination</i>	the destination to receive on
<i>selector</i>	the selector to use
<i>noLocal</i>	whether or not to receive messages from the same connection

Returns

the consumer resource

Exceptions

<i>cms::CMS-Exception</i> (p. 826)	if something goes wrong.
--	--------------------------

6.399.3.8 `virtual cms::MessageProducer* activemq::cmsutil::PooledSession-
::createCachedProducer (const cms::Destination * destination)
[virtual]`

First checks the internal producer cache and creates one if none exist for the given destination.

If created, the producer is added to the pool's lifecycle manager.

Parameters

<i>destination</i>	the destination to send on
--------------------	----------------------------

Returns

the producer resource

Exceptions

<i>cms::CMS-Exception</i> (p. 826)	if something goes wrong.
--	--------------------------

6.399.3.9 `virtual cms::MessageConsumer* activemq::cmsutil::PooledSession-
::createConsumer (const cms::Destination * destination) [inline,
virtual]`

Creates a MessageConsumer for the specified destination.

Parameters

<i>destination</i>	the Destination that this consumer receiving messages for.
--------------------	--

Returns

pointer to a new MessageConsumer that is owned by the caller (caller deletes)

Exceptions

<i>CMSException</i>	- If an internal error occurs.
<i>InvalidDestination-Exception</i>	- if an invalid destination is specified.

Implements **cms::Session** (p. 2368).

References cms::Session::createConsumer().

6.399.3.10 virtual **cms::MessageConsumer*** **activemq::cmsutil::PooledSession-**
::createConsumer (const cms::Destination * destination, const std::string &
selector) [inline, virtual]

Creates a MessageConsumer for the specified destination, using a message selector.

Parameters

<i>destination</i>	the Destination that this consumer receiving messages for.
<i>selector</i>	the Message Selector to use

Returns

pointer to a new MessageConsumer that is owned by the caller (caller deletes)

Exceptions

<i>CMSException</i>	- If an internal error occurs.
<i>InvalidDestination-Exception</i>	- if an invalid destination is specified.
<i>InvalidSelector-Exception</i>	- if the message selector is invalid.

Implements **cms::Session** (p. 2368).

References cms::Session::createConsumer().

6.399.3.11 `virtual cms::MessageConsumer* activemq::cmsutil::PooledSession-
::createConsumer (const cms::Destination * destination, const std::string &
selector, bool noLocal) [inline, virtual]`

Creates a MessageConsumer for the specified destination, using a message selector.

Parameters

<i>destination</i>	the Destination that this consumer receiving messages for.
<i>selector</i>	the Message Selector to use
<i>noLocal</i>	if true, and the destination is a topic, inhibits the delivery of messages published by its own connection. The behavior for NoLocal is not specified if the destination is a queue.

Returns

pointer to a new MessageConsumer that is owned by the caller (caller deletes)

Exceptions

<i>CMSException</i>	- If an internal error occurs.
<i>InvalidDestination-Exception</i>	- if an invalid destination is specified.
<i>InvalidSelector-Exception</i>	- if the message selector is invalid.

Implements **cms::Session** (p. 2369).

References cms::Session::createConsumer().

6.399.3.12 `virtual cms::MessageConsumer* activemq::cmsutil::PooledSession-
::createDurableConsumer (const cms::Topic * destination, const std::string
& name, const std::string & selector, bool noLocal = false) [inline,
virtual]`

Creates a durable subscriber to the specified topic, using a Message selector.

Sessions that create durable consumers must use the same client Id as was used the last time the subscription was created in order to receive all messages that were delivered while the client was offline.

Parameters

<i>destination</i>	the topic to subscribe to
<i>name</i>	The name used to identify the subscription
<i>selector</i>	the Message Selector to use
<i>noLocal</i>	if true, and the destination is a topic, inhibits the delivery of messages published by its own connection. The behavior for NoLocal is not specified if the destination is a queue.

Returns

pointer to a new durable MessageConsumer that is owned by the caller (caller deletes)

Exceptions

<i>CMSEException</i>	- If an internal error occurs.
<i>InvalidDestination-Exception</i>	- if an invalid destination is specified.
<i>InvalidSelector-Exception</i>	- if the message selector is invalid.

Implements **cms::Session** (p. 2370).

References cms::Session::createDurableConsumer().

6.399.3.13 virtual **cms::MapMessage*** **activemq::cmsutil::PooledSession::createMapMessage ()** [*inline*, *virtual*]

Creates a new MapMessage.

Exceptions

<i>CMSEException</i>	- If an internal error occurs.
----------------------	--------------------------------

Implements **cms::Session** (p. 2371).

References cms::Session::createMapMessage().

6.399.3.14 virtual **cms::Message*** **activemq::cmsutil::PooledSession::createMessage ()** [*inline*, *virtual*]

Creates a new Message.

Exceptions

<i>CMSEException</i>	- If an internal error occurs.
----------------------	--------------------------------

Implements **cms::Session** (p. 2371).

References cms::Session::createMessage().

6.399.3.15 virtual **cms::MessageProducer*** **activemq::cmsutil::PooledSession::createProducer (const cms::Destination * destination)** [*inline*, *virtual*]

Creates a MessageProducer to send messages to the specified destination.

Parameters

<i>destination</i>	the Destination to send on
--------------------	----------------------------

Returns

New MessageProducer that is owned by the caller.

Exceptions

<i>CMSEException</i>	- If an internal error occurs.
<i>InvalidDestination-Exception</i>	- if an invalid destination is specified.

Implements **cms::Session** (p. 2371).

References cms::Session::createProducer().

6.399.3.16 **virtual cms::Queue* activemq::cmsutil::PooledSession::createQueue (**
const std::string & queueName) [inline, virtual]

Creates a queue identity given a Queue name.

Parameters

<i>queueName</i>	the name of the new Queue
------------------	---------------------------

Returns

new Queue pointer that is owned by the caller.

Exceptions

<i>CMSEException</i>	- If an internal error occurs.
----------------------	--------------------------------

Implements **cms::Session** (p. 2372).

References cms::Session::createQueue().

6.399.3.17 **virtual cms::StreamMessage* activemq::cmsutil::-**
PooledSession::createStreamMessage () [inline,
 virtual]

Creates a new StreamMessage.

Exceptions

<i>CMSEException</i>	- If an internal error occurs.
----------------------	--------------------------------

Implements **cms::Session** (p. 2372).

References cms::Session::createStreamMessage().

6.399.3.18 **virtual cms::TemporaryQueue* activemq::cmsutil::PooledSession::createTemporaryQueue ()** [inline, virtual]

Creates a TemporaryQueue object.

Returns

new TemporaryQueue pointer that is owned by the caller.

Exceptions

<i>CMSEException</i>	- If an internal error occurs.
----------------------	--------------------------------

Implements **cms::Session** (p. 2373).

References cms::Session::createTemporaryQueue().

6.399.3.19 **virtual cms::TemporaryTopic* activemq::cmsutil::PooledSession::createTemporaryTopic ()** [inline, virtual]

Creates a TemporaryTopic object.

Exceptions

<i>CMSEException</i>	- If an internal error occurs.
----------------------	--------------------------------

Implements **cms::Session** (p. 2373).

References cms::Session::createTemporaryTopic().

6.399.3.20 **virtual cms::TextMessage* activemq::cmsutil::PooledSession::createTextMessage ()** [inline, virtual]

Creates a new TextMessage.

Exceptions

<i>CMSEException</i>	- If an internal error occurs.
----------------------	--------------------------------

Implements **cms::Session** (p. 2373).

References cms::Session::createTextMessage().

6.399.3.21 `virtual cms::TextMessage* activemq::cmsutil::PooledSession-
::createTextMessage (const std::string & text) [inline,
virtual]`

Creates a new TextMessage and set the text to the value given.

Parameters

<i>text</i>	the initial text for the message
-------------	----------------------------------

Exceptions

<i>CMSEException</i>	- If an internal error occurs.
----------------------	--------------------------------

Implements **cms::Session** (p. 2374).

References cms::Session::createTextMessage().

6.399.3.22 `virtual cms::Topic* activemq::cmsutil::PooledSession::createTopic (
const std::string & topicName) [inline, virtual]`

Creates a topic identity given a Queue name.

Parameters

<i>topicName</i>	the name of the new Topic
------------------	---------------------------

Returns

new Topic pointer that is owned by the caller.

Exceptions

<i>CMSEException</i>	- If an internal error occurs.
----------------------	--------------------------------

Implements **cms::Session** (p. 2374).

References cms::Session::createTopic().

6.399.3.23 `virtual cms::Session::AcknowledgeMode activemq::cmsutil:-
PooledSession::getAcknowledgeMode () const [inline,
virtual]`

Returns the acknowledgment mode of the session.

Returns

the Sessions Acknowledge Mode

Exceptions

<i>CMSEException</i>	- If an internal error occurs.
----------------------	--------------------------------

Implements **cms::Session** (p. 2374).

References cms::Session::getAcknowledgeMode().

6.399.3.24 **virtual cms::Session* activemq::cmsutil::PooledSession::getSession ()** [inline, virtual]

Returns a non-constant reference to the internal session object.

Returns

the session object.

6.399.3.25 **virtual const cms::Session* activemq::cmsutil::PooledSession::getSession () const** [inline, virtual]

Returns a constant reference to the internal session object.

Returns

the session object.

6.399.3.26 **virtual bool activemq::cmsutil::PooledSession::isTransacted () const** [inline, virtual]

Gets if the Sessions is a Transacted Session.

Returns

transacted true - false.

Exceptions

<i>CMSEException</i>	- If an internal error occurs.
----------------------	--------------------------------

Implements **cms::Session** (p. 2375).

References cms::Session::isTransacted().

6.399.3.27 `virtual void activemq::cmsutil::PooledSession::recover () [inline, virtual]`

Stops message delivery in this session, and restarts message delivery with the oldest unacknowledged message.

All consumers deliver messages in a serial order. Acknowledging a received message automatically acknowledges all messages that have been delivered to the client.

Restarting a session causes it to take the following actions:

- Stop message delivery
- Mark all messages that might have been delivered but not acknowledged as "re-delivered"
- Restart the delivery sequence including all unacknowledged messages that had been previously delivered. Redelivered messages do not have to be delivered in exactly their original delivery order.

Exceptions

<i>CMSException</i>	- if the CMS provider fails to stop and restart message delivery due to some internal error.
<i>IllegalStateException</i>	- if the method is called by a transacted session.

Implements **cms::Session** (p. 2375).

References cms::Session::recover().

6.399.3.28 `virtual void activemq::cmsutil::PooledSession::rollback () [inline, virtual]`

Rolls back all messages done in this transaction and releases any locks currently held.

Exceptions

<i>CMSException</i>	- If an internal error occurs.
<i>IllegalStateException</i>	- if the method is not called by a transacted session.

Implements **cms::Session** (p. 2376).

References cms::Session::rollback().

6.399.3.29 `virtual void activemq::cmsutil::PooledSession::start () [inline, virtual]`

Starts the service.

Exceptions

<i>CMSEException</i>	if an internal error occurs while starting.
----------------------	---

Implements **cms::Startable** (p. 2534).

References cms::Startable::start().

6.399.3.30 **virtual void activemq::cmsutil::PooledSession::stop ()** [inline, virtual]

Stops this service.

Exceptions

<i>CMSEException</i>	- if an internal error occurs while stopping the Service.
----------------------	---

Implements **cms::Stoppable** (p. 2602).

References cms::Stoppable::stop().

6.399.3.31 **virtual void activemq::cmsutil::PooledSession::unsubscribe (const std::string & name)** [inline, virtual]

Unsubscribes a durable subscription that has been created by a client.

This method deletes the state being maintained on behalf of the subscriber by its provider. It is erroneous for a client to delete a durable subscription while there is an active MessageConsumer or Subscriber for the subscription, or while a consumed message is part of a pending transaction or has not been acknowledged in the session.

Parameters

<i>name</i>	The name used to identify this subscription
-------------	---

Exceptions

<i>CMSEException</i>	- If an internal error occurs.
----------------------	--------------------------------

Implements **cms::Session** (p. 2376).

References cms::Session::unsubscribe().

The documentation for this class was generated from the following file:

- src/main/activemq/cmsutil/**PooledSession.h**

6.400 decaf::net::PortUnreachableException Class Reference

```
#include <src/main/decaf/net/PortUnreachableException.h>
```

Inheritance diagram for decaf::net::PortUnreachableException:

Public Member Functions

- **PortUnreachableException** () throw ()
Default Constructor.
- **PortUnreachableException** (const **Exception** &ex) throw ()
Conversion Constructor from some other Exception.
- **PortUnreachableException** (const **PortUnreachableException** &ex) throw ()
Copy Constructor.
- **PortUnreachableException** (const char *file, const int lineNumber, const std::exception *cause, const char *msg,...) throw ()
Constructor - Initializes the file name and line number where this message occurred.
- **PortUnreachableException** (const std::exception *cause) throw ()
Constructor.
- **PortUnreachableException** (const char *file, const int lineNumber, const char *msg,...) throw ()
Constructor - Initializes the file name and line number where this message occurred.
- virtual **PortUnreachableException** * **clone** () const
Clones this exception.
- virtual ~**PortUnreachableException** () throw ()

6.400.1 Constructor & Destructor Documentation

6.400.1.1 **decaf::net::PortUnreachableException::PortUnreachableException ()**
throw () [inline]

Default Constructor.

6.400.1.2 **decaf::net::PortUnreachableException::PortUnreachableException (**
const **Exception** & ex) throw () [inline]

Conversion Constructor from some other Exception.

Parameters

ex	An exception that should become this type of Exception
----	--

6.400.1.3 **decaf::net::PortUnreachableException::PortUnreachableException (**
const PortUnreachableException & ex) throw () `[inline]`

Copy Constructor.

Parameters

<i>ex</i>	An exception that should become this type of Exception
-----------	--

6.400.1.4 **decaf::net::PortUnreachableException::PortUnreachableException (**
const char * file, const int lineNumber, const std::exception * cause, const char * msg, ...) throw () `[inline]`

Constructor - Initializes the file name and line number where this message occurred.

Sets the message to report, using an optional list of arguments to parse into the message

Parameters

<i>file</i>	The file name where exception occurs
<i>lineNumber</i>	The line number where the exception occurred.
<i>cause</i>	The exception that was the cause for this one to be thrown.
<i>msg</i>	The message to report
<i>...</i>	list of primitives that are formatted into the message

6.400.1.5 **decaf::net::PortUnreachableException::PortUnreachableException (**
const std::exception * cause) throw () `[inline]`

Constructor.

Parameters

<i>cause</i>	Pointer to the exception that caused this one to be thrown, the object is cloned caller retains ownership.
--------------	--

6.400.1.6 **decaf::net::PortUnreachableException::PortUnreachableException**
(const char * file, const int lineNumber, const char * msg, ...) throw ()
`[inline]`

Constructor - Initializes the file name and line number where this message occurred.

Sets the message to report, using an optional list of arguments to parse into the message

Parameters

<i>file</i>	The file name where exception occurs
<i>lineNumber</i>	The line number where the exception occurred.
<i>msg</i>	The message to report
...	list of primitives that are formatted into the message

6.400.1.7 `virtual decaf::net::PortUnreachableException::~~PortUnreachableException () throw () [inline, virtual]`

6.400.2 Member Function Documentation

6.400.2.1 `virtual PortUnreachableException* decaf::net::PortUnreachableException::clone () const [inline, virtual]`

Clones this exception.

This is useful for cases where you need to preserve the type of the original exception as well as the message. All subclasses should override.

Returns

a new Exception instance that is a copy of this Exception object.

Reimplemented from **decaf::net::SocketException** (p. 2473).

The documentation for this class was generated from the following file:

- `src/main/decaf/net/PortUnreachableException.h`

6.401 **activemq::core::PrefetchPolicy Class Reference**

Interface for a Policy object that controls message Prefetching on various destination types in ActiveMQ-CPP.

```
#include <src/main/activemq/core/PrefetchPolicy.h>
```

Inheritance diagram for `activemq::core::PrefetchPolicy`:

Public Member Functions

- `virtual ~PrefetchPolicy ()`
- `virtual void setDurableTopicPrefetch (int value)=0`
Sets the amount of prefetched messages for a Durable Topic.

- virtual int **getDurableTopicPrefetch** () const =0
Gets the amount of messages to prefetch for a Durable Topic.
- virtual void **setQueuePrefetch** (int value)=0
Sets the amount of prefetched messages for a Queue.
- virtual int **getQueuePrefetch** () const =0
Gets the amount of messages to prefetch for a Queue.
- virtual void **setQueueBrowserPrefetch** (int value)=0
Sets the amount of prefetched messages for a Queue Browser.
- virtual int **getQueueBrowserPrefetch** () const =0
Gets the amount of messages to prefetch for a Queue Browser.
- virtual void **setTopicPrefetch** (int value)=0
Sets the amount of prefetched messages for a Topic.
- virtual int **getTopicPrefetch** () const =0
Gets the amount of messages to prefetch for a Topic.
- virtual int **getMaxPrefetchLimit** (int value) const =0
Given a requested value for a new prefetch limit, compare it against some max prefetch value and return either the requested value or the maximum allowable value for prefetch.
- virtual **PrefetchPolicy** * **clone** () const =0
Clone the Policy and return a new pointer to that clone.
- virtual void **configure** (const **decaf::util::Properties** &properties)
Checks the supplied properties object for properties matching the configurable settings of this class.

Protected Member Functions

- **PrefetchPolicy** ()

6.401.1 Detailed Description

Interface for a Policy object that controls message Prefetching on various destination types in ActiveMQ-CPP.

Since

3.2.0

6.401.2 Constructor & Destructor Documentation

6.401.2.1 **activemq::core::PrefetchPolicy::PrefetchPolicy** () [protected]

6.401.2.2 **virtual activemq::core::PrefetchPolicy::~PrefetchPolicy** ()
[virtual]

6.401.3 Member Function Documentation

6.401.3.1 `virtual PrefetchPolicy* activemq::core::PrefetchPolicy::clone () const`
`[pure virtual]`

Clone the Policy and return a new pointer to that clone.

Returns

pointer to a new **PrefetchPolicy** (p. 2110) instance that is a clone of this one.

Implemented in **activemq::core::policies::DefaultPrefetchPolicy** (p. 1147).

6.401.3.2 `virtual void activemq::core::PrefetchPolicy::configure (const`
`decaf::util::Properties & properties) [virtual]`

Checks the supplied properties object for properties matching the configurable settings of this class.

The default implementation looks for properties named with the prefix `cms.Prefetch-Policy.XXX` where XXX is the name of a property with a public setter method. For instance `cms.PrefetchPolicy.topicPrefetch` will be used to set the value of the topic prefetch limit.

Subclasses can override this method to add more configuration options or to exclude certain parameters from being set via the properties object.

Parameters

<i>properties</i>	The Properties object used to configure this object.
-------------------	--

Exceptions

<i>NumberFormatException</i>	if a property that is numeric cannot be converted
<i>IllegalArgumentException</i>	if a property can't be converted to the correct type.

6.401.3.3 `virtual int activemq::core::PrefetchPolicy::getDurableTopicPrefetch ()`
`const [pure virtual]`

Gets the amount of messages to prefetch for a Durable Topic.

Returns

value of the number of messages to prefetch.

Implemented in **activemq::core::policies::DefaultPrefetchPolicy** (p. 1147).

6.401.3.4 `virtual int activemq::core::PrefetchPolicy::getMaxPrefetchLimit (int value) const [pure virtual]`

Given a requested value for a new prefetch limit, compare it against some max prefetch value and return either the requested value or the maximum allowable value for prefetch.

Returns

the allowable value for a prefetch limit, either requested or the max.

Implemented in **activemq::core::policies::DefaultPrefetchPolicy** (p. 1147).

6.401.3.5 `virtual int activemq::core::PrefetchPolicy::getQueueBrowserPrefetch () const [pure virtual]`

Gets the amount of messages to prefetch for a Queue Browser.

Returns

value of the number of messages to prefetch.

Implemented in **activemq::core::policies::DefaultPrefetchPolicy** (p. 1148).

6.401.3.6 `virtual int activemq::core::PrefetchPolicy::getQueuePrefetch () const [pure virtual]`

Gets the amount of messages to prefetch for a Queue.

Returns

value of the number of messages to prefetch.

Implemented in **activemq::core::policies::DefaultPrefetchPolicy** (p. 1148).

6.401.3.7 `virtual int activemq::core::PrefetchPolicy::getTopicPrefetch () const [pure virtual]`

Gets the amount of messages to prefetch for a Topic.

Returns

value of the number of messages to prefetch.

Implemented in **activemq::core::policies::DefaultPrefetchPolicy** (p. 1148).

6.401.3.8 `virtual void activemq::core::PrefetchPolicy::setDurableTopicPrefetch (int value)` [pure virtual]

Sets the amount of prefetched messages for a Durable Topic.

Parameters

<i>value</i>	The number of messages to prefetch.
--------------	-------------------------------------

Implemented in **activemq::core::policies::DefaultPrefetchPolicy** (p. 1148).

6.401.3.9 `virtual void activemq::core::PrefetchPolicy::setQueueBrowserPrefetch (int value)` [pure virtual]

Sets the amount of prefetched messages for a Queue Browser.

Parameters

<i>value</i>	The number of messages to prefetch.
--------------	-------------------------------------

Implemented in **activemq::core::policies::DefaultPrefetchPolicy** (p. 1149).

6.401.3.10 `virtual void activemq::core::PrefetchPolicy::setQueuePrefetch (int value)` [pure virtual]

Sets the amount of prefetched messages for a Queue.

Parameters

<i>value</i>	The number of messages to prefetch.
--------------	-------------------------------------

Implemented in **activemq::core::policies::DefaultPrefetchPolicy** (p. 1149).

6.401.3.11 `virtual void activemq::core::PrefetchPolicy::setTopicPrefetch (int value)` [pure virtual]

Sets the amount of prefetched messages for a Topic.

Parameters

<i>value</i>	The number of messages to prefetch.
--------------	-------------------------------------

Implemented in **activemq::core::policies::DefaultPrefetchPolicy** (p. 1149).

The documentation for this class was generated from the following file:

- `src/main/activemq/core/PrefetchPolicy.h`

6.402 activemq::util::PrimitiveList Class Reference

List of primitives.

```
#include <src/main/activemq/util/PrimitiveList.h>
```

Inheritance diagram for activemq::util::PrimitiveList:

Public Member Functions

- **PrimitiveList** ()
Default Constructor, creates an Empty list.
- virtual **~PrimitiveList** ()
- **PrimitiveList** (const **decaf::util::List**< **PrimitiveValueNode** > &src)
Copy Constructor.
- **PrimitiveList** (const **PrimitiveList** &src)
Copy Constructor.
- std::string **toString** () const
Converts the contents into a formatted string that can be output in a Log File or other debugging tool.
- virtual bool **getBool** (int index) const
Gets the Boolean value at the specified index.
- virtual void **setBool** (int index, bool value)
Sets the value at the given index to the new value specified, this method overwrites any data that was previously at the index given, but does not insert a new element if the index is greater than the size of the list.
- virtual unsigned char **getByte** (int index) const
Gets the Byte value at the specified index.
- virtual void **setByte** (int index, unsigned char value)
Sets the value at the given index to the new value specified, this method overwrites any data that was previously at the index given, but does not insert a new element if the index is greater than the size of the list.
- virtual char **getChar** (int index) const
Gets the Character value at the specified index.
- virtual void **setChar** (int index, char value)
Sets the value at the given index to the new value specified, this method overwrites any data that was previously at the index given, but does not insert a new element if the index is greater than the size of the list.
- virtual short **getShort** (int index) const
Gets the Short value at the specified index.
- virtual void **setShort** (int index, short value)
Sets the value at the given index to the new value specified, this method overwrites any data that was previously at the index given, but does not insert a new element if the index is greater than the size of the list.

- virtual int **getInt** (int index) const
Gets the Integer value at the specified index.
- virtual void **setInt** (int index, int value)
Sets the value at the given index to the new value specified, this method overwrites any data that was previously at the index given, but does not insert a new element if the index is greater than the size of the list.
- virtual long long **getLong** (int index) const
Gets the Long value at the specified index.
- virtual void **setLong** (int index, long long value)
Sets the value at the given index to the new value specified, this method overwrites any data that was previously at the index given, but does not insert a new element if the index is greater than the size of the list.
- virtual float **getFloat** (int index) const
Gets the Float value at the specified index.
- virtual void **setFloat** (int index, float value)
Sets the value at the given index to the new value specified, this method overwrites any data that was previously at the index given, but does not insert a new element if the index is greater than the size of the list.
- virtual double **getDouble** (int index) const
Gets the Double value at the specified index.
- virtual void **setDouble** (int index, double value)
Sets the value at the given index to the new value specified, this method overwrites any data that was previously at the index given, but does not insert a new element if the index is greater than the size of the list.
- virtual std::string **getString** (int index) const
Gets the String value at the specified index.
- virtual void **setString** (int index, const std::string &value)
Sets the value at the given index to the new value specified, this method overwrites any data that was previously at the index given, but does not insert a new element if the index is greater than the size of the list.
- virtual std::vector< unsigned char > **getByteArray** (int index) const
Gets the Byte Array value at the specified index.
- virtual void **setByteArray** (int index, const std::vector< unsigned char > &value)
Sets the value at the given index to the new value specified, this method overwrites any data that was previously at the index given, but does not insert a new element if the index is greater than the size of the list.

6.402.1 Detailed Description

List of primitives.

6.402.2 Constructor & Destructor Documentation

6.402.2.1 activemq::util::PrimitiveList::PrimitiveList ()

Default Constructor, creates an Empty list.

6.402.2.2 `virtual activemq::util::PrimitiveList::~~PrimitiveList () [virtual]`

6.402.2.3 `activemq::util::PrimitiveList::PrimitiveList (const decaf::util::List< PrimitiveValueNode > & src)`

Copy Constructor.

Parameters

<i>src</i>	- the Decaf List of PrimitiveNodeValues to copy
------------	---

6.402.2.4 `activemq::util::PrimitiveList::PrimitiveList (const PrimitiveList & src)`

Copy Constructor.

Parameters

<i>src</i>	- the PrimitiveList (p. 2114) to copy
------------	--

6.402.3 Member Function Documentation

6.402.3.1 `virtual bool activemq::util::PrimitiveList::getBool (int index) const [virtual]`

Gets the Boolean value at the specified index.

Parameters

<i>index</i>	- index to get value from
--------------	---------------------------

Returns

value contained at the given index

Exceptions

<i>IndexOutOfBoundsException</i>	if index is > size() (p. 1658)
<i>UnsupportedOperationException</i>	if the type at index is not of the type that this method is to return or can convert to.

6.402.3.2 `virtual unsigned char activemq::util::PrimitiveList::getByte (int index) const [virtual]`

Gets the Byte value at the specified index.

Parameters

<i>index</i>	- index to get value from
--------------	---------------------------

Returns

value contained at the given index

Exceptions

<i>IndexOutOfBoundsException</i>	if index is > size() (p. 1658)
<i>UnsupportedOperationException</i>	if the type at index is not of the type that this method is to return or can convert to.

6.402.3.3 **virtual std::vector<unsigned char> activemq::util::PrimitiveList::getBytesArray (int *index*) const** [virtual]

Gets the Byte Array value at the specified index.

Parameters

<i>index</i>	- index to get value from
--------------	---------------------------

Returns

value contained at the given index

Exceptions

<i>IndexOutOfBoundsException</i>	if index is > size() (p. 1658)
<i>UnsupportedOperationException</i>	if the type at index is not of the type that this method is to return or can convert to.

6.402.3.4 **virtual char activemq::util::PrimitiveList::getChar (int *index*) const** [virtual]

Gets the Character value at the specified index.

Parameters

<i>index</i>	- index to get value from
--------------	---------------------------

Returns

value contained at the given index

Exceptions

<i>IndexOutOfBoundsException</i>	if index is > size() (p. 1658)
<i>UnsupportedOperationException</i>	if the type at index is not of the type that this method is to return or can convert to.

6.402.3.5 virtual double **activemq::util::PrimitiveList::getDouble** (int *index*) const
[virtual]

Gets the Double value at the specified index.

Parameters

<i>index</i>	- index to get value from
--------------	---------------------------

Returns

value contained at the given index

Exceptions

<i>IndexOutOfBoundsException</i>	if index is > size() (p. 1658)
<i>UnsupportedOperationException</i>	if the type at index is not of the type that this method is to return or can convert to.

6.402.3.6 virtual float **activemq::util::PrimitiveList::getFloat** (int *index*) const
[virtual]

Gets the Float value at the specified index.

Parameters

<i>index</i>	- index to get value from
--------------	---------------------------

Returns

value contained at the given index

Exceptions

<i>IndexOutOfBoundsException</i>	if index is > size() (p. 1658)
<i>UnsupportedOperationException</i>	if the type at index is not of the type that this method is to return or can convert to.

6.402.3.7 `virtual int activemq::util::PrimitiveList::getInt (int index) const`
`[virtual]`

Gets the Integer value at the specified index.

Parameters

<i>index</i>	- index to get value from
--------------	---------------------------

Returns

value contained at the given index

Exceptions

<i>IndexOutOfBoundsException</i>	if index is > size() (p. 1658)
<i>UnsupportedOperationException</i>	if the type at index is not of the type that this method is to return or can convert to.

6.402.3.8 `virtual long long activemq::util::PrimitiveList::getLong (int index) const`
`[virtual]`

Gets the Long value at the specified index.

Parameters

<i>index</i>	- index to get value from
--------------	---------------------------

Returns

value contained at the given index

Exceptions

<i>IndexOutOfBoundsException</i>	if index is > size() (p. 1658)
<i>UnsupportedOperationException</i>	if the type at index is not of the type that this method is to return or can convert to.

6.402.3.9 `virtual short activemq::util::PrimitiveList::getShort (int index) const`
[virtual]

Gets the Short value at the specified index.

Parameters

<i>index</i>	- index to get value from
--------------	---------------------------

Returns

value contained at the given index

Exceptions

<i>IndexOutOfBoundsException</i>	if index is > size() (p. 1658)
<i>UnsupportedOperationException</i>	if the type at index is not of the type that this method is to return or can convert to.

6.402.3.10 `virtual std::string activemq::util::PrimitiveList::getString (int index) const`
[virtual]

Gets the String value at the specified index.

Parameters

<i>index</i>	- index to get value from
--------------	---------------------------

Returns

value contained at the given index

Exceptions

<i>IndexOutOfBoundsException</i>	if index is > size() (p. 1658)
<i>UnsupportedOperationException</i>	if the type at index is not of the type that this method is to return or can convert to.

6.402.3.11 `virtual void activemq::util::PrimitiveList::setBool (int index, bool value)`
[virtual]

Sets the value at the given index to the new value specified, this method overwrites any data that was previously at the index given, but does not insert a new element if the index is greater than the size of the list.

Parameters

<i>index</i>	- location to set in the list
<i>value</i>	- the new value to assign to the element at index

Exceptions

<i>IndexOutOfBoundsException</i>	if index > size() (p. 1658).
----------------------------------	-------------------------------------

6.402.3.12 virtual void **activemq::util::PrimitiveList::setByte** (int *index*, unsigned char *value*) [virtual]

Sets the value at the given index to the new value specified, this method overwrites any data that was previously at the index given, but does not insert a new element if the index is greater than the size of the list.

Parameters

<i>index</i>	- location to set in the list
<i>value</i>	- the new value to assign to the element at index

Exceptions

<i>IndexOutOfBoundsException</i>	if index > size() (p. 1658).
----------------------------------	-------------------------------------

6.402.3.13 virtual void **activemq::util::PrimitiveList::setByteArray** (int *index*, const std::vector< unsigned char > & *value*) [virtual]

Sets the value at the given index to the new value specified, this method overwrites any data that was previously at the index given, but does not insert a new element if the index is greater than the size of the list.

Parameters

<i>index</i>	- location to set in the list
<i>value</i>	- the new value to assign to the element at index

Exceptions

<i>IndexOutOfBoundsException</i>	if index > size() (p. 1658).
----------------------------------	-------------------------------------

6.402.3.14 `virtual void activemq::util::PrimitiveList::setChar (int index, char value)`
[virtual]

Sets the value at the given index to the new value specified, this method overwrites any data that was previously at the index given, but does not insert a new element if the index is greater than the size of the list.

Parameters

<i>index</i>	- location to set in the list
<i>value</i>	- the new value to assign to the element at index

Exceptions

<i>IndexOutOfBoundsException</i>	if index > size() (p. 1658).
----------------------------------	-------------------------------------

6.402.3.15 `virtual void activemq::util::PrimitiveList::setDouble (int index, double value)`
[virtual]

Sets the value at the given index to the new value specified, this method overwrites any data that was previously at the index given, but does not insert a new element if the index is greater than the size of the list.

Parameters

<i>index</i>	- location to set in the list
<i>value</i>	- the new value to assign to the element at index

Exceptions

<i>IndexOutOfBoundsException</i>	if index > size() (p. 1658).
----------------------------------	-------------------------------------

6.402.3.16 `virtual void activemq::util::PrimitiveList::setFloat (int index, float value)`
[virtual]

Sets the value at the given index to the new value specified, this method overwrites any data that was previously at the index given, but does not insert a new element if the index is greater than the size of the list.

Parameters

<i>index</i>	- location to set in the list
<i>value</i>	- the new value to assign to the element at index

Exceptions

<i>IndexOutOfBoundsException</i>	if index > size() (p. 1658).
----------------------------------	-------------------------------------

6.402.3.17 `virtual void activemq::util::PrimitiveList::setInt (int index, int value)`
`[virtual]`

Sets the value at the given index to the new value specified, this method overwrites any data that was previously at the index given, but does not insert a new element if the index is greater than the size of the list.

Parameters

<i>index</i>	- location to set in the list
<i>value</i>	- the new value to assign to the element at index

Exceptions

<i>IndexOutOfBoundsException</i>	if index > size() (p. 1658).
----------------------------------	-------------------------------------

6.402.3.18 `virtual void activemq::util::PrimitiveList::setLong (int index, long long value)`
`[virtual]`

Sets the value at the given index to the new value specified, this method overwrites any data that was previously at the index given, but does not insert a new element if the index is greater than the size of the list.

Parameters

<i>index</i>	- location to set in the list
<i>value</i>	- the new value to assign to the element at index

Exceptions

<i>IndexOutOfBoundsException</i>	if index > size() (p. 1658).
----------------------------------	-------------------------------------

6.402.3.19 `virtual void activemq::util::PrimitiveList::setShort (int index, short value)`
`[virtual]`

Sets the value at the given index to the new value specified, this method overwrites any data that was previously at the index given, but does not insert a new element if the index is greater than the size of the list.

Parameters

<i>index</i>	- location to set in the list
<i>value</i>	- the new value to assign to the element at index

Exceptions

<i>IndexOutOfBoundsException</i>	if index > size() (p. 1658).
----------------------------------	-------------------------------------

6.402.3.20 virtual void activemq::util::PrimitiveList::setString (int *index*, const std::string & *value*) [virtual]

Sets the value at the given index to the new value specified, this method overwrites any data that was previously at the index given, but does not insert a new element if the index is greater than the size of the list.

Parameters

<i>index</i>	- location to set in the list
<i>value</i>	- the new value to assign to the element at index

Exceptions

<i>IndexOutOfBoundsException</i>	if index > size() (p. 1658).
----------------------------------	-------------------------------------

6.402.3.21 std::string activemq::util::PrimitiveList::toString () const

Converts the contents into a formatted string that can be output in a Log File or other debugging tool.

Returns

formatted String of all elements in the list.

The documentation for this class was generated from the following file:

- src/main/activemq/util/**PrimitiveList.h**

6.403 activemq::util::PrimitiveMap Class Reference

Map of named primitives.

```
#include <src/main/activemq/util/PrimitiveMap.h>
```

Inheritance diagram for `activemq::util::PrimitiveMap`:

Public Member Functions

- **PrimitiveMap** ()
Default Constructor, creates an empty map.
- virtual **~PrimitiveMap** ()
- **PrimitiveMap** (const **decaf::util::Map**< std::string, **PrimitiveValueNode** > &source)
Copy Constructor.
- **PrimitiveMap** (const **PrimitiveMap** &source)
Copy Constructor.
- std::string **toString** () const
Converts the contents into a formatted string that can be output in a Log File or other debugging tool.
- virtual bool **getBool** (const std::string &key) const
Gets the Boolean value at the given key, if the key is not in the map or cannot be returned as the requested value then an exception of type `NoSuchElementException` is thrown.
- virtual void **setBool** (const std::string &key, bool value)
Sets the value at key to the specified type.
- virtual unsigned char **getByte** (const std::string &key) const
Gets the Byte value at the given key, if the key is not in the map or cannot be returned as the requested value then an exception of type `NoSuchElementException` is thrown.
- virtual void **setByte** (const std::string &key, unsigned char value)
Sets the value at key to the specified type.
- virtual char **getChar** (const std::string &key) const
Gets the Character value at the given key, if the key is not in the map or cannot be returned as the requested value then an exception of type `NoSuchElementException` is thrown.
- virtual void **setChar** (const std::string &key, char value)
Sets the value at key to the specified type.
- virtual short **getShort** (const std::string &key) const
Gets the Short value at the given key, if the key is not in the map or cannot be returned as the requested value then an exception of type `NoSuchElementException` is thrown.
- virtual void **setShort** (const std::string &key, short value)
Sets the value at key to the specified type.
- virtual int **getInt** (const std::string &key) const
Gets the Integer value at the given key, if the key is not in the map or cannot be returned as the requested value then an exception of type `NoSuchElementException` is thrown.
- virtual void **setInt** (const std::string &key, int value)
Sets the value at key to the specified type.

- virtual long long **getLong** (const std::string &key) const
Gets the Long value at the given key, if the key is not in the map or cannot be returned as the requested value then an exception of type NoSuchElementException is thrown.
- virtual void **setLong** (const std::string &key, long long value)
Sets the value at key to the specified type.
- virtual float **getFloat** (const std::string &key) const
Gets the Float value at the given key, if the key is not in the map or cannot be returned as the requested value then an exception of type NoSuchElementException is thrown.
- virtual void **setFloat** (const std::string &key, float value)
Sets the value at key to the specified type.
- virtual double **getDouble** (const std::string &key) const
Gets the Double value at the given key, if the key is not in the map or cannot be returned as the requested value then an exception of type NoSuchElementException is thrown.
- virtual void **setDouble** (const std::string &key, double value)
Sets the value at key to the specified type.
- virtual std::string **getString** (const std::string &key) const
Gets the String value at the given key, if the key is not in the map or cannot be returned as the requested value then an exception of type NoSuchElementException is thrown.
- virtual void **setString** (const std::string &key, const std::string &value)
Sets the value at key to the specified type.
- virtual std::vector< unsigned char > **getByteArray** (const std::string &key) const
Gets the Byte Array value at the given key, if the key is not in the map or cannot be returned as the requested value then an exception of type NoSuchElementException is thrown.
- virtual void **setByteArray** (const std::string &key, const std::vector< unsigned char > &value)
Sets the value at key to the specified type.

6.403.1 Detailed Description

Map of named primitives.

6.403.2 Constructor & Destructor Documentation

6.403.2.1 activemq::util::PrimitiveMap::PrimitiveMap ()

Default Constructor, creates an empty map.

6.403.2.2 virtual activemq::util::PrimitiveMap::~~PrimitiveMap () [virtual]

6.403.2.3 activemq::util::PrimitiveMap::PrimitiveMap (const decaf::util::Map< std::string, PrimitiveValueNode > &source)

Copy Constructor.

Parameters

<i>source</i>	The Decaf Library Map instance whose elements will be copied into this Map.
---------------	---

6.403.2.4 `activemq::util::PrimitiveMap::PrimitiveMap (const PrimitiveMap & source)`

Copy Constructor.

Parameters

<i>source</i>	The PrimitiveMap (p.2125) whose elements will be copied into this Map.
---------------	---

6.403.3 Member Function Documentation

6.403.3.1 `virtual bool activemq::util::PrimitiveMap::getBool (const std::string & key) const [virtual]`

Gets the Boolean value at the given key, if the key is not in the map or cannot be returned as the requested value then an exception of type `NoSuchElementException` is thrown.

Parameters

<i>key</i>	- the location to return the value from.
------------	--

Returns

the value at key in the type requested.

Exceptions

<i>NoSuchElementException</i>	if key is not in the map.
<i>UnsupportedOperationException</i>	if the value cannot be converted to the type this method returns

6.403.3.2 `virtual unsigned char activemq::util::PrimitiveMap::getByte (const std::string & key) const [virtual]`

Gets the Byte value at the given key, if the key is not in the map or cannot be returned as the requested value then an exception of type `NoSuchElementException` is thrown.

Parameters

<i>key</i>	- the location to return the value from.
------------	--

Returns

the value at key in the type requested.

Exceptions

<i>NoSuchElementException</i>	if key is not in the map.
<i>UnsupportedOperationException</i>	if the value cannot be converted to the type this method returns

6.403.3.3 `virtual std::vector<unsigned char> activemq::util::PrimitiveMap::getByteArray (const std::string & key) const`
`[virtual]`

Gets the Byte Array value at the given key, if the key is not in the map or cannot be returned as the requested value then an exception of type `NoSuchElementException` is thrown.

Parameters

<i>key</i>	- the location to return the value from.
------------	--

Returns

the value at key in the type requested.

Exceptions

<i>NoSuchElementException</i>	if key is not in the map.
<i>UnsupportedOperationException</i>	if the value cannot be converted to the type this method returns

6.403.3.4 `virtual char activemq::util::PrimitiveMap::getChar (const std::string & key) const`
`[virtual]`

Gets the Character value at the given key, if the key is not in the map or cannot be returned as the requested value then an exception of type `NoSuchElementException` is thrown.

Parameters

<i>key</i>	- the location to return the value from.
------------	--

Returns

the value at key in the type requested.

Exceptions

<i>NoSuchElementException</i>	if key is not in the map.
<i>UnsupportedOperationException</i>	if the value cannot be converted to the type this method returns

6.403.3.5 virtual double **activemq::util::PrimitiveMap::getDouble** (const std::string & *key*) const [virtual]

Gets the Double value at the given key, if the key is not in the map or cannot be returned as the requested value then an exception of type *NoSuchElementException* is thrown.

Parameters

<i>key</i>	- the location to return the value from.
------------	--

Returns

the value at key in the type requested.

Exceptions

<i>NoSuchElementException</i>	if key is not in the map.
<i>UnsupportedOperationException</i>	if the value cannot be converted to the type this method returns

6.403.3.6 virtual float **activemq::util::PrimitiveMap::getFloat** (const std::string & *key*) const [virtual]

Gets the Float value at the given key, if the key is not in the map or cannot be returned as the requested value then an exception of type *NoSuchElementException* is thrown.

Parameters

<i>key</i>	- the location to return the value from.
------------	--

Returns

the value at key in the type requested.

Exceptions

<i>NoSuchElementException</i>	if key is not in the map.
<i>UnsupportedOperationException</i>	if the value cannot be converted to the type this method returns

6.403.3.7 `virtual int activemq::util::PrimitiveMap::getInt (const std::string & key) const`
[virtual]

Gets the Integer value at the given key, if the key is not in the map or cannot be returned as the requested value then an exception of type `NoSuchElementException` is thrown.

Parameters

<i>key</i>	- the location to return the value from.
------------	--

Returns

the value at key in the type requested.

Exceptions

<i>NoSuchElementException</i>	if key is not in the map.
<i>UnsupportedOperationException</i>	if the value cannot be converted to the type this method returns

6.403.3.8 `virtual long long activemq::util::PrimitiveMap::getLong (const std::string & key) const`
[virtual]

Gets the Long value at the given key, if the key is not in the map or cannot be returned as the requested value then an exception of type `NoSuchElementException` is thrown.

Parameters

<i>key</i>	- the location to return the value from.
------------	--

Returns

the value at key in the type requested.

Exceptions

<i>NoSuchElementException</i>	if key is not in the map.
<i>UnsupportedOperationException</i>	if the value cannot be converted to the type this method returns

6.403.3.9 virtual short **activemq::util::PrimitiveMap::getShort** (const std::string & key)
const [virtual]

Gets the Short value at the given key, if the key is not in the map or cannot be returned as the requested value then an exception of type `NoSuchElementException` is thrown.

Parameters

<i>key</i>	- the location to return the value from.
------------	--

Returns

the value at key in the type requested.

Exceptions

<i>NoSuchElementException</i>	if key is not in the map.
<i>UnsupportedOperationException</i>	if the value cannot be converted to the type this method returns

6.403.3.10 virtual std::string **activemq::util::PrimitiveMap::getString** (const std::string & key) const [virtual]

Gets the String value at the given key, if the key is not in the map or cannot be returned as the requested value then an exception of type `NoSuchElementException` is thrown.

Parameters

<i>key</i>	- the location to return the value from.
------------	--

Returns

the value at key in the type requested.

Exceptions

<i>NoSuchElementException</i>	if key is not in the map.
<i>UnsupportedOperationException</i>	if the value cannot be converted to the type this method returns

6.403.3.11 virtual void **activemq::util::PrimitiveMap::setBool** (const std::string & *key*,
bool *value*) [virtual]

Sets the value at key to the specified type.

Overwrites any data that was previously at this key or inserts a new element at key.

Parameters

<i>key</i>	- the map key to set or insert.
<i>value</i>	- the new value to set at the key location.

6.403.3.12 virtual void **activemq::util::PrimitiveMap::setByte** (const std::string & *key*,
unsigned char *value*) [virtual]

Sets the value at key to the specified type.

Overwrites any data that was previously at this key or inserts a new element at key.

Parameters

<i>key</i>	- the map key to set or insert.
<i>value</i>	- the new value to set at the key location.

6.403.3.13 virtual void **activemq::util::PrimitiveMap::setByteArray** (const std::string &
key, const std::vector< unsigned char > & *value*) [virtual]

Sets the value at key to the specified type.

Overwrites any data that was previously at this key or inserts a new element at key.

Parameters

<i>key</i>	- the map key to set or insert.
<i>value</i>	- the new value to set at the key location.

6.403.3.14 virtual void **activemq::util::PrimitiveMap::setChar** (const std::string & *key*,
char *value*) [virtual]

Sets the value at key to the specified type.

Overwrites any data that was previously at this key or inserts a new element at key.

Parameters

<i>key</i>	- the map key to set or insert.
<i>value</i>	- the new value to set at the key location.

6.403.3.15 **virtual void activemq::util::PrimitiveMap::setDouble** (const std::string & *key*, double *value*) [virtual]

Sets the value at key to the specified type.

Overwrites any data that was previously at this key or inserts a new element at key.

Parameters

<i>key</i>	- the map key to set or insert.
<i>value</i>	- the new value to set at the key location.

6.403.3.16 **virtual void activemq::util::PrimitiveMap::setFloat** (const std::string & *key*, float *value*) [virtual]

Sets the value at key to the specified type.

Overwrites any data that was previously at this key or inserts a new element at key.

Parameters

<i>key</i>	- the map key to set or insert.
<i>value</i>	- the new value to set at the key location.

6.403.3.17 **virtual void activemq::util::PrimitiveMap::setInt** (const std::string & *key*, int *value*) [virtual]

Sets the value at key to the specified type.

Overwrites any data that was previously at this key or inserts a new element at key.

Parameters

<i>key</i>	- the map key to set or insert.
<i>value</i>	- the new value to set at the key location.

6.403.3.18 **virtual void activemq::util::PrimitiveMap::setLong** (const std::string & *key*, long long *value*) [virtual]

Sets the value at key to the specified type.

Overwrites any data that was previously at this key or inserts a new element at key.

Parameters

<i>key</i>	- the map key to set or insert.
<i>value</i>	- the new value to set at the key location.

6.404 activemq::wireformat::openwire::marshal::PrimitiveTypesMarshaller Class Reference 2141

6.403.3.19 `virtual void activemq::util::PrimitiveMap::setShort (const std::string & key, short value) [virtual]`

Sets the value at key to the specified type.

Overwrites any data that was previously at this key or inserts a new element at key.

Parameters

<i>key</i>	- the map key to set or insert.
<i>value</i>	- the new value to set at the key location.

6.403.3.20 `virtual void activemq::util::PrimitiveMap::setString (const std::string & key, const std::string & value) [virtual]`

Sets the value at key to the specified type.

Overwrites any data that was previously at this key or inserts a new element at key.

Parameters

<i>key</i>	- the map key to set or insert.
<i>value</i>	- the new value to set at the key location.

6.403.3.21 `std::string activemq::util::PrimitiveMap::toString () const`

Converts the contents into a formatted string that can be output in a Log File or other debugging tool.

Returns

formatted String of all elements in the map.

The documentation for this class was generated from the following file:

- `src/main/activemq/util/PrimitiveMap.h`

6.404 activemq::wireformat::openwire::marshal::PrimitiveTypesMarshaller Class Reference

This class wraps the functionality needed to marshal a primitive map to the Openwire Format's expectation of what the map looks like on the wire.

```
#include <src/main/activemq/wireformat/openwire/marshal/-  
PrimitiveTypesMarshaller.h>
```

Public Member Functions

- **PrimitiveTypesMarshaller** ()
- virtual **~PrimitiveTypesMarshaller** ()

Static Public Member Functions

- static void **marshal** (const **util::PrimitiveMap** *map, std::vector< unsigned char > &buffer)
Marshal a primitive map object to the given byte buffer.
- static void **unmarshal** (**util::PrimitiveMap** *map, const std::vector< unsigned char > &buffer)
Unmarshal a PrimitiveMap from the provided Byte buffer.
- static void **marshal** (const **util::PrimitiveList** *list, std::vector< unsigned char > &buffer)
Marshal a primitive list object to the given byte buffer.
- static void **unmarshal** (**util::PrimitiveList** *list, const std::vector< unsigned char > &buffer)
Unmarshal a PrimitiveList from the provided byte buffer.
- static void **marshalMap** (const **util::PrimitiveMap** *map, **decaf::io::DataOutputStream** &dataOut)
Marshal a primitive map object to the given DataOutputStream.
- static **util::PrimitiveMap** * **unmarshalMap** (**decaf::io::DataInputStream** &dataIn)
Unmarshal a PrimitiveMap from the provided DataInputStream.
- static void **marshalList** (const **util::PrimitiveList** *list, **decaf::io::DataOutputStream** &dataOut)
Marshal a PrimitiveList to the given DataOutputStream.
- static **util::PrimitiveList** * **unmarshalList** (**decaf::io::DataInputStream** &dataIn)
Unmarshal a PrimitiveList from the given DataInputStream.

Static Protected Member Functions

- static void **marshalPrimitiveMap** (**decaf::io::DataOutputStream** &dataOut, const **decaf::util::Map**< std::string, **util::PrimitiveValueNode** > &map)
Marshal a Map of Primitives to the given OutputStream, can result in recursive calls to this method if the map contains maps of maps.
- static void **marshalPrimitiveList** (**decaf::io::DataOutputStream** &dataOut, const **decaf::util::List**< **util::PrimitiveValueNode** > &list)
Marshal a List of Primitives to the given OutputStream, can result in recursive calls to this method if the list contains lists of lists.
- static void **marshalPrimitive** (**decaf::io::DataOutputStream** &dataOut, const **util::PrimitiveValueNode** &value)
Used to Marshal the Primitive types out on the Wire.

- static void **unmarshalPrimitiveMap** (**decaf::io::DataInputStream** &dataIn, **util::PrimitiveMap** &map)
Unmarshals a Map of Primitives from the given InputStream, can result in recursive calls to this method if the map contains maps of maps.
- static void **unmarshalPrimitiveList** (**decaf::io::DataInputStream** &dataIn, **decaf::util::LinkedList< util::PrimitiveValueNode >** &list)
Unmarshals a List of Primitives from the given InputStream, can result in recursive calls to this method if the list contains lists of lists.
- static **util::PrimitiveValueNode** **unmarshalPrimitive** (**decaf::io::DataInputStream** &dataIn)
Unmarshals a Primitive Type from the stream, and returns it as a value Node.

6.404.1 Detailed Description

This class wraps the functionality needed to marshal a primitive map to the Openwire Format's expectation of what the map looks like on the wire.

6.404.2 Constructor & Destructor Documentation

- 6.404.2.1 **activemq::wireformat::openwire::marshal::PrimitiveTypesMarshaller::PrimitiveTypesMarshaller ()**
[inline]
- 6.404.2.2 **virtual activemq::wireformat::openwire::marshal::PrimitiveTypesMarshaller::~~PrimitiveTypesMarshaller ()** [inline, virtual]

6.404.3 Member Function Documentation

- 6.404.3.1 **static void activemq::wireformat::openwire::marshal::PrimitiveTypesMarshaller::marshal (const util::PrimitiveMap * map, std::vector< unsigned char > & buffer)** [static]

Marshal a primitive map object to the given byte buffer.

Parameters

<i>map</i>	Map to Marshal.
<i>buffer</i>	The byte buffer to write the marshaled data to.

Exceptions

<i>Exception</i>	if an error occurs during the marshaling process.
------------------	---

6.404.3.2 `static void activemq::wireformat::openwire::marshal::PrimitiveTypesMarshaller::marshal (const util::PrimitiveList * list, std::vector< unsigned char > & buffer) [static]`

Marshal a primitive list object to the given byte buffer.

Parameters

<i>map</i>	The PrimitiveList to Marshal.
<i>buffer</i>	The byte buffer to write the marshaled data to.

Exceptions

<i>Exception</i>	if an error occurs during the marshaling process.
------------------	---

6.404.3.3 `static void activemq::wireformat::openwire::marshal::PrimitiveTypesMarshaller::marshalList (const util::PrimitiveList * list, decaf::io::DataOutputStream & dataOut) [static]`

Marshal a PrimitiveList to the given DataOutputStream.

Parameters

<i>list</i>	The list object to Marshal
<i>dataOut</i>	Reference to a DataOutputStream to write the marshaled data to.

Exceptions

<i>Exception</i>	if an error occurs during the marshaling process.
------------------	---

6.404.3.4 `static void activemq::wireformat::openwire::marshal::PrimitiveTypesMarshaller::marshalMap (const util::PrimitiveMap * map, decaf::io::DataOutputStream & dataOut) [static]`

Marshal a primitive map object to the given DataOutputStream.

Parameters

<i>map</i>	Map to Marshal.
<i>dataOut</i>	Reference to a DataOutputStream to write the marshaled data to.

Exceptions

<i>Exception</i>	if an error occurs during the marshaling process.
------------------	---

6.404 activemq::wireformat::openwire::marshal::PrimitiveTypesMarshaller Class Reference 2145

6.404.3.5 **static void** **activemq::wireformat::openwire::marshal::PrimitiveTypesMarshaller::marshalPrimitive** (**decaf::io::DataOutputStream** & *dataOut*, **const util::PrimitiveValueNode** & *value*) [static, protected]

Used to Marshal the Primitive types out on the Wire.

Parameters

<i>dataOut</i>	- the DataOutputStream to write to
<i>value</i>	- the ValueNode to write.

Exceptions

<i>IOException</i>	if an I/O error occurs during this operation.
--------------------	---

6.404.3.6 **static void** **activemq::wireformat::openwire::marshal::PrimitiveTypesMarshaller::marshalPrimitiveList** (**decaf::io::DataOutputStream** & *dataOut*, **const decaf::util::List**< **util::PrimitiveValueNode** > & *list*) [static, protected]

Marshal a List of Primitives to the given OutputStream, can result in recursive calls to this method if the list contains lists of lists.

Parameters

<i>dataOut</i>	- the DataOutputStream to write to
<i>list</i>	- the ValueNode to write.

Exceptions

<i>IOException</i>	if an I/O error occurs during this operation.
--------------------	---

6.404.3.7 **static void** **activemq::wireformat::openwire::marshal::PrimitiveTypesMarshaller::marshalPrimitiveMap** (**decaf::io::DataOutputStream** & *dataOut*, **const decaf::util::Map**< **std::string**, **util::PrimitiveValueNode** > & *map*) [static, protected]

Marshal a Map of Primitives to the given OutputStream, can result in recursive calls to this method if the map contains maps of maps.

Parameters

<i>dataOut</i>	- the DataOutputStream to write to
<i>map</i>	- the ValueNode to write.

Exceptions

<i>IOException</i>	if an I/O error occurs during this operation.
--------------------	---

6.404.3.8 **static void activemq::wireformat::openwire::marshal::PrimitiveTypes-Marshaller::unmarshal (util::PrimitiveMap * *map*, const std::vector< unsigned char > & *buffer*) [static]**

Unmarshal a PrimitiveMap from the provided Byte buffer.

Parameters

<i>map</i>	The Map to populate with values from the marshaled data.
<i>buffer</i>	The byte buffer containing the marshaled Map.

Exceptions

<i>Exception</i>	if an error occurs during the unmarshal process.
------------------	--

6.404.3.9 **static void activemq::wireformat::openwire::marshal::PrimitiveTypes-Marshaller::unmarshal (util::PrimitiveList * *list*, const std::vector< unsigned char > & *buffer*) [static]**

Unmarshal a PrimitiveList from the provided byte buffer.

Parameters

<i>map</i>	The List to populate with values from the marshaled data.
<i>buffer</i>	The byte buffer containing the marshaled Map.

Exceptions

<i>Exception</i>	if an error occurs during the unmarshal process.
------------------	--

6.404.3.10 **static util::PrimitiveList* activemq::wireformat::openwire::marshal::PrimitiveTypesMarshaller::unmarshalList (decaf::io::DataInputStream & *dataIn*) [static]**

Unmarshal a PrimitiveList from the given DataInputStream.

Parameters

<i>dataIn</i>	The DataInputStream instance to read the marshaled PrimitiveList from.
---------------	--

6.404 activemq::wireformat::openwire::marshal::PrimitiveTypesMarshaller Class Reference 2147

Returns

a pointer to a newly allocated PrimitiveList instance.

Exceptions

<i>Exception</i>	if an error occurs during the unmarshal process.
------------------	--

6.404.3.11 static util::PrimitiveMap* activemq::wireformat::openwire::marshal::PrimitiveTypesMarshaller::unmarshalMap (decaf::io::DataInputStream & dataIn) [static]

Unmarshal a PrimitiveMap from the provided DataInputStream.

Parameters

<i>dataIn</i>	The DataInputStream instance to read the marshaled PrimitiveMap from.
---------------	---

Returns

a pointer to a newly allocated PrimitiveMap instance.

Exceptions

<i>Exception</i>	if an error occurs during the unmarshal process.
------------------	--

6.404.3.12 static util::PrimitiveValueNode activemq::wireformat::openwire::marshal::PrimitiveTypesMarshaller::unmarshalPrimitive (decaf::io::DataInputStream & dataIn) [static, protected]

Unmarshals a Primitive Type from the stream, and returns it as a value Node.

Parameters

<i>dataIn</i>	- DataInputStream to read from.
---------------	---------------------------------

Returns

a PrimitiveValueNode containing the data.

Exceptions

<i>IOException</i>	if an I/O error occurs during this operation.
--------------------	---

6.404.3.13 **static void activemq::wireformat::openwire::marshal::PrimitiveTypes-Marshaller::unmarshalPrimitiveList (decaf::io::DataInputStream & dataIn, decaf::util::LinkedList< util::PrimitiveValueNode > & list)** [static, protected]

Unmarshals a List of Primitives from the given InputStream, can result in recursive calls to this method if the list contains lists of lists.

Parameters

<i>dataIn</i>	- DataInputStream to read from.
<i>list</i>	- the ValueNode to write.

Exceptions

<i>IOException</i>	if an I/O error occurs during this operation.
--------------------	---

6.404.3.14 **static void activemq::wireformat::openwire::marshal::PrimitiveTypes-Marshaller::unmarshalPrimitiveMap (decaf::io::DataInputStream & dataIn, util::PrimitiveMap & map)** [static, protected]

Unmarshals a Map of Primitives from the given InputStream, can result in recursive calls to this method if the map contains maps of maps.

Parameters

<i>dataIn</i>	- DataInputStream to read from.
<i>map</i>	- the map to fill with data.

Exceptions

<i>IOException</i>	if an I/O error occurs during this operation.
--------------------	---

The documentation for this class was generated from the following file:

- src/main/activemq/wireformat/openwire/marshal/**PrimitiveTypesMarshaller.h**

6.405 activemq::util::PrimitiveValueNode::PrimitiveValue Union - Reference

Define a union type comprised of the various types.

```
#include <src/main/activemq/util/PrimitiveValueNode.h>
```

Data Fields

- bool **boolValue**
- unsigned char **byteValue**
- char **charValue**
- short **shortValue**
- int **intValue**
- long long **longValue**
- double **doubleValue**
- float **floatValue**
- std::string * **stringValue**
- std::vector< unsigned char > * **byteArrayValue**
- **decaf::util::List** < **PrimitiveValueNode** > * **listValue**
- **decaf::util::Map**< std::string, **PrimitiveValueNode** > * **mapValue**

6.405.1 Detailed Description

Define a union type comprised of the various types.

6.405.2 Field Documentation

6.405.2.1 bool **activemq::util::PrimitiveValueNode::PrimitiveValue::boolValue**

6.405.2.2 **std::vector<unsigned char>*** **activemq::util::PrimitiveValueNode::PrimitiveValue::byteArrayValue**

6.405.2.3 unsigned char **activemq::util::PrimitiveValueNode::PrimitiveValue::byteValue**

6.405.2.4 char **activemq::util::PrimitiveValueNode::PrimitiveValue::charValue**

6.405.2.5 double **activemq::util::PrimitiveValueNode::PrimitiveValue::doubleValue**

6.405.2.6 float **activemq::util::PrimitiveValueNode::PrimitiveValue::floatValue**

6.405.2.7 int **activemq::util::PrimitiveValueNode::PrimitiveValue::intValue**

6.405.2.8 **decaf::util::List<PrimitiveValueNode>*** **activemq::util::PrimitiveValueNode::PrimitiveValue::listValue**

6.405.2.9 long long **activemq::util::PrimitiveValueNode::PrimitiveValue::longValue**

6.405.2.10 **decaf::util::Map<std::string, PrimitiveValueNode>*** **activemq::util::PrimitiveValueNode::PrimitiveValue::mapValue**

6.405.2.11 short **activemq::util::PrimitiveValueNode::PrimitiveValue::shortValue**

6.405.2.12 `std::string*` `activemq::util::PrimitiveValueNode::PrimitiveValue::stringValue`

The documentation for this union was generated from the following file:

- `src/main/activemq/util/PrimitiveValueNode.h`

6.406 `activemq::util::PrimitiveValueConverter` Class Reference

Class controls the conversion of data contained in a **PrimitiveValueNode** (p. 2145) from one type to another.

```
#include <src/main/activemq/util/PrimitiveValueConverter.h>
```

Public Member Functions

- **PrimitiveValueConverter** ()
- virtual **~PrimitiveValueConverter** ()
- `template<typename TO >`
TO convert (const **PrimitiveValueNode** &value) const

6.406.1 Detailed Description

Class controls the conversion of data contained in a **PrimitiveValueNode** (p. 2145) from one type to another.

If the conversion is supported then calling the convert method will throw an `UnsupportedOperationException` to indicate that its not possible to perform the conversion.

This class is used to implement the rules of conversion on CMS Message properties, the following conversion table must be implemented. A value written as the row type can be read in the column type.

		boolean	byte	short	int	long	float	double	String	-----				
-----		boolean	X X	byte	X X X X X	short	X X X X	int	X X X	long	X X	float	X X	
X		double	X X	String	X X X X X X X X		-----							

Since

3.0

6.406.2 Constructor & Destructor Documentation

6.406.2.1 `activemq::util::PrimitiveValueConverter::PrimitiveValueConverter ()` `[inline]`

6.406.2.2 `virtual activemq::util::PrimitiveValueConverter::~~PrimitiveValueConverter() [inline, virtual]`

6.406.3 Member Function Documentation

6.406.3.1 `std::vector< unsigned char > activemq::util::PrimitiveValueConverter::convert(const PrimitiveValueNode & value) const [inline]`

The documentation for this class was generated from the following file:

- `src/main/activemq/util/PrimitiveValueConverter.h`

6.407 activemq::util::PrimitiveValueNode Class Reference

Class that wraps around a single value of one of the many types.

```
#include <src/main/activemq/util/PrimitiveValueNode.h>
```

Data Structures

- union **PrimitiveValue**
Define a union type comprised of the various types.

Public Types

- enum **PrimitiveType** { **NULL_TYPE** = 0, **BOOLEAN_TYPE** = 1, **BYTE_TYPE** = 2, **CHAR_TYPE** = 3, **SHORT_TYPE** = 4, **INTEGER_TYPE** = 5, **LONG_TYPE** = 6, **DOUBLE_TYPE** = 7, **FLOAT_TYPE** = 8, **STRING_TYPE** = 9, **BYTE_ARRAY_TYPE** = 10, **MAP_TYPE** = 11, **LIST_TYPE** = 12, **BIG_STRING_TYPE** = 13 }
- Enumeration for the various primitive types.*

Public Member Functions

- **PrimitiveValueNode** ()
Default Constructor, creates a value of the NULL_TYPE.
- **PrimitiveValueNode** (bool value)
Boolean Value Constructor.
- **PrimitiveValueNode** (unsigned char value)
Byte Value Constructor.
- **PrimitiveValueNode** (char value)
Char Value Constructor.
- **PrimitiveValueNode** (short value)

Short Value Constructor.

- **PrimitiveValueNode** (int value)

Int Value Constructor.

- **PrimitiveValueNode** (long long value)

Long Value Constructor.

- **PrimitiveValueNode** (float value)

Float Value Constructor.

- **PrimitiveValueNode** (double value)

Double Value Constructor.

- **PrimitiveValueNode** (const char *value)

String Value Constructor.

- **PrimitiveValueNode** (const std::string &value)

String Value Constructor.

- **PrimitiveValueNode** (const std::vector< unsigned char > &value)

Byte Array Value Constructor.

- **PrimitiveValueNode** (const **decaf::util::List**< **PrimitiveValueNode** > &value)

Primitive List Constructor.

- **PrimitiveValueNode** (const **decaf::util::Map**< std::string, **PrimitiveValueNode** > &value)

Primitive Map Value Constructor.

- **PrimitiveValueNode** (const **PrimitiveValueNode** &node)

Copy constructor.

- **~PrimitiveValueNode** ()

- **PrimitiveValueNode & operator=** (const **PrimitiveValueNode** &node)

Assignment operator, copies the data from the other node.

- bool **operator==** (const **PrimitiveValueNode** &node) const

Comparison Operator, compares this node to the other node.

- **PrimitiveType** **getType** () const

Gets the Value Type of this type wrapper.

- **PrimitiveValue** **getValue** () const

Gets the internal Primitive Value object from this wrapper.

- void **setValue** (const **PrimitiveValue** &value, **PrimitiveType** valueType)

Sets the internal PrimitiveVale object to the new value along with the tag for the type that it consists of.

- void **clear** ()

Clears the value from this wrapper converting it back to a blank NULL_ TYPE value.

- void **setBool** (bool value)

Sets the value of this value node to the new value specified, this method overwrites any data that was previously at the index given.

- bool **getBool** () const

Gets the Boolean value of this Node.

- void **setByte** (unsigned char value)

Sets the value of this value node to the new value specified, this method overwrites any data that was previously at the index given.

- unsigned char **getByte** () const
Gets the Byte value of this Node.
- void **setChar** (char value)
Sets the value of this value node to the new value specified, this method overwrites any data that was previously at the index given.
- char **getChar** () const
Gets the Character value of this Node.
- void **setShort** (short value)
Sets the value of this value node to the new value specified, this method overwrites any data that was previously at the index given.
- short **getShort** () const
Gets the Short value of this Node.
- void **setInt** (int value)
Sets the value of this value node to the new value specified, this method overwrites any data that was previously at the index given.
- int **getInt** () const
Gets the Integer value of this Node.
- void **setLong** (long long value)
Sets the value of this value node to the new value specified, this method overwrites any data that was previously at the index given.
- long long **getLong** () const
Gets the Long value of this Node.
- void **setFloat** (float value)
Sets the value of this value node to the new value specified, this method overwrites any data that was previously at the index given.
- float **getFloat** () const
Gets the Float value of this Node.
- void **setDouble** (double value)
Sets the value of this value node to the new value specified, this method overwrites any data that was previously at the index given.
- double **getDouble** () const
Gets the Double value of this Node.
- void **setString** (const std::string &value)
Sets the value of this value node to the new value specified, this method overwrites any data that was previously at the index given.
- std::string **getString** () const
Gets the String value of this Node.
- void **setByteArray** (const std::vector< unsigned char > &value)
Sets the value of this value node to the new value specified, this method overwrites any data that was previously at the index given.
- std::vector< unsigned char > **getByteArray** () const
Gets the Byte Array value of this Node.
- void **setList** (const decaf::util::List< PrimitiveValueNode > &value)

Sets the value of this value node to the new value specified, this method overwrites any data that was previously at the index given.

- const **decaf::util::List** < **PrimitiveValueNode** > & **getList** () const

Gets the Primitive List value of this Node.

- void **setMap** (const **decaf::util::Map**< std::string, **PrimitiveValueNode** > &value)

Sets the value of this value node to the new value specified, this method overwrites any data that was previously at the index given.

- const **decaf::util::Map** < std::string, **PrimitiveValueNode** > & **getMap** () const

Gets the Primitive Map value of this Node.

- std::string **toString** () const

Creates a string representation of this value.

6.407.1 Detailed Description

Class that wraps around a single value of one of the many types.

Manages memory for complex types, such as strings. Note: the destructor was left non-virtual so no virtual table will be created. This probably isn't necessary, but will avoid needless memory allocation. Since we'll never extend this class, not having a virtual destructor isn't a concern.

6.407.2 Member Enumeration Documentation

6.407.2.1 enum activemq::util::PrimitiveValueNode::PrimitiveType

Enumeration for the various primitive types.

Enumerator:

NULL_TYPE
BOOLEAN_TYPE
BYTE_TYPE
CHAR_TYPE
SHORT_TYPE
INTEGER_TYPE
LONG_TYPE
DOUBLE_TYPE
FLOAT_TYPE
STRING_TYPE
BYTE_ARRAY_TYPE
MAP_TYPE
LIST_TYPE
BIG_STRING_TYPE

6.407.3 Constructor & Destructor Documentation

6.407.3.1 `activemq::util::PrimitiveValueNode::PrimitiveValueNode ()`

Default Constructor, creates a value of the NULL_TYPE.

6.407.3.2 `activemq::util::PrimitiveValueNode::PrimitiveValueNode (bool value)`

Boolean Value Constructor.

Parameters

<i>value</i>	- the new value to store.
--------------	---------------------------

6.407.3.3 `activemq::util::PrimitiveValueNode::PrimitiveValueNode (unsigned char value)`

Byte Value Constructor.

Parameters

<i>value</i>	- the new value to store.
--------------	---------------------------

6.407.3.4 `activemq::util::PrimitiveValueNode::PrimitiveValueNode (char value)`

Char Value Constructor.

Parameters

<i>value</i>	- the new value to store.
--------------	---------------------------

6.407.3.5 `activemq::util::PrimitiveValueNode::PrimitiveValueNode (short value)`

Short Value Constructor.

Parameters

<i>value</i>	- the new value to store.
--------------	---------------------------

6.407.3.6 `activemq::util::PrimitiveValueNode::PrimitiveValueNode (int value)`

Int Value Constructor.

Parameters

<i>value</i>	- the new value to store.
--------------	---------------------------

6.407.3.7 **activemq::util::PrimitiveValueNode::PrimitiveValueNode (long long *value*)**

Long Value Constructor.

Parameters

<i>value</i>	- the new value to store.
--------------	---------------------------

6.407.3.8 **activemq::util::PrimitiveValueNode::PrimitiveValueNode (float *value*)**

Float Value Constructor.

Parameters

<i>value</i>	- the new value to store.
--------------	---------------------------

6.407.3.9 **activemq::util::PrimitiveValueNode::PrimitiveValueNode (double *value*)**

Double Value Constructor.

Parameters

<i>value</i>	- the new value to store.
--------------	---------------------------

6.407.3.10 **activemq::util::PrimitiveValueNode::PrimitiveValueNode (const char * *value*)**

String Value Constructor.

Parameters

<i>value</i>	- the new value to store.
--------------	---------------------------

6.407.3.11 **activemq::util::PrimitiveValueNode::PrimitiveValueNode (const std::string & *value*)**

String Value Constructor.

Parameters

<i>value</i>	- the new value to store.
--------------	---------------------------

6.407.3.12 `activemq::util::PrimitiveValueNode::PrimitiveValueNode (const
std::vector< unsigned char > & value)`

Byte Array Value Constructor.

Parameters

<i>value</i>	- the new value to store.
--------------	---------------------------

6.407.3.13 `activemq::util::PrimitiveValueNode::PrimitiveValueNode (const
decaf::util::List< PrimitiveValueNode > & value)`

Primitive List Constructor.

Parameters

<i>value</i>	- the new value to store.
--------------	---------------------------

6.407.3.14 `activemq::util::PrimitiveValueNode::PrimitiveValueNode (const
decaf::util::Map< std::string, PrimitiveValueNode > & value)`

Primitive Map Value Constructor.

Parameters

<i>value</i>	- the new value to store.
--------------	---------------------------

6.407.3.15 `activemq::util::PrimitiveValueNode::PrimitiveValueNode (const
PrimitiveValueNode & node)`

Copy constructor.

Parameters

<i>node</i>	The instance of another node to copy to this one.
-------------	---

6.407.3.16 `activemq::util::PrimitiveValueNode::~~PrimitiveValueNode ()
[inline]`

6.407.4 Member Function Documentation

6.407.4.1 void activemq::util::PrimitiveValueNode::clear ()

Clears the value from this wrapper converting it back to a blank NULL_TYPE value.

6.407.4.2 bool activemq::util::PrimitiveValueNode::getBool () const

Gets the Boolean value of this Node.

Returns

value contained at the given index

Exceptions

<i>NoSuchElementException</i>	this node cannot be returned as the requested type.
-------------------------------	---

6.407.4.3 unsigned char activemq::util::PrimitiveValueNode::getByte () const

Gets the Byte value of this Node.

Returns

value contained at the given index

Exceptions

<i>NoSuchElementException</i>	this node cannot be returned as the requested type.
-------------------------------	---

6.407.4.4 std::vector<unsigned char> activemq::util::PrimitiveValueNode::getByteArray () const

Gets the Byte Array value of this Node.

Returns

value contained at the given index

Exceptions

<i>NoSuchElementException</i>	this node cannot be returned as the requested type.
-------------------------------	---

6.407.4.5 char activemq::util::PrimitiveValueNode::getChar () const

Gets the Character value of this Node.

Returns

value contained at the given index

Exceptions

<i>NoSuchElementException</i>	this node cannot be returned as the requested type.
-------------------------------	---

6.407.4.6 double activemq::util::PrimitiveValueNode::getDouble () const

Gets the Double value of this Node.

Returns

value contained at the given index

Exceptions

<i>NoSuchElementException</i>	this node cannot be returned as the requested type.
-------------------------------	---

6.407.4.7 float activemq::util::PrimitiveValueNode::getFloat () const

Gets the Float value of this Node.

Returns

value contained at the given index

Exceptions

<i>NoSuchElementException</i>	this node cannot be returned as the requested type.
-------------------------------	---

6.407.4.8 int activemq::util::PrimitiveValueNode::getInt () const

Gets the Integer value of this Node.

Returns

value contained at the given index

Exceptions

<i>NoSuchElementException</i>	this node cannot be returned as the requested type.
-------------------------------	---

6.407.4.9 `const decaf::util::List<PrimitiveValueNode>&
activemq::util::PrimitiveValueNode::getList () const`

Gets the Primitive List value of this Node.

Returns

value contained at the given index

Exceptions

<i>NoSuchElementException</i>	this node cannot be returned as the requested type.
-------------------------------	---

6.407.4.10 `long long activemq::util::PrimitiveValueNode::getLong () const`

Gets the Long value of this Node.

Returns

value contained at the given index

Exceptions

<i>NoSuchElementException</i>	this node cannot be returned as the requested type.
-------------------------------	---

6.407.4.11 `const decaf::util::Map<std::string, PrimitiveValueNode>&
activemq::util::PrimitiveValueNode::getMap () const`

Gets the Primitive Map value of this Node.

Returns

value contained at the given index

Exceptions

<i>NoSuchElementException</i>	this node cannot be returned as the requested type.
-------------------------------	---

6.407.4.12 short activemq::util::PrimitiveValueNode::getShort () const

Gets the Short value of this Node.

Returns

value contained at the given index

Exceptions

<i>NoSuchElementException</i>	this node cannot be returned as the requested type.
-------------------------------	---

6.407.4.13 std::string activemq::util::PrimitiveValueNode::getString () const

Gets the String value of this Node.

Returns

value contained at the given index

Exceptions

<i>NoSuchElementException</i>	this node cannot be returned as the requested type.
-------------------------------	---

6.407.4.14 PrimitiveType activemq::util::PrimitiveValueNode::getType () const
[inline]

Gets the Value Type of this type wrapper.

Returns

the PrimitiveType value for this wrapper.

6.407.4.15 PrimitiveValue activemq::util::PrimitiveValueNode::getValue () const
[inline]

Gets the internal Primitive Value object from this wrapper.

Returns

a copy of the contained **PrimitiveValue** (p. 2142)

6.407.4.16 `PrimitiveValueNode& activemq::util::PrimitiveValueNode::operator= (const PrimitiveValueNode & node)`

Assignment operator, copies the data from the other node.

Parameters

<i>node</i>	The instance of another node to copy to this one.
-------------	---

6.407.4.17 `bool activemq::util::PrimitiveValueNode::operator== (const PrimitiveValueNode & node) const`

Comparison Operator, compares this node to the other node.

Returns

true if the values are the same false otherwise.

6.407.4.18 `void activemq::util::PrimitiveValueNode::setBool (bool value)`

Sets the value of this value node to the new value specified, this method overwrites any data that was previously at the index given.

Parameters

<i>value</i>	- the new value to assign to the element at index
--------------	---

6.407.4.19 `void activemq::util::PrimitiveValueNode::setByte (unsigned char value)`

Sets the value of this value node to the new value specified, this method overwrites any data that was previously at the index given.

Parameters

<i>value</i>	- the new value to assign to the element at index
--------------	---

6.407.4.20 void **activemq::util::PrimitiveValueNode::setByteArray** (const
std::vector< unsigned char > & *value*)

Sets the value of this value node to the new value specified, this method overwrites any data that was previously at the index given.

Parameters

<i>value</i>	- the new value to assign to the element at index
--------------	---

6.407.4.21 void **activemq::util::PrimitiveValueNode::setChar** (char *value*)

Sets the value of this value node to the new value specified, this method overwrites any data that was previously at the index given.

Parameters

<i>value</i>	- the new value to assign to the element at index
--------------	---

6.407.4.22 void **activemq::util::PrimitiveValueNode::setDouble** (double *value*)

Sets the value of this value node to the new value specified, this method overwrites any data that was previously at the index given.

Parameters

<i>value</i>	- the new value to assign to the element at index
--------------	---

6.407.4.23 void **activemq::util::PrimitiveValueNode::setFloat** (float *value*)

Sets the value of this value node to the new value specified, this method overwrites any data that was previously at the index given.

Parameters

<i>value</i>	- the new value to assign to the element at index
--------------	---

6.407.4.24 void **activemq::util::PrimitiveValueNode::setInt** (int *value*)

Sets the value of this value node to the new value specified, this method overwrites any data that was previously at the index given.

Parameters

<i>value</i>	- the new value to assign to the element at index
--------------	---

6.407.4.25 `void activemq::util::PrimitiveValueNode::setList (const decaf::util::List< PrimitiveValueNode > & value)`

Sets the value of this value node to the new value specified, this method overwrites any data that was previously at the index given.

Parameters

<i>value</i>	- the new value to assign to the element at index
--------------	---

6.407.4.26 `void activemq::util::PrimitiveValueNode::setLong (long long value)`

Sets the value of this value node to the new value specified, this method overwrites any data that was previously at the index given.

Parameters

<i>value</i>	- the new value to assign to the element at index
--------------	---

6.407.4.27 `void activemq::util::PrimitiveValueNode::setMap (const decaf::util::Map< std::string, PrimitiveValueNode > & value)`

Sets the value of this value node to the new value specified, this method overwrites any data that was previously at the index given.

Parameters

<i>value</i>	- the new value to assign to the element at index
--------------	---

6.407.4.28 `void activemq::util::PrimitiveValueNode::setShort (short value)`

Sets the value of this value node to the new value specified, this method overwrites any data that was previously at the index given.

Parameters

<i>value</i>	- the new value to assign to the element at index
--------------	---

6.407.4.29 `void activemq::util::PrimitiveValueNode::setString (const std::string & value)`

Sets the value of this value node to the new value specified, this method overwrites any data that was previously at the index given.

Parameters

<i>value</i>	- the new value to assign to the element at index
--------------	---

6.407.4.30 void **activemq::util::PrimitiveValueNode::setValue** (const **PrimitiveValue** & *value*, **PrimitiveType** *valueType*)

Sets the internal PrimitiveVale object to the new value along with the tag for the type that it consists of.

Parameters

<i>value</i>	The value to set as the value contained in this Node.
<i>valueType</i>	The type of the value being set into this one.

6.407.4.31 std::string **activemq::util::PrimitiveValueNode::toString** () const

Creates a string representation of this value.

Returns

string value of this type wrapper.

The documentation for this class was generated from the following file:

- src/main/activemq/util/**PrimitiveValueNode.h**

6.408 decaf::security::Principal Class Reference

Base interface for a principal, which can represent an individual or organization.

```
#include <src/main/decaf/security/Principal.h>
```

Inheritance diagram for decaf::security::Principal:

Public Member Functions

- virtual ~**Principal** ()
- virtual bool **equals** (const **Principal** &another) const =0
Compares two principals to see if they are the same.
- virtual std::string **getName** () const =0
Provides the name of this principal.

6.408.1 Detailed Description

Base interface for a principal, which can represent an individual or organization.

6.408.2 Constructor & Destructor Documentation

6.408.2.1 `virtual decaf::security::Principal::~Principal () [inline, virtual]`

6.408.3 Member Function Documentation

6.408.3.1 `virtual bool decaf::security::Principal::equals (const Principal & another) const [pure virtual]`

Compares two principals to see if they are the same.

Parameters

<i>another</i>	A principal to be tested for equality to this one.
----------------	--

Returns

true if the given principal is equivalent to this one.

6.408.3.2 `virtual std::string decaf::security::Principal::getName () const [pure virtual]`

Provides the name of this principal.

Returns

the name of this principal.

Implemented in `decaf::security::auth::x500::X500Principal` (p. 2914).

The documentation for this class was generated from the following file:

- `src/main/decaf/security/Principal.h`

6.409 decaf::util::PriorityQueue< E > Class Template Reference

An unbounded priority queue based on a binary heap algorithm.

```
#include <src/main/decaf/util/PriorityQueue.h>
```

Inheritance diagram for `decaf::util::PriorityQueue< E >`:

Data Structures

- class **ConstPriorityQueueIterator**
- class **PriorityQueueIterator**

Public Member Functions

- **PriorityQueue** ()
*Creates a **Priority Queue** (p. 2222) with the default initial capacity.*
- **PriorityQueue** (int initialCapacity)
*Creates a **Priority Queue** (p. 2222) with the capacity value supplied.*
- **PriorityQueue** (int initialCapacity, **Comparator**< E > *comparator)
*Creates a **Priority Queue** (p. 2222) with the default initial capacity.*
- **PriorityQueue** (const **Collection**< E > &source)
*Creates a **PriorityQueue** (p. 2160) containing the elements in the specified **Collection** (p. 851).*
- **PriorityQueue** (const **PriorityQueue**< E > &source)
*Creates a **PriorityQueue** (p. 2160) containing the elements in the specified priority queue.*
- virtual **~PriorityQueue** ()
- **PriorityQueue**< E > & **operator=** (const **Collection**< E > &source)
*Assignment operator, assign another **Collection** (p. 851) to this one.*
- **PriorityQueue**< E > & **operator=** (const **PriorityQueue**< E > &source)
*Assignment operator, assign another **PriorityQueue** (p. 2160) to this one.*
- virtual **decaf::util::Iterator** < E > * **iterator** ()
- virtual **decaf::util::Iterator** < E > * **iterator** () const
- virtual int **size** () const
Returns the number of elements in this collection.
- virtual void **clear** ()
*Removes all of the elements from this collection (optional operation).
The collection will be empty after this method returns.
This implementation iterates over this collection, removing each element using the - **Iterator.remove** (p. 1560) operation. Most implementations will probably choose to override this method for efficiency.
Note that this implementation will throw an **UnsupportedOperationException** if the iterator returned by this collection's iterator method does not implement the remove method and this collection is non-empty.*

Exceptions

UnsupportedOperation-Exception	if the clear operation is not supported by this collection
--------------------------------	--

This implementation repeatedly invokes poll until it returns false.

- virtual bool **offer** (const E &value)
Inserts the specified element into the queue provided that the condition allows such an operation.
- virtual bool **poll** (E &result)

Gets and removes the element in the head of the queue.

- virtual bool **peek** (E &result) const

Gets but not removes the element in the head of the queue.

- virtual E **remove** ()

Gets and removes the element in the head of the queue.

*Throws a **NoSuchElementException** (p. 1984) if there is no element in the queue.*

Returns

the element in the head of the queue.

Exceptions

NoSuchElementException (p. 1984)	<i>if there is no element in the queue.</i>
--	---

This implementation returns the result of poll unless the queue is empty.

- virtual bool **remove** (const E &value)

Removes a single instance of the specified element from the collection.

More formally, removes an element e such that (value == NULL ? e == NULL : value == e), if this collection contains one or more such elements. Returns true if this collection contained the specified element (or equivalently, if this collection changed as a result of the call).

Parameters

value	<i>The reference to the element to remove from this Collection (p. 851).</i>
-------	---

Returns

true if the collection was changed, false otherwise.

Exceptions

UnsupportedOperationException	<i>if this is an unmodifiable collection.</i>
NullPointerException	<i>if the Collection (p. 851) is a container of pointers and does not allow NULL values.</i>

This implementation iterates over the collection looking for the specified element. - If it finds the element, it removes the element from the collection using the iterator's remove method.

Note that this implementation throws an UnsupportedOperationException if the iterator returned by this collection's iterator method does not implement the remove method and this collection contains the specified object.

- virtual bool **add** (const E &value)

Returns true if this collection changed as a result of the call.

(Returns false if this collection does not permit duplicates and already contains the specified element.)

*Collections that support this operation may place limitations on what elements may be added to this collection. In particular, some collections will refuse to add null elements, and others will impose restrictions on the type of elements that may be added. **Collection** (p. 851) classes should clearly specify in their documentation any restrictions on what elements may be added.*

If a collection refuses to add a particular element for any reason other than that it already contains the element, it must throw an exception (rather than returning false).

This preserves the invariant that a collection always contains the specified element

after this call returns.

For non-pointer values, i.e. class instances or string's the object will be copied into the collection, thus the object must support being copied, must not hide the copy constructor and assignment operator.

Parameters

value	The reference to the element to add to this Collection (p. 851).
-------	---

Returns

true if the element was added to this **Collection** (p. 851).

Exceptions

UnsupportedOperationException	if this is an unmodifiable collection.
NullPointerException	if the Collection (p. 851) is a container of pointers and does not allow NULL values.
IllegalArgumentException	if some property of the element prevents it from being added to this collection
IllegalStateException	if the element cannot be added at this time due to insertion restrictions.

This implementation returns true if offer succeeds, else throws an *IllegalStateException*.

- **decaf::lang::Pointer < Comparator< E > > comparator () const**

obtains a Copy of the Pointer instance that this **PriorityQueue** (p. 2160) is using to compare the elements in the queue with.

Friends

- class **PriorityQueueIterator**

6.409.1 Detailed Description

```
template<typename E>class decaf::util::PriorityQueue< E >
```

An unbounded priority queue based on a binary heap algorithm.

The elements of the priority queue are ordered according to their natural ordering, or by a **Comparator** (p. 888) provided to one of the constructors that accepts Comparators. A priority queue relying on natural ordering also does not permit insertion of non-comparable objects (doing so may result in a compiler error).

The head of this queue is the least element with respect to the specified ordering. - If multiple elements are tied for least value, the head is one of those elements -- ties are broken arbitrarily. The queue retrieval operations poll, remove, peek, and element access the element at the head of the queue.

A priority queue is unbounded, but has an internal capacity governing the size of an array used to store the elements on the queue. It is always at least as large as the queue size. As elements are added to a priority queue, its capacity grows automatically. The details of the growth policy are not specified.

This class and its iterator implement all of the optional methods of the **Collection** (p. 851) and **Iterator** (p. 1559) interfaces. The **Iterator** (p. 1559) provided in method **iterator()** (p. 2167) is not guaranteed to traverse the elements of the priority queue in any particular order. If you need ordered traversal, consider using `Arrays::sort(pq.toArray())`.

Note that this implementation is not synchronized. Multiple threads should not access a **PriorityQueue** (p. 2160) instance concurrently if any of the threads modifies the queue. Instead, use the thread-safe `PriorityBlockingQueue` class.

Implementation note: this implementation provides $O(\log(n))$ time for the enqueueing and dequeuing methods (`offer`, `poll`, **`remove()`** (p. 2169) and `add`); linear time for the `remove(-Object)` and `contains(Object)` methods; and constant time for the retrieval methods (`peek`, `element`, and `size`).

Since

1.0

6.409.2 Constructor & Destructor Documentation

6.409.2.1 `template<typename E> decaf::util::PriorityQueue< E >::PriorityQueue ()`
[inline]

Creates a **Priority Queue** (p. 2222) with the default initial capacity.

6.409.2.2 `template<typename E> decaf::util::PriorityQueue< E >::PriorityQueue (int
initialCapacity)` [inline]

Creates a **Priority Queue** (p. 2222) with the capacity value supplied.

Parameters

<i>initial-Capacity</i>	The initial number of elements allocated to this PriorityQueue (p. 2160).
-------------------------	--

6.409.2.3 `template<typename E> decaf::util::PriorityQueue< E >::PriorityQueue (int
initialCapacity, Comparator< E > * comparator)` [inline]

Creates a **Priority Queue** (p. 2222) with the default initial capacity.

This new **PriorityQueue** (p. 2160) takes ownership of the passed **Comparator** (p. 888) instance and uses that to determine the ordering of the elements in the **Queue** (p. 2222).

Parameters

<i>initial-Capacity</i>	The initial number of elements allocated to this PriorityQueue (p. 2160).
<i>comparator</i>	The Comparator (p. 888) instance to use in sorting the elements in the Queue (p. 2222).

Exceptions

<i>NullPointerException</i>	if the passed Comparator (p. 888) is NULL.
-----------------------------	---

6.409.2.4 `template<typename E> decaf::util::PriorityQueue< E >::PriorityQueue (const Collection< E > & source) [inline]`

Creates a **PriorityQueue** (p. 2160) containing the elements in the specified **Collection** (p. 851).

Parameters

<i>source</i>	the Collection (p. 851) whose elements are to be placed into this priority queue
---------------	---

6.409.2.5 `template<typename E> decaf::util::PriorityQueue< E >::PriorityQueue (const PriorityQueue< E > & source) [inline]`

Creates a **PriorityQueue** (p. 2160) containing the elements in the specified priority queue.

This priority queue will be ordered according to the same ordering as the given priority queue.

Parameters

<i>source</i>	the priority queue whose elements are to be placed into this priority queue
---------------	---

6.409.2.6 `template<typename E> virtual decaf::util::PriorityQueue< E >::~~PriorityQueue () [inline, virtual]`

6.409.3 Member Function Documentation

6.409.3.1 `template<typename E> virtual bool decaf::util::PriorityQueue< E >::add (const E & value) [inline, virtual]`

Returns true if this collection changed as a result of the call.

(Returns false if this collection does not permit duplicates and already contains the specified element.)

Collections that support this operation may place limitations on what elements may be added to this collection. In particular, some collections will refuse to add null elements, and others will impose restrictions on the type of elements that may be added. **Collection** (p. 851) classes should clearly specify in their documentation any restrictions on what elements may be added.

If a collection refuses to add a particular element for any reason other than that it already contains the element, it must throw an exception (rather than returning false). This preserves the invariant that a collection always contains the specified element after this call returns.

For non-pointer values, i.e. class instances or string's the object will be copied into the collection, thus the object must support being copied, must not hide the copy constructor and assignment operator.

Parameters

<i>value</i>	The reference to the element to add to this Collection (p. 851).
--------------	---

Returns

true if the element was added to this **Collection** (p. 851).

Exceptions

<i>UnsupportedOperationException</i>	if this is an unmodifiable collection.
<i>NullPointerException</i>	if the Collection (p. 851) is a container of pointers and does not allow NULL values.
<i>IllegalArgumentException</i>	if some property of the element prevents it from being added to this collection
<i>IllegalStateException</i>	if the element cannot be added at this time due to insertion restrictions.

This implementation returns true if offer succeeds, else throws an `IllegalStateException`.

This implementation returns true if offer succeeds, else throws an `IllegalStateException`.

Reimplemented from **decaf::util::AbstractQueue< E >** (p. 139).

References `DECAF_CATCH_EXCEPTION_CONVERT`, `DECAF_CATCH_RETHROW`, `DECAF_CATCHALL_THROW`, and `decaf::util::PriorityQueue< E >::offer()`.

6.409.3.2 `template<typename E> virtual void decaf::util::PriorityQueue< E >::clear ()`
`[inline, virtual]`

Removes all of the elements from this collection (optional operation).

The collection will be empty after this method returns.

This implementation iterates over this collection, removing each element using the **Iterator.remove** (p. 1560) operation. Most implementations will probably choose to override this method for efficiency.

Note that this implementation will throw an `UnsupportedOperationException` if the iterator returned by this collection's iterator method does not implement the remove method and this collection is non-empty.

Exceptions

<i>Unsupported- OperationException</i>	if the clear operation is not supported by this collection
--	--

This implementation repeatedly invokes poll until it returns false.

This implementation repeatedly invokes poll until it returns false.

Reimplemented from **decaf::util::AbstractQueue< E >** (p. 141).

6.409.3.3 `template<typename E> decaf::lang::Pointer< Comparator<E> >
decaf::util::PriorityQueue< E >::comparator () const [inline]`

obtains a Copy of the Pointer instance that this **PriorityQueue** (p.2160) is using to compare the elements in the queue with.

The returned value is a copy, the caller cannot change the value if the internal Pointer value.

Returns

a copy of the **Comparator** (p. 888) Pointer being used by this **Queue** (p. 2222).

6.409.3.4 `template<typename E> virtual decaf::util::Iterator<E>*
decaf::util::PriorityQueue< E >::iterator () [inline, virtual]`

Returns

an iterator over a set of elements of type T.

Implements **decaf::lang::Iterable< E >** (p. 1557).

References **decaf::util::PriorityQueue< E >::PriorityQueueIterator**.

6.409.3.5 `template<typename E> virtual decaf::util::Iterator<E>*
decaf::util::PriorityQueue< E >::iterator () const [inline,
virtual]`

Implements **decaf::lang::Iterable< E >** (p. 1558).

6.409.3.6 `template<typename E> virtual bool decaf::util::PriorityQueue< E >::offer (
const E & value) [inline, virtual]`

Inserts the specified element into the queue provided that the condition allows such an operation.

The method is generally preferable to the `collection.add(E)`, since the latter might throw an exception if the operation fails.

Parameters

<i>value</i>	the specified element to insert into the queue.
--------------	---

Returns

true if the operation succeeds and false if it fails.

Exceptions

<i>NullPointerException</i>	if the Queue (p. 2222) implementation does not allow Null values to be inserted into the Queue (p. 2222).
<i>IllegalArgumentException</i>	if some property of the specified element prevents it from being added to this queue

Implements **decaf::util::Queue< E >** (p. 2224).

Referenced by **decaf::util::PriorityQueue< E >::add()**.

```
6.409.3.7  template<typename E> PriorityQueue<E>& decaf::util::PriorityQueue< E
>::operator= ( const Collection< E > & source )  [inline]
```

Assignment operator, assign another **Collection** (p. 851) to this one.

Parameters

<i>source</i>	The Collection (p. 851) to copy to this one.
---------------	---

```
6.409.3.8  template<typename E> PriorityQueue<E>& decaf::util::PriorityQueue< E
>::operator= ( const PriorityQueue< E > & source )  [inline]
```

Assignment operator, assign another **PriorityQueue** (p. 2160) to this one.

Parameters

<i>source</i>	The PriorityQueue (p. 2160) to copy to this one.
---------------	---

```
6.409.3.9  template<typename E> virtual bool decaf::util::PriorityQueue< E >::peek ( E
& result ) const  [inline, virtual]
```

Gets but not removes the element in the head of the queue.

The result if successful is assigned to the result parameter.

Parameters

<i>result</i>	Reference to an instance of the contained type to assigned the removed value to.
---------------	--

Returns

true if the element at the head of the queue was removed and assigned to the result parameter.

Implements **decaf::util::Queue< E >** (p. 2225).

References **decaf::util::AbstractCollection< E >::isEmpty()**.

6.409.3.10 `template<typename E> virtual bool decaf::util::PriorityQueue< E >::poll (E & result) [inline, virtual]`

Gets and removes the element in the head of the queue.

If the operation succeeds the value of the element at the head of the **Queue** (p. 2222) is assigned to the result parameter and the method returns true. If the operation fails the method returns false and the value of the result parameter is undefined.

Parameters

<i>result</i>	Reference to an instance of the contained type to assigned the removed value to.
---------------	--

Returns

true if the element at the head of the queue was removed and assigned to the result parameter.

Implements **decaf::util::Queue< E >** (p. 2225).

References **decaf::util::AbstractCollection< E >::isEmpty()**.

6.409.3.11 `template<typename E> virtual E decaf::util::PriorityQueue< E >::remove () [inline, virtual]`

Gets and removes the element in the head of the queue.

Throws a **NoSuchElementException** (p. 1984) if there is no element in the queue.

Returns

the element in the head of the queue.

Exceptions

NoSuchElementException (p. 1984)	if there is no element in the queue.
--	--------------------------------------

This implementation returns the result of poll unless the queue is empty.

This implementation returns the result of poll unless the queue is empty.

Reimplemented from **decaf::util::AbstractQueue< E >** (p. 142).

References **decaf::util::AbstractCollection< E >::isEmpty()**.

6.409.3.12 `template<typename E> virtual bool decaf::util::PriorityQueue< E >::remove
(const E & value) [inline, virtual]`

Removes a single instance of the specified element from the collection.

More formally, removes an element *e* such that (*value* == NULL ? *e* == NULL : *value* == *e*), if this collection contains one or more such elements. Returns true if this collection contained the specified element (or equivalently, if this collection changed as a result of the call).

Parameters

<i>value</i>	The reference to the element to remove from this Collection (p. 851).
--------------	--

Returns

true if the collection was changed, false otherwise.

Exceptions

<i>UnsupportedOperationException</i>	if this is an unmodifiable collection.
<i>NullPointerException</i>	if the Collection (p. 851) is a container of pointers and does not allow NULL values.

This implementation iterates over the collection looking for the specified element. If it finds the element, it removes the element from the collection using the iterator's remove method.

Note that this implementation throws an `UnsupportedOperationException` if the iterator returned by this collection's iterator method does not implement the remove method and this collection contains the specified object.

This implementation iterates over the collection looking for the specified element. If it finds the element, it removes the element from the collection using the iterator's remove method.

Note that this implementation throws an `UnsupportedOperationException` if the iterator returned by this collection's iterator method does not implement the remove method and this collection contains the specified object.

Reimplemented from **decaf::util::AbstractCollection< E >** (p. 116).

6.409.3.13 `template<typename E> virtual int decaf::util::PriorityQueue< E >::size ()
const [inline, virtual]`

Returns the number of elements in this collection.

If this collection contains more than Integer::MAX_VALUE elements, returns Integer::MAX_VALUE.

Returns

the number of elements in this collection

Implements `decaf::util::Collection< E >` (p. 864).

6.409.4 Friends And Related Function Documentation

6.409.4.1 `template<typename E> friend class PriorityQueueIterator [friend]`

Referenced by `decaf::util::PriorityQueue< E >::iterator()`.

The documentation for this class was generated from the following file:

- `src/main/decaf/util/ProducerAck.h`

6.410 activemq::commands::ProducerAck Class Reference

```
#include <src/main/activemq/commands/ProducerAck.h>
```

Inheritance diagram for `activemq::commands::ProducerAck`:

Public Member Functions

- **ProducerAck** ()
- virtual **~ProducerAck** ()
- virtual unsigned char **getDataStructureType** () const
*Get the **DataStructure** (p. 1133) Type as defined in CommandTypes.h.*
- virtual **ProducerAck * cloneDataStructure** () const
Clone this object and return a new instance that the caller now owns, this will be an exact copy of this one.
- virtual void **copyDataStructure** (const **DataStructure** *src)
Copy the contents of the passed object into this objects members, overwriting any existing data.
- virtual std::string **toString** () const
*Returns a string containing the information for this **DataStructure** (p. 1133) such as its type and value of its elements.*

- virtual bool **equals** (const **DataSet** *value) const
*Compares the **DataSet** (p. 1133) passed in to this one, and returns if they are equivalent.*
- virtual const **Pointer** < **ProducerId** > & **getProducerId** () const
- virtual **Pointer**< **ProducerId** > & **getProducerId** ()
- virtual void **setProducerId** (const **Pointer**< **ProducerId** > &producerId)
- virtual int **getSize** () const
- virtual void **setSize** (int size)
- virtual bool **isProducerAck** () const
- virtual **Pointer**< **Command** > **visit** (activemq::state::CommandVisitor *visitor)
Allows a Visitor to visit this command and return a response to the command based on the command type being visited.

Static Public Attributes

- static const unsigned char **ID_PRODUCERACK** = 19

Protected Attributes

- **Pointer**< **ProducerId** > producerId
- int size

6.410.1 Constructor & Destructor Documentation

6.410.1.1 **activemq::commands::ProducerAck::ProducerAck** ()

6.410.1.2 **virtual activemq::commands::ProducerAck::~~ProducerAck** ()
[virtual]

6.410.2 Member Function Documentation

6.410.2.1 **virtual ProducerAck* activemq::commands::ProducerAck::cloneDataSet** () const [virtual]

Clone this object and return a new instance that the caller now owns, this will be an exact copy of this one.

Returns

new copy of this object.

Implements **activemq::commands::DataSet** (p. 1133).

6.410.2.2 `virtual void activemq::commands::ProducerAck::copyDataStructure (const DataStructure * src) [virtual]`

Copy the contents of the passed object into this objects members, overwriting any existing data.

Returns

src - Source Object

Reimplemented from **activemq::commands::BaseCommand** (p. 493).

6.410.2.3 `virtual bool activemq::commands::ProducerAck::equals (const DataStructure * value) const [virtual]`

Compares the **DataStructure** (p. 1133) passed in to this one, and returns if they are equivalent.

Equivalent here means that they are of the same type, and that each element of the objects are the same.

Returns

true if **DataStructure** (p. 1133)'s are Equal.

Reimplemented from **activemq::commands::BaseCommand** (p. 494).

6.410.2.4 `virtual unsigned char activemq::commands::ProducerAck::getDataStructureType () const [virtual]`

Get the **DataStructure** (p. 1133) Type as defined in CommandTypes.h.

Returns

The type of the data structure

Implements **activemq::commands::DataStructure** (p. 1137).

6.410.2.5 `virtual const Pointer<ProducerId>& activemq::commands::ProducerAck::getProducerId () const [virtual]`

6.410.2.6 `virtual Pointer<ProducerId>& activemq::commands::ProducerAck::getProducerId () [virtual]`

6.410.2.7 `virtual int activemq::commands::ProducerAck::getSize () const [virtual]`

6.410.2.8 `virtual bool activemq::commands::ProducerAck::isProducerAck () const [inline, virtual]`

Returns

an answer of true to the **isProducerAck()** (p. 2173) query.

Reimplemented from **activemq::commands::BaseCommand** (p. 496).

6.410.2.9 **virtual void activemq::commands::ProducerAck::setProducerId (const Pointer< ProducerId > & producerId)** [virtual]

6.410.2.10 **virtual void activemq::commands::ProducerAck::setSize (int size)** [virtual]

6.410.2.11 **virtual std::string activemq::commands::ProducerAck::toString () const** [virtual]

Returns a string containing the information for this **DataStructure** (p. 1133) such as its type and value of its elements.

Returns

formatted string useful for debugging.

Reimplemented from **activemq::commands::BaseCommand** (p. 498).

6.410.2.12 **virtual Pointer<Command> activemq::commands::ProducerAck::visit (activemq::state::CommandVisitor * visitor)** [virtual]

Allows a Visitor to visit this command and return a response to the command based on the command type being visited.

The command will call the proper processXXX method in the visitor.

Returns

a **Response** (p. 2298) to the visitor being called or NULL if no response.

Implements **activemq::commands::Command** (p. 871).

6.410.3 Field Documentation

6.410.3.1 **const unsigned char activemq::commands::ProducerAck::ID_PRODUCER-ACK = 19** [static]

6.410.3.2 **Pointer<ProducerId> activemq::commands::ProducerAck::producerId** [protected]

6.410.3.3 **int activemq::commands::ProducerAck::size** [protected]

The documentation for this class was generated from the following file:

6.411

activemq::wireformat::openwire::marshal::generated::ProducerAckMarshaller

Class Reference

2181

- src/main/activemq/commands/ProducerAck.h

6.411 activemq::wireformat::openwire::marshal::generated::- ProducerAckMarshaller Class Reference

Marshaling code for Open Wire Format for **ProducerAckMarshaller** (p. 2175).

```
#include <src/main/activemq/wireformat/openwire/marshal/generated/-  
ProducerAckMarshaller.h>
```

Inheritance diagram for activemq::wireformat::openwire::marshal::generated::ProducerAckMarshaller:

Public Member Functions

- **ProducerAckMarshaller** ()
- virtual ~**ProducerAckMarshaller** ()
- virtual **commands::DataStructure** * **createObject** () const
Creates a new instance of the class that this class is a marshaling director for.
- virtual unsigned char **getDataStructureType** () const
Gets the DataStructureType that this class marshals/unmarshals.
- virtual void **tightUnmarshal** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataInputStream** *dataIn, **utils::BooleanStream** *bs)
Tight Un-marhsal to the given stream.
- virtual int **tightMarshal1** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **utils::BooleanStream** *bs)
Tight Marhsal to the given stream.
- virtual void **tightMarshal2** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataOutputStream** *dataOut, **utils::BooleanStream** *bs)
Tight Marhsal to the given stream.
- virtual void **looseUnmarshal** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataInputStream** *dataIn)
Loose Un-marhsal to the given stream.
- virtual void **looseMarshal** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataOutputStream** *dataOut)
Tight Marhsal to the given stream.

6.411.1 Detailed Description

Marshaling code for Open Wire Format for **ProducerAckMarshaller** (p. 2175).

NOTE!: This file is auto generated - do not modify! if you need to make a change, please see the Java Classes in the activemq-openwire-generator module

6.411.2 Constructor & Destructor Documentation

6.411.2.1 **activemq::wireformat::openwire::marshal::generated-
::ProducerAckMarshaller::ProducerAckMarshaller ()**
[inline]

6.411.2.2 **virtual activemq::wireformat::openwire::marshal::generated:-
ProducerAckMarshaller::~~ProducerAckMarshaller ()** [inline,
virtual]

6.411.3 Member Function Documentation

6.411.3.1 **virtual commands::DataStructure* activemq::wireformat::openwire-
::marshal::generated::ProducerAckMarshaller::createObject ()** const
[virtual]

Creates a new instance of the class that this class is a marshaling director for.

Returns

newly allocated Command

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller**
(p. 1120).

6.411.3.2 **virtual unsigned char activemq::wireformat::openwire::marshal-
::generated::ProducerAckMarshaller::getDataStructureType ()** const
[virtual]

Gets the DataStructureType that this class marshals/unmarshals.

Returns

byte Id of this classes DataStructureType

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller**
(p. 1122).

6.411

activemq::wireformat::openwire::marshal::generated::ProducerAckMarshaller

Class Reference

2183

6.411.3.3 virtual void **activemq::wireformat::openwire::marshal::generated::ProducerAckMarshaller::looseMarshal** (**OpenWireFormat** * *format*,
commands::DataStructure * *command*, **decaf::io::DataOutputStream** * *ds*
) [virtual]

Tight Marhsal to the given stream.

Parameters

<i>format</i>	- The OpenWireFormat properties
<i>command</i>	- the object to Marshal
<i>ds</i>	- DataOutputStream to marshal to

Exceptions

<i>IOException</i>	if an error occurs.
--------------------	---------------------

Reimplemented from **activemq::wireformat::openwire::marshal::generated::Base-CommandMarshaller** (p. 500).

6.411.3.4 virtual void **activemq::wireformat::openwire::marshal::generated::ProducerAckMarshaller::looseUnmarshal** (**OpenWireFormat** * *format*,
commands::DataStructure * *command*, **decaf::io::DataInputStream** * *dis*)
[virtual]

Loose Un-marhsal to the given stream.

Parameters

<i>format</i>	- The OpenWireFormat properties
<i>command</i>	- the object to Un-Marshal
<i>dis</i>	- the DataInputStream to Un-Marshal from

Exceptions

<i>IOException</i>	if an error occurs.
--------------------	---------------------

Reimplemented from **activemq::wireformat::openwire::marshal::generated::Base-CommandMarshaller** (p. 501).

6.411.3.5 virtual int **activemq::wireformat::openwire::marshal::generated::ProducerAckMarshaller::tightMarshal1** (**OpenWireFormat** * *format*,
commands::DataStructure * *command*, **utils::BooleanStream** * *bs*)
[virtual]

Tight Marhsal to the given stream.

Parameters

<i>format</i>	The OpenwireFormat properties
<i>command</i>	The object to Marshal
<i>bs</i>	The boolean stream to marshal to.

Exceptions

<i>IOException</i>	if an error occurs.
--------------------	---------------------

Reimplemented from **activemq::wireformat::openwire::marshal::generated::BaseCommandMarshaller** (p. 503).

6.411.3.6 `virtual void activemq::wireformat::openwire::marshal::generated::ProducerAckMarshaller::tightMarshal2 (OpenWireFormat * format, commands::DataStructure * command, decaf::io::DataOutputStream * ds, utils::BooleanStream * bs) [virtual]`

Tight Marshal to the given stream.

Parameters

<i>format</i>	- The OpenwireFormat properties
<i>command</i>	- the object to Marshal
<i>ds</i>	- the DataOutputStream to Marshal to
<i>bs</i>	- boolean stream to marshal to.

Exceptions

<i>IOException</i>	if an error occurs.
--------------------	---------------------

Reimplemented from **activemq::wireformat::openwire::marshal::generated::BaseCommandMarshaller** (p. 504).

6.411.3.7 `virtual void activemq::wireformat::openwire::marshal::generated::ProducerAckMarshaller::tightUnmarshal (OpenWireFormat * format, commands::DataStructure * command, decaf::io::DataInputStream * dis, utils::BooleanStream * bs) [virtual]`

Tight Un-marshal to the given stream.

Parameters

<i>format</i>	- The OpenwireFormat properties
<i>command</i>	- the object to Un-Marshal
<i>dis</i>	- the DataInputStream to Un-Marshal from
<i>bs</i>	- boolean stream to unmarshal from.

Exceptions

<i>IOException</i>	if an error occurs.
--------------------	---------------------

Reimplemented from **activemq::wireformat::openwire::marshal::generated::Base-CommandMarshaller** (p. 505).

The documentation for this class was generated from the following file:

- src/main/activemq/wireformat/openwire/marshal/generated/**ProducerAck-Marshaller.h**

6.412 activemq::cmsutil::ProducerCallback Class Reference

Callback for sending a message to a CMS destination.

```
#include <src/main/activemq/cmsutil/ProducerCallback.h>
```

Inheritance diagram for activemq::cmsutil::ProducerCallback:

Public Member Functions

- virtual **~ProducerCallback** () throw ()
- virtual void **doInCms** (**cms::Session** *session, **cms::MessageProducer** *producer)=0

Execute an action given a session and producer.

6.412.1 Detailed Description

Callback for sending a message to a CMS destination.

6.412.2 Constructor & Destructor Documentation

6.412.2.1 virtual **activemq::cmsutil::ProducerCallback::~ProducerCallback** ()
throw () [inline, virtual]

6.412.3 Member Function Documentation

6.412.3.1 virtual void **activemq::cmsutil::ProducerCallback::doInCms** (
cms::Session * session, **cms::MessageProducer** * producer) [pure
virtual]

Execute an action given a session and producer.

Parameters

<i>session</i>	the CMS <code>Session</code>
<i>producer</i>	the CMS <code>Producer</code>

Exceptions

<code>cms::CMS-Exception</code> (p. 826)	if thrown by CMS API methods
--	------------------------------

Implemented in **`activemq::cmsutil::CmsTemplate::SendExecutor`** (p. 2342).

The documentation for this class was generated from the following file:

- `src/main/activemq/cmsutil/ProducerCallback.h`

6.413 **`activemq::cmsutil::CmsTemplate::ProducerExecutor` Class - Reference**

```
#include <src/main/activemq/cmsutil/CmsTemplate.h>
```

Inheritance diagram for `activemq::cmsutil::CmsTemplate::ProducerExecutor`:

Public Member Functions

- **`ProducerExecutor`** (`ProducerCallback *action`, `CmsTemplate *parent`, `cms::Destination *destination`)
- virtual **`~ProducerExecutor`** () throw ()
- virtual void **`doInCms`** (`cms::Session *session`)
Execute any number of operations against the supplied CMS session.
- virtual **`cms::Destination * getDestination`** (`cms::Session *session` AMQCPP-UNUSED)

Protected Attributes

- **`ProducerCallback * action`**
- **`CmsTemplate * parent`**
- **`cms::Destination * destination`**

6.413.1 Constructor & Destructor Documentation

6.413 activemq::cmsutil::CmsTemplate::ProducerExecutor Class Reference 2187

6.413.1.1 `activemq::cmsutil::CmsTemplate::ProducerExecutor::ProducerExecutor (ProducerCallback * action, CmsTemplate * parent, cms::Destination * destination)` `[inline]`

6.413.1.2 `virtual activemq::cmsutil::CmsTemplate::Producer-Executor::~~ProducerExecutor () throw ()` `[inline, virtual]`

6.413.2 Member Function Documentation

6.413.2.1 `virtual void activemq::cmsutil::CmsTemplate::Producer-Executor::doInCms (cms::Session * session)` `[virtual]`

Execute any number of operations against the supplied CMS session.

Parameters

<i>session</i>	the CMS Session
----------------	-----------------

Exceptions

<i>CMSException</i>	if thrown by CMS API methods
---------------------	------------------------------

Implements `activemq::cmsutil::SessionCallback` (p. 2377).

6.413.2.2 `virtual cms::Destination* activemq::cmsutil::CmsTemplate::Producer-Executor::getDestination (cms::Session *session AMQCPP_UNUSED)` `[inline, virtual]`

6.413.3 Field Documentation

6.413.3.1 `ProducerCallback* activemq::cmsutil::CmsTemplate::Producer-Executor::action` `[protected]`

6.413.3.2 `cms::Destination* activemq::cmsutil::CmsTemplate::Producer-Executor::destination` `[protected]`

6.413.3.3 `CmsTemplate* activemq::cmsutil::CmsTemplate::ProducerExecutor::parent` `[protected]`

The documentation for this class was generated from the following file:

- `src/main/activemq/cmsutil/CmsTemplate.h`

6.414 activemq::commands::ProducerId Class Reference

```
#include <src/main/activemq/commands/ProducerId.h>
```

Inheritance diagram for activemq::commands::ProducerId:

Public Types

- typedef **decaf::lang::PointerComparator** < **ProducerId** > **COMPARATOR**

Public Member Functions

- **ProducerId** ()
- **ProducerId** (const **ProducerId** &other)
- **ProducerId** (const **SessionId** &sessionId, long long consumerId)
- **ProducerId** (std::string producerId)
- virtual ~**ProducerId** ()
- virtual unsigned char **getDataStructureType** () const
*Get the **DataStructure** (p. 1133) Type as defined in CommandTypes.h.*
- virtual **ProducerId** * **cloneDataStructure** () const
Clone this object and return a new instance that the caller now owns, this will be an exact copy of this one.
- virtual void **copyDataStructure** (const **DataStructure** *src)
Copy the contents of the passed object into this objects members, overwriting any existing data.
- virtual std::string **toString** () const
*Returns a string containing the information for this **DataStructure** (p. 1133) such as its type and value of its elements.*
- virtual bool **equals** (const **DataStructure** *value) const
*Compares the **DataStructure** (p. 1133) passed in to this one, and returns if they are equivalent.*
- const **Pointer**< **SessionId** > & **getParentId** () const
- void **setProducerSessionKey** (std::string sessionKey)
- virtual const std::string & **getConnectionId** () const
- virtual std::string & **getConnectionId** ()
- virtual void **setConnectionId** (const std::string &connectionId)
- virtual long long **getValue** () const
- virtual void **setValue** (long long value)
- virtual long long **getSessionId** () const
- virtual void **setSessionId** (long long sessionId)
- virtual int **compareTo** (const **ProducerId** &value) const
- virtual bool **equals** (const **ProducerId** &value) const
- virtual bool **operator==** (const **ProducerId** &value) const
- virtual bool **operator<** (const **ProducerId** &value) const
- **ProducerId** & **operator=** (const **ProducerId** &other)

Static Public Attributes

- static const unsigned char **ID_PRODUCERID** = 123

Protected Attributes

- std::string **connectionId**
- long long **value**
- long long **sessionId**

6.414.1 Member Typedef Documentation

6.414.1.1 `typedef decaf::lang::PointerComparator<ProducerId>
activemq::commands::ProducerId::COMPARATOR`

6.414.2 Constructor & Destructor Documentation

6.414.2.1 `activemq::commands::ProducerId::ProducerId ()`

6.414.2.2 `activemq::commands::ProducerId::ProducerId (const ProducerId & other
)`

6.414.2.3 `activemq::commands::ProducerId::ProducerId (const SessionId &
sessionId, long long consumerId)`

6.414.2.4 `activemq::commands::ProducerId::ProducerId (std::string producerId)`

6.414.2.5 `virtual activemq::commands::ProducerId::~~ProducerId ()
[virtual]`

6.414.3 Member Function Documentation

6.414.3.1 `virtual ProducerId* activemq::commands::ProducerId::cloneData-
Structure () const [virtual]`

Clone this object and return a new instance that the caller now owns, this will be an exact copy of this one.

Returns

new copy of this object.

Implements `activemq::commands::DataStructure` (p. 1133).

6.414.3.2 `virtual int activemq::commands::ProducerId::compareTo (const
ProducerId & value) const [virtual]`

6.414.3.3 `virtual void activemq::commands::ProducerId::copyDataStructure (const DataStructure * src) [virtual]`

Copy the contents of the passed object into this objects members, overwriting any existing data.

Returns

src - Source Object

Implements `activemq::commands::DataStructure` (p. 1134).

6.414.3.4 `virtual bool activemq::commands::ProducerId::equals (const DataStructure * value) const [virtual]`

Compares the `DataStructure` (p. 1133) passed in to this one, and returns if they are equivalent.

Equivalent here means that they are of the same type, and that each element of the objects are the same.

Returns

true if `DataStructure` (p. 1133)'s are Equal.

Implements `activemq::commands::DataStructure` (p. 1135).

6.414.3.5 `virtual bool activemq::commands::ProducerId::equals (const ProducerId & value) const [virtual]`

6.414.3.6 `virtual const std::string& activemq::commands::ProducerId::getConnectionId () const [virtual]`

6.414.3.7 `virtual std::string& activemq::commands::ProducerId::getConnectionId () [virtual]`

6.414.3.8 `virtual unsigned char activemq::commands::ProducerId::getDataStructureType () const [virtual]`

Get the `DataStructure` (p. 1133) Type as defined in `CommandTypes.h`.

Returns

The type of the data structure

Implements `activemq::commands::DataStructure` (p. 1137).

- 6.414.3.9 `const Pointer<SessionId>& activemq::commands::ProducerId::getParentId () const`
- 6.414.3.10 `virtual long long activemq::commands::ProducerId::getSessionId () const [virtual]`
- 6.414.3.11 `virtual long long activemq::commands::ProducerId::getValue () const [virtual]`
- 6.414.3.12 `virtual bool activemq::commands::ProducerId::operator< (const ProducerId & value) const [virtual]`
- 6.414.3.13 `ProducerId& activemq::commands::ProducerId::operator= (const ProducerId & other)`
- 6.414.3.14 `virtual bool activemq::commands::ProducerId::operator== (const ProducerId & value) const [virtual]`
- 6.414.3.15 `virtual void activemq::commands::ProducerId::setConnectionId (const std::string & connectionId) [virtual]`
- 6.414.3.16 `void activemq::commands::ProducerId::setProducerSessionKey (std::string sessionKey)`
- 6.414.3.17 `virtual void activemq::commands::ProducerId::setSessionId (long long sessionId) [virtual]`
- 6.414.3.18 `virtual void activemq::commands::ProducerId::setValue (long long value) [virtual]`
- 6.414.3.19 `virtual std::string activemq::commands::ProducerId::toString () const [virtual]`

Returns a string containing the information for this **DataStructure** (p. 1133) such as its type and value of its elements.

Returns

formatted string useful for debugging.

Reimplemented from **activemq::commands::BaseDataStructure** (p. 531).

6.414.4 Field Documentation

- 6.414.4.1 `std::string activemq::commands::ProducerId::connectionId [protected]`

6.414.4.2 `const unsigned char activemq::commands::ProducerId::ID_PRODUCERID = 123 [static]`

Referenced by `activemq::state::CommandVisitorAdapter::processRemoveInfo()`.

6.414.4.3 `long long activemq::commands::ProducerId::sessionId [protected]`

6.414.4.4 `long long activemq::commands::ProducerId::value [protected]`

The documentation for this class was generated from the following file:

- `src/main/activemq/commands/ProducerId.h`

6.415 `activemq::wireformat::openwire::marshal::generated::-` `ProducerIdMarshaller` Class Reference

Marshaling code for Open Wire Format for `ProducerIdMarshaller` (p. 2186).

```
#include <src/main/activemq/wireformat/openwire/marshal/generated/-
ProducerIdMarshaller.h>
```

Inheritance diagram for `activemq::wireformat::openwire::marshal::generated::-`
`ProducerIdMarshaller`:

Public Member Functions

- `ProducerIdMarshaller ()`
- `virtual ~ProducerIdMarshaller ()`
- `virtual commands::DataStructure * createObject () const`
Creates a new instance of the class that this class is a marshaling director for.
- `virtual unsigned char getDataStructureType () const`
Gets the DataStructureType that this class marshals/unmarshals.
- `virtual void tightUnmarshal (OpenWireFormat *wireFormat, commands::DataStructure *dataStructure, decaf::io::DataInputStream *dataIn, utils::BooleanStream *bs)`
Tight Un-marhsal to the given stream.
- `virtual int tightMarshal1 (OpenWireFormat *wireFormat, commands::DataStructure *dataStructure, utils::BooleanStream *bs)`
Tight Marhsal to the given stream.
- `virtual void tightMarshal2 (OpenWireFormat *wireFormat, commands::DataStructure *dataStructure, decaf::io::DataOutputStream *dataOut, utils::BooleanStream *bs)`
Tight Marhsal to the given stream.

6.415

activemq::wireformat::openwire::marshal::generated::ProducerIdMarshaller

Class Reference

2193

- virtual void **looseUnmarshal** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataInputStream** *dataIn)

Loose Un-marhsal to the given stream.

- virtual void **looseMarshal** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataOutputStream** *dataOut)

Tight Marhsal to the given stream.

6.415.1 Detailed Description

Marshaling code for Open Wire Format for **ProducerIdMarshaller** (p. 2186).

NOTE!: This file is auto generated - do not modify! if you need to make a change, please see the Java Classes in the activemq-openwire-generator module

6.415.2 Constructor & Destructor Documentation

6.415.2.1 **activemq::wireformat::openwire::marshal::generated::ProducerIdMarshaller::ProducerIdMarshaller** ()
[inline]

6.415.2.2 virtual **activemq::wireformat::openwire::marshal::generated::ProducerIdMarshaller::~~ProducerIdMarshaller** () [inline, virtual]

6.415.3 Member Function Documentation

6.415.3.1 virtual **commands::DataStructure*** **activemq::wireformat::openwire::marshal::generated::ProducerIdMarshaller::createObject** () const
[virtual]

Creates a new instance of the class that this class is a marshaling director for.

Returns

newly allocated Command

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1120).

6.415.3.2 virtual unsigned char **activemq::wireformat::openwire::marshal::generated::ProducerIdMarshaller::getDataStructureType** () const
[virtual]

Gets the DataStructureType that this class marshals/unmarshals.

Returns

byte Id of this classes DataStructureType

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller**
(p. 1122).

6.415.3.3 **virtual void activemq::wireformat::openwire::marshal::generated::-
ProducerIdMarshaller::looseMarshal (OpenWireFormat * *format*,
commands::DataStructure * *command*, decaf::io::DataOutputStream * *ds*
) [virtual]**

Tight Marshal to the given stream.

Parameters

<i>format</i>	- The OpenWireFormat properties
<i>command</i>	- the object to Marshal
<i>ds</i>	- DataOutputStream to marshal to

Exceptions

<i>IOException</i>	if an error occurs.
--------------------	---------------------

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller**
(p. 1123).

6.415.3.4 **virtual void activemq::wireformat::openwire::marshal::generated::-
ProducerIdMarshaller::looseUnmarshal (OpenWireFormat * *format*,
commands::DataStructure * *command*, decaf::io::DataInputStream * *dis*)
[virtual]**

Loose Un-marshal to the given stream.

Parameters

<i>format</i>	- The OpenWireFormat properties
<i>command</i>	- the object to Un-Marshal
<i>dis</i>	- the DataInputStream to Un-Marshal from

Exceptions

<i>IOException</i>	if an error occurs.
--------------------	---------------------

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller**
(p. 1125).

6.415

activemq::wireformat::openwire::marshal::generated::ProducerIdMarshaller

Class Reference

2195

6.415.3.5 virtual int **activemq::wireformat::openwire::marshal::generated::ProducerIdMarshaller::tightMarshal1** (OpenWireFormat * *format*, commands::DataStructure * *command*, utils::BooleanStream * *bs*)
[virtual]

Tight Marhsal to the given stream.

Parameters

<i>format</i>	The OpenwireFormat properties
<i>command</i>	The object to Marshal
<i>bs</i>	The boolean stream to marshal to.

Exceptions

<i>IOException</i>	if an error occurs.
--------------------	---------------------

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1127).

6.415.3.6 virtual void **activemq::wireformat::openwire::marshal::generated::ProducerIdMarshaller::tightMarshal2** (OpenWireFormat * *format*, commands::DataStructure * *command*, decaf::io::DataOutputStream * *ds*, utils::BooleanStream * *bs*) [virtual]

Tight Marhsal to the given stream.

Parameters

<i>format</i>	- The OpenwireFormat properties
<i>command</i>	- the object to Marshal
<i>ds</i>	- the DataOutputStream to Marshal to
<i>bs</i>	- boolean stream to marshal to.

Exceptions

<i>IOException</i>	if an error occurs.
--------------------	---------------------

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1129).

6.415.3.7 virtual void **activemq::wireformat::openwire::marshal::generated::ProducerIdMarshaller::tightUnmarshal** (OpenWireFormat * *format*, commands::DataStructure * *command*, decaf::io::DataInputStream * *dis*, utils::BooleanStream * *bs*) [virtual]

Tight Un-marhsal to the given stream.

Parameters

<i>format</i>	- The OpenwireFormat properties
<i>command</i>	- the object to Un-Marshal
<i>dis</i>	- the DataInputStream to Un-Marshal from
<i>bs</i>	- boolean stream to unmarshal from.

Exceptions

<i>IOException</i>	if an error occurs.
--------------------	---------------------

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1131).

The documentation for this class was generated from the following file:

- src/main/activemq/wireformat/openwire/marshal/generated/**ProducerIdMarshaller.h**

6.416 activemq::commands::ProducerInfo Class Reference

```
#include <src/main/activemq/commands/ProducerInfo.h>
```

Inheritance diagram for activemq::commands::ProducerInfo:

Public Member Functions

- **ProducerInfo** ()
- virtual **~ProducerInfo** ()
- virtual unsigned char **getDataStructureType** () const
*Get the **DataStructure** (p. 1133) Type as defined in CommandTypes.h.*
- virtual **ProducerInfo * cloneDataStructure** () const
Clone this obbect and return a new instance that the caller now owns, this will be an exact copy of this one.
- virtual void **copyDataStructure** (const **DataStructure** *src)
Copy the contents of the passed object into this objects members, overwriting any existing data.
- virtual std::string **toString** () const
*Returns a string containing the information for this **DataStructure** (p. 1133) such as its type and value of its elements.*
- virtual bool **equals** (const **DataStructure** *value) const
*Compares the **DataStructure** (p. 1133) passed in to this one, and returns if they are equivalent.*
- **Pointer< RemoveInfo > createRemoveCommand** () const
- virtual const **Pointer < ProducerId > & getProducerId** () const

- virtual **Pointer**< **ProducerId** > & **getProducerId** ()
- virtual void **setProducerId** (const **Pointer**< **ProducerId** > &producerId)
- virtual const **Pointer** < **ActiveMQDestination** > & **getDestination** () const
- virtual **Pointer** < **ActiveMQDestination** > & **getDestination** ()
- virtual void **setDestination** (const **Pointer**< **ActiveMQDestination** > &destination)
- virtual const std::vector < **decaf::lang::Pointer** < **BrokerId** > > & **getBrokerPath** () const
- virtual std::vector < **decaf::lang::Pointer** < **BrokerId** > > & **getBrokerPath** ()
- virtual void **setBrokerPath** (const std::vector< **decaf::lang::Pointer**< **BrokerId** > > &brokerPath)
- virtual bool **isDispatchAsync** () const
- virtual void **setDispatchAsync** (bool dispatchAsync)
- virtual int **getWindowSize** () const
- virtual void **setWindowSize** (int windowSize)
- virtual bool **isProducerInfo** () const
- virtual **Pointer**< **Command** > **visit** (activemq::state::CommandVisitor *visitor)

Allows a Visitor to visit this command and return a response to the command based on the command type being visited.

Static Public Attributes

- static const unsigned char **ID_PRODUCERINFO** = 6

Protected Attributes

- **Pointer**< **ProducerId** > **producerId**
- **Pointer**< **ActiveMQDestination** > **destination**
- std::vector < **decaf::lang::Pointer** < **BrokerId** > > **brokerPath**
- bool **dispatchAsync**
- int **windowSize**

6.416.1 Constructor & Destructor Documentation

6.416.1.1 **activemq::commands::ProducerInfo::ProducerInfo** ()

6.416.1.2 **virtual activemq::commands::ProducerInfo::~~ProducerInfo** ()
[virtual]

6.416.2 Member Function Documentation

6.416.2.1 **virtual ProducerInfo* activemq::commands::ProducerInfo::cloneDataStructure** () const [virtual]

Clone this object and return a new instance that the caller now owns, this will be an exact copy of this one.

Returns

new copy of this object.

Implements **activemq::commands::DataStructure** (p. 1133).

6.416.2.2 **virtual void activemq::commands::ProducerInfo::copyDataStructure (const DataStructure * src) [virtual]**

Copy the contents of the passed object into this objects members, overwriting any existing data.

Returns

src - Source Object

Reimplemented from **activemq::commands::BaseCommand** (p. 493).

6.416.2.3 **Pointer<RemoveInfo> activemq::commands::ProducerInfo::createRemoveCommand () const**

6.416.2.4 **virtual bool activemq::commands::ProducerInfo::equals (const DataStructure * value) const [virtual]**

Compares the **DataStructure** (p. 1133) passed in to this one, and returns if they are equivalent.

Equivalent here means that they are of the same type, and that each element of the objects are the same.

Returns

true if **DataStructure** (p. 1133)'s are Equal.

Reimplemented from **activemq::commands::BaseCommand** (p. 494).

6.416.2.5 **virtual const std::vector< decaf::lang::Pointer<BrokerId> >& activemq::commands::ProducerInfo::getBrokerPath () const [virtual]**

6.416.2.6 **virtual std::vector< decaf::lang::Pointer<BrokerId> >& activemq::commands::ProducerInfo::getBrokerPath () [virtual]**

6.416.2.7 **virtual unsigned char activemq::commands::ProducerInfo::getDataStructureType () const [virtual]**

Get the **DataStructure** (p. 1133) Type as defined in CommandTypes.h.

Returns

The type of the data structure

Implements **activemq::commands::DataStructure** (p. 1137).

6.416.2.8 **virtual const Pointer<ActiveMQDestination>& activemq-
::commands::ProducerInfo::getDestination () const**
[virtual]

6.416.2.9 **virtual Pointer<ActiveMQDestination>& activemq-
::commands::ProducerInfo::getDestination ()**
[virtual]

6.416.2.10 **virtual const Pointer<ProducerId>& activemq-
::commands::ProducerInfo::getProducerId () const**
[virtual]

6.416.2.11 **virtual Pointer<ProducerId>& activemq::commands::ProducerInfo-
::getProducerId ()** [virtual]

6.416.2.12 **virtual int activemq::commands::ProducerInfo::getWindowSize () const**
[virtual]

6.416.2.13 **virtual bool activemq::commands::ProducerInfo::isDispatchAsync ()**
const [virtual]

6.416.2.14 **virtual bool activemq::commands::ProducerInfo::isProducerInfo ()**
const [inline, virtual]

Returns

an answer of true to the **isProducerInfo()** (p. 2193) query.

Reimplemented from **activemq::commands::BaseCommand** (p. 496).

6.416.2.15 **virtual void activemq::commands::ProducerInfo::setBrokerPath (**
const std::vector< decaf::lang::Pointer< BrokerId > > & brokerPath)
[virtual]

6.416.2.16 **virtual void activemq::commands::ProducerInfo::setDestination (const**
Pointer< ActiveMQDestination > & destination) [virtual]

6.416.2.17 **virtual void activemq::commands::ProducerInfo::setDispatchAsync (**
bool dispatchAsync) [virtual]

6.416.2.18 **virtual void activemq::commands::ProducerInfo::setProducerId (const**
Pointer< ProducerId > & producerId) [virtual]

6.416.2.19 `virtual void activemq::commands::ProducerInfo::setWindowSize (int windowSize) [virtual]`

6.416.2.20 `virtual std::string activemq::commands::ProducerInfo::toString () const [virtual]`

Returns a string containing the information for this **DataStructure** (p. 1133) such as its type and value of its elements.

Returns

formatted string useful for debugging.

Reimplemented from **activemq::commands::BaseCommand** (p. 498).

6.416.2.21 `virtual Pointer<Command> activemq::commands::ProducerInfo::visit (activemq::state::CommandVisitor * visitor) [virtual]`

Allows a Visitor to visit this command and return a response to the command based on the command type being visited.

The command will call the proper processXXX method in the visitor.

Returns

a **Response** (p. 2298) to the visitor being called or NULL if no response.

Implements **activemq::commands::Command** (p. 871).

6.416.3 Field Documentation

6.416.3.1 `std::vector< decaf::lang::Pointer<BrokerId> > activemq::commands::ProducerInfo::brokerPath [protected]`

6.416.3.2 `Pointer<ActiveMQDestination> activemq::commands::ProducerInfo::destination [protected]`

6.416.3.3 `bool activemq::commands::ProducerInfo::dispatchAsync [protected]`

6.416.3.4 `const unsigned char activemq::commands::ProducerInfo::ID_PRODUCERINFO = 6 [static]`

6.416.3.5 `Pointer<ProducerId> activemq::commands::ProducerInfo::producerId [protected]`

6.416.3.6 `int activemq::commands::ProducerInfo::windowSize [protected]`

The documentation for this class was generated from the following file:

6.417

activemq::wireformat::openwire::marshal::generated::ProducerInfoMarshaller

Class Reference

2201

- src/main/activemq/commands/ProducerInfo.h

6.417 activemq::wireformat::openwire::marshal::generated::- ProducerInfoMarshaller Class Reference

Marshaling code for Open Wire Format for **ProducerInfoMarshaller** (p. 2195).

```
#include <src/main/activemq/wireformat/openwire/marshal/generated/-  
ProducerInfoMarshaller.h>
```

Inheritance diagram for activemq::wireformat::openwire::marshal::generated::Producer-
InfoMarshaller:

Public Member Functions

- **ProducerInfoMarshaller** ()
- virtual ~**ProducerInfoMarshaller** ()
- virtual **commands::DataStructure** * **createObject** () const
Creates a new instance of the class that this class is a marshaling director for.
- virtual unsigned char **getDataStructureType** () const
Gets the DataStructureType that this class marshals/unmarshals.
- virtual void **tightUnmarshal** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataInputStream** *dataIn, **utils::BooleanStream** *bs)
Tight Un-marhsal to the given stream.
- virtual int **tightMarshal1** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **utils::BooleanStream** *bs)
Tight Marhsal to the given stream.
- virtual void **tightMarshal2** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataOutputStream** *dataOut, **utils::BooleanStream** *bs)
Tight Marhsal to the given stream.
- virtual void **looseUnmarshal** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataInputStream** *dataIn)
Loose Un-marhsal to the given stream.
- virtual void **looseMarshal** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataOutputStream** *dataOut)
Tight Marhsal to the given stream.

6.417.1 Detailed Description

Marshaling code for Open Wire Format for **ProducerInfoMarshaller** (p. 2195).

NOTE!: This file is auto generated - do not modify! if you need to make a change, please see the Java Classes in the activemq-openwire-generator module

6.417.2 Constructor & Destructor Documentation

6.417.2.1 **activemq::wireformat::openwire::marshal::generated-
::ProducerInfoMarshaller::ProducerInfoMarshaller ()**
[inline]

6.417.2.2 **virtual activemq::wireformat::openwire::marshal::generated:-
ProducerInfoMarshaller::~~ProducerInfoMarshaller ()** [inline,
virtual]

6.417.3 Member Function Documentation

6.417.3.1 **virtual commands::DataStructure* activemq::wireformat::openwire-
::marshal::generated::ProducerInfoMarshaller::createObject ()** const
[virtual]

Creates a new instance of the class that this class is a marshaling director for.

Returns

newly allocated Command

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller**
(p. 1120).

6.417.3.2 **virtual unsigned char activemq::wireformat::openwire::marshal-
::generated::ProducerInfoMarshaller::getDataStructureType ()** const
[virtual]

Gets the DataStructureType that this class marshals/unmarshals.

Returns

byte Id of this classes DataStructureType

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller**
(p. 1122).

6.417

activemq::wireformat::openwire::marshal::generated::ProducerInfoMarshaller

Class Reference

2203

6.417.3.3 virtual void **activemq::wireformat::openwire::marshal::generated::ProducerInfoMarshaller::looseMarshal** (**OpenWireFormat** * *format*,
commands::DataStructure * *command*, **decaf::io::DataOutputStream** * *ds*
) [virtual]

Tight Marhsal to the given stream.

Parameters

<i>format</i>	- The OpenWireFormat properties
<i>command</i>	- the object to Marshal
<i>ds</i>	- DataOutputStream to marshal to

Exceptions

<i>IOException</i>	if an error occurs.
--------------------	---------------------

Reimplemented from **activemq::wireformat::openwire::marshal::generated::Base-CommandMarshaller** (p. 500).

6.417.3.4 virtual void **activemq::wireformat::openwire::marshal::generated::ProducerInfoMarshaller::looseUnmarshal** (**OpenWireFormat** * *format*,
commands::DataStructure * *command*, **decaf::io::DataInputStream** * *dis*)
[virtual]

Loose Un-marhsal to the given stream.

Parameters

<i>format</i>	- The OpenWireFormat properties
<i>command</i>	- the object to Un-Marshal
<i>dis</i>	- the DataInputStream to Un-Marshal from

Exceptions

<i>IOException</i>	if an error occurs.
--------------------	---------------------

Reimplemented from **activemq::wireformat::openwire::marshal::generated::Base-CommandMarshaller** (p. 501).

6.417.3.5 virtual int **activemq::wireformat::openwire::marshal::generated::ProducerInfoMarshaller::tightMarshal1** (**OpenWireFormat** * *format*,
commands::DataStructure * *command*, **utils::BooleanStream** * *bs*)
[virtual]

Tight Marhsal to the given stream.

Parameters

<i>format</i>	The OpenwireFormat properties
<i>command</i>	The object to Marshal
<i>bs</i>	The boolean stream to marshal to.

Exceptions

<i>IOException</i>	if an error occurs.
--------------------	---------------------

Reimplemented from **activemq::wireformat::openwire::marshal::generated::Base-CommandMarshaller** (p. 503).

6.417.3.6 `virtual void activemq::wireformat::openwire::marshal::generated::ProducerInfoMarshaller::tightMarshal2 (OpenWireFormat * format, commands::DataStructure * command, decaf::io::DataOutputStream * ds, utils::BooleanStream * bs) [virtual]`

Tight Marshal to the given stream.

Parameters

<i>format</i>	- The OpenwireFormat properties
<i>command</i>	- the object to Marshal
<i>ds</i>	- the DataOutputStream to Marshal to
<i>bs</i>	- boolean stream to marshal to.

Exceptions

<i>IOException</i>	if an error occurs.
--------------------	---------------------

Reimplemented from **activemq::wireformat::openwire::marshal::generated::Base-CommandMarshaller** (p. 504).

6.417.3.7 `virtual void activemq::wireformat::openwire::marshal::generated::ProducerInfoMarshaller::tightUnmarshal (OpenWireFormat * format, commands::DataStructure * command, decaf::io::DataInputStream * dis, utils::BooleanStream * bs) [virtual]`

Tight Un-marshal to the given stream.

Parameters

<i>format</i>	- The OpenwireFormat properties
<i>command</i>	- the object to Un-Marshal
<i>dis</i>	- the DataInputStream to Un-Marshal from
<i>bs</i>	- boolean stream to unmarshal from.

Exceptions

<code>IOException</code>	if an error occurs.
--------------------------	---------------------

Reimplemented from `activemq::wireformat::openwire::marshal::generated::BaseCommandMarshaller` (p. 505).

The documentation for this class was generated from the following file:

- `src/main/activemq/wireformat/openwire/marshal/generated/ProducerInfoMarshaller.h`

6.418 `activemq::state::ProducerState` Class Reference

```
#include <src/main/activemq/state/ProducerState.h>
```

Public Member Functions

- **`ProducerState`** (const **`Pointer< ProducerInfo >`** &info)
- virtual **`~ProducerState`** ()
- **`std::string toString`** () const
- const **`Pointer< ProducerInfo >`** & **`getInfo`** () const
- void **`setTransactionState`** (const **`Pointer< TransactionState >`** &transactionState)
- **`Pointer< TransactionState >`** **`getTransactionState`** () const

6.418.1 Constructor & Destructor Documentation

6.418.1.1 **`activemq::state::ProducerState::ProducerState (const Pointer< ProducerInfo > & info)`**

6.418.1.2 **`virtual activemq::state::ProducerState::~~ProducerState ()`**
[virtual]

6.418.2 Member Function Documentation

6.418.2.1 **`const Pointer<ProducerInfo>& activemq::state::ProducerState::getInfo () const`** [inline]

6.418.2.2 **`Pointer<TransactionState> activemq::state::ProducerState::getTransactionState () const`**

6.418.2.3 **`void activemq::state::ProducerState::setTransactionState (const Pointer< TransactionState > & transactionState)`**

6.418.2.4 `std::string activemq::state::ProducerState::toString () const`

The documentation for this class was generated from the following file:

- `src/main/activemq/state/ProducerState.h`

6.419 `decaf::util::Properties` Class Reference

Java-like properties class for mapping string names to string values.

```
#include <src/main/decaf/util/Properties.h>
```

Public Member Functions

- **Properties** ()
- **Properties** (const **Properties** &src)
- virtual **~Properties** ()
- **Properties & operator=** (const **Properties** &src)
Assignment Operator.
- bool **isEmpty** () const
Returns true if the properties object is empty.
- int **size** () const
- const char * **getProperty** (const std::string &name) const
Looks up the value for the given property.
- std::string **getProperty** (const std::string &name, const std::string &defaultValue) const
Looks up the value for the given property.
- std::string **setProperty** (const std::string &name, const std::string &value)
Sets the value for a given property.
- bool **hasProperty** (const std::string &name) const
Check to see if the Property exists in the set.
- std::string **remove** (const std::string &name)
Removes the property with the given name.
- std::vector< std::string > **propertyNames** () const
Returns an enumeration of all the keys in this property list, including distinct keys in the default property list if a key of the same name has not already been found from the main properties list.
- std::vector< std::pair < std::string, std::string > > **toArray** () const
Method that serializes the contents of the property map to an array.
- void **copy** (const **Properties** &source)
*Copies the contents of the given properties object to this one, if the given **Properties** (p. 2200) instance is NULL then this **List** (p. 1658) is not modified.*
- **Properties** * **clone** () const
Clones this object.

- void **clear** ()
Clears all properties from the map.
- bool **equals** (const **Properties** &source) const
*Test whether two **Properties** (p. 2200) objects are equivalent.*
- std::string **toString** () const
*Formats the contents of the **Properties** (p. 2200) Object into a string that can be logged, etc.*
- void **load** (decaf::io::InputStream *stream)
Reads a property list (key and element pairs) from the input byte stream.
- void **load** (decaf::io::Reader *reader)
Reads a property list (key and element pairs) from the input character stream in a simple line-oriented format.
- void **store** (decaf::io::OutputStream *out, const std::string &comment)
*Writes this property list (key and element pairs) in this **Properties** (p. 2200) table to the output stream in a format suitable for loading into a **Properties** (p. 2200) table using the load(InputStream) method.*
- void **store** (decaf::io::Writer *writer, const std::string &comments)
*Writes this property list (key and element pairs) in this **Properties** (p. 2200) table to the output character stream in a format that can be read by the load(Reader) method.*

Protected Attributes

- decaf::lang::Pointer< **Properties** > **defaults**
Default list used to answer for any keys not found in the properties list, can be filled in by another implementation of this class.

6.419.1 Detailed Description

Java-like properties class for mapping string names to string values.

The **Properties** (p. 2200) list contains a key value pair of properties that can be loaded and stored to a stream. Each **Properties** (p. 2200) instance can contain an internal - **Properties** (p. 2200) list that contains default values for keys not found in the **Properties** (p. 2200) **List** (p. 1658).

The **Properties** (p. 2200) list if a Thread Safe class, it can be shared amongst objects in multiple threads without the need for additional synchronization.

Since

1.0

6.419.2 Constructor & Destructor Documentation

6.419.2.1 decaf::util::Properties::Properties ()

6.419.2.2 **decaf::util::Properties::Properties** (**const Properties & src**)

6.419.2.3 **virtual decaf::util::Properties::~~Properties** () [virtual]

6.419.3 Member Function Documentation

6.419.3.1 **void decaf::util::Properties::clear** ()

Clears all properties from the map.

6.419.3.2 **Properties* decaf::util::Properties::clone** () **const**

Clones this object.

Returns

a replica of this object.

6.419.3.3 **void decaf::util::Properties::copy** (**const Properties & source**)

Copies the contents of the given properties object to this one, if the given **Properties** (p. 2200) instance is NULL then this **List** (p. 1658) is not modified.

Parameters

<i>source</i>	The source properties object.
---------------	-------------------------------

6.419.3.4 **bool decaf::util::Properties::equals** (**const Properties & source**) **const**

Test whether two **Properties** (p. 2200) objects are equivalent.

Two **Properties** (p. 2200) Objects are considered equivalent when they each contain the same number of elements and each key / value pair contained within the two are equal.

This comparison does not check the contents of the Defaults instance.

Parameters

<i>source</i>	The Properties (p. 2200) object to compare this instance to.
---------------	---

Returns

true if the contents of the two **Properties** (p. 2200) objects are the same.

6.419.3.5 `const char* decaf::util::Properties::getProperty (const std::string & name)`
`const`

Looks up the value for the given property.

Parameters

<i>name</i>	The name of the property to be looked up.
-------------	---

Returns

the value of the property with the given name, if it exists. If it does not exist, returns NULL.

6.419.3.6 `std::string decaf::util::Properties::getProperty (const std::string & name,`
`const std::string & defaultValue) const`

Looks up the value for the given property.

Parameters

<i>name</i>	The name of the property to be looked up.
<i>defaultValue</i>	The value to be returned if the given property does not exist.

Returns

The value of the property specified by *name*, if it exists, otherwise the *defaultValue*.

6.419.3.7 `bool decaf::util::Properties::hasProperty (const std::string & name) const`

Check to see if the Property exists in the set.

Parameters

<i>name</i>	The property name to check for in this properties set.
-------------	--

Returns

true if property exists, false otherwise.

6.419.3.8 `bool decaf::util::Properties::isEmpty () const`

Returns true if the properties object is empty.

Returns

true if empty

6.419.3.9 void decaf::util::Properties::load (decaf::io::InputStream * *stream*)

Reads a property list (key and element pairs) from the input byte stream.

The input stream is in a simple line-oriented format as specified in load(Reader) and is assumed to use the ISO 8859-1 character encoding.

This method does not close the stream upon its return.

Parameters

<i>stream</i>	The stream to read the properties data from.
---------------	--

Exceptions

<i>IOException</i>	if there is an error while reading from the stream.
<i>IllegalArgument-Exception</i>	if malformed data is found while reading the properties.
<i>NullPointerException</i>	if the passed stream is Null.

6.419.3.10 void decaf::util::Properties::load (decaf::io::Reader * *reader*)

Reads a property list (key and element pairs) from the input character stream in a simple line-oriented format.

Properties (p. 2200) are processed in terms of lines. There are two kinds of line, natural lines and logical lines. A natural line is defined as a line of characters that is terminated either by a set of line terminator characters (

or or

) or by the end of the stream. A natural line may be either a blank line, a comment line, or hold all or some of a key-element pair. A logical line holds all the data of a key-element pair, which may be spread out across several adjacent natural lines by escaping the line terminator sequence with a backslash character \. Note that a comment line cannot be extended in this manner; every natural line that is a comment must have its own comment indicator, as described below. Lines are read from input until the end of the stream is reached.

A natural line that contains only white space characters is considered blank and is ignored. A comment line has an ASCII '#' or '!' as its first non-white space character; comment lines are also ignored and do not encode key-element information. In addition to line terminators, this format considers the characters space (' '), tab (""), and form feed ("") to be white space.

If a logical line is spread across several natural lines, the backslash escaping the line terminator sequence, the line terminator sequence, and any white space at the start of

the following line have no affect on the key or element values. The remainder of the discussion of key and element parsing (when loading) will assume all the characters constituting the key and element appear on a single natural line after line continuation characters have been removed. Note that it is not sufficient to only examine the character preceding a line terminator sequence to decide if the line terminator is escaped; there must be an odd number of contiguous backslashes for the line terminator to be escaped. Since the input is processed from left to right, a non-zero even number of $2n$ contiguous backslashes before a line terminator (or elsewhere) encodes n backslashes after escape processing.

The key contains all of the characters in the line starting with the first non-white space character and up to, but not including, the first unescaped '=', ':', or white space character other than a line terminator. All of these key termination characters may be included in the key by escaping them with a preceding backslash character; for example,

```
\:|=
```

would be the two-character key "=:=". Line terminator characters can be included using and

escape sequences. Any white space after the key is skipped; if the first non-white space character after the key is '=' or ':', then it is ignored and any white space characters after it are also skipped. All remaining characters on the line become part of the associated element string; if there are no remaining characters, the element is the empty string "". Once the raw character sequences constituting the key and element are identified, escape processing is performed as described above.

As an example, each of the following three lines specifies the key "Truth" and the associated element value "Beauty":

```
Truth = Beauty Truth:Beauty Truth :Beauty
```

As another example, the following three lines specify a single property:

```
fruits apple, banana, pear, \ cantaloupe, watermelon, \ kiwi, mango
```

The key is "fruits" and the associated element is: "apple, banana, pear, cantaloupe, watermelon, kiwi, mango"

Note that a space appears before each \ so that a space will appear after each comma in the final result; the \, line terminator, and leading white space on the continuation line are merely discarded and are not replaced by one or more other characters.

As a third example, the line:

```
cheeses
```

specifies that the key is "cheeses" and the associated element is the empty string "".

Characters in keys and elements can be represented in escape sequences similar to those used for character and string literals (see §3.3 and §3.10.6 of the Java Language Specification). The differences from the character escape sequences and Unicode escapes used for characters and strings are:

- Octal escapes are not recognized.
- The character sequence **does** not represent a backspace character.

- The method does not treat a backslash character, `\`, before a non-valid escape character as an error; the backslash is silently dropped. For example, in a C++ string the sequence `"\z"` would cause a compile time error. In contrast, this method silently drops the backslash. Therefore, this method treats the two character sequence `"\b"` as equivalent to the single character `'b'`.
- Escapes are not necessary for single and double quotes; however, by the rule above, single and double quote characters preceded by a backslash still yield single and double quote characters, respectively.

This method does not close the Reader upon its return.

Parameters

<i>reader</i>	The Reader that provides an character stream as input.
---------------	--

Exceptions

<i>IOException</i>	if there is an error while reading from the stream.
<i>IllegalArgument-Exception</i>	if malformed data is found while reading the properties.
<i>NullPointerException</i>	if the passed stream is Null.

6.419.3.11 **Properties&** `decaf::util::Properties::operator= (const Properties & src)`

Assignment Operator.

Parameters

<i>src</i>	The Properties (p. 2200) list to copy to this List (p. 1658).
------------	---

Returns

a reference to this **List** (p. 1658) for use in chaining.

6.419.3.12 `std::vector<std::string> decaf::util::Properties::propertyNames () const`

Returns an enumeration of all the keys in this property list, including distinct keys in the default property list if a key of the same name has not already been found from the main properties list.

Returns

a set of keys in this property list where the key and its corresponding value are strings, including the keys in the default property list.

6.419.3.13 `std::string decaf::util::Properties::remove (const std::string & name)`

Removes the property with the given name.

Parameters

<i>name</i>	The name of the property to remove.
-------------	-------------------------------------

Returns

the previous value of the property if set, or empty string.

6.419.3.14 `std::string decaf::util::Properties::setProperty (const std::string & name, const std::string & value)`

Sets the value for a given property.

If the property already exists, overwrites the value.

Parameters

<i>name</i>	The name of the value to be written.
<i>value</i>	The value to be written.

Returns

the old value of the property or empty string if not set.

6.419.3.15 `int decaf::util::Properties::size () const`**Returns**

The number of **Properties** (p. 2200) in this **Properties** (p. 2200) Object.

6.419.3.16 `void decaf::util::Properties::store (decaf::io::OutputStream * out, const std::string & comment)`

Writes this property list (key and element pairs) in this **Properties** (p. 2200) table to the output stream in a format suitable for loading into a **Properties** (p. 2200) table using the load(InputStream) method.

Properties (p. 2200) from the defaults table of this **Properties** (p. 2200) table (if any) are not written out by this method.

This method outputs the comments, properties keys and values in the same format as specified in store(Writer), with the following differences:

- The stream is written using the ISO 8859-1 character encoding.

- Characters not in Latin-1 in the comments are written as for their appropriate unicode hexadecimal value xxxx.
- Characters less than and characters greater than in property keys or values are written as for the appropriate hexadecimal value xxxx.

After the entries have been written, the output stream is flushed. The output stream remains open after this method returns.

Parameters

<i>out</i>	The OutputStream instance to write the properties to.
<i>comment</i>	A description of these properties that is written to the output stream.

Exceptions

<i>IOException</i>	if there is an error while writing from the stream.
<i>NullPointerException</i>	if the passed stream is Null.

6.419.3.17 `void decaf::util::Properties::store (decaf::io::Writer * writer, const std::string & comments)`

Writes this property list (key and element pairs) in this **Properties** (p. 2200) table to the output character stream in a format that can be read by the load(Reader) method.

Properties (p. 2200) from the defaults table of this **Properties** (p. 2200) table (if any) are not written out by this method.

If the comments argument is not empty, then an ASCII # character, the comments string, and a line separator are first written to the output stream. Thus, the comments can serve as an identifying comment. Any one of a line feed (

'), a carriage return ("), or a carriage return followed immediately by a line feed in comments is replaced by a line separator generated by the Writer and if the next character in comments is not character # or character ! then an ASCII # is written out after that line separator.

Next, a comment line is always written, consisting of an ASCII # character, the current date and time (as if produced by the toString method of **Date** (p. 1139) for the current time), and a line separator as generated by the Writer.

Then every entry in this **Properties** (p. 2200) table is written out, one per line. For each entry the key string is written, then an ASCII =, then the associated element string. For the key, all space characters are written with a preceding \ character. For the element, leading space characters, but not embedded or trailing space characters, are written with a preceding \ character. The key and element characters #, !, =, and : are written with a preceding backslash to ensure that they are properly loaded.

After the entries have been written, the output stream is flushed. The output stream remains open after this method returns.

Parameters

<i>writer</i>	The Writer instance to use to output the properties.
<i>comments</i>	A description of these properties that is written before writing the properties.

Exceptions

<i>IOException</i>	if there is an error while writing from the stream.
<i>NullPointerException</i>	if the passed stream is Null.

6.419.3.18 `std::vector< std::pair< std::string, std::string > >`
`decaf::util::Properties::toArray () const`

Method that serializes the contents of the property map to an array.

Returns

list of pairs where the first is the name and the second is the value.

6.419.3.19 `std::string decaf::util::Properties::toString () const`

Formats the contents of the **Properties** (p. 2200) Object into a string that can be logged, etc.

Returns

string value of this object.

6.419.4 Field Documentation

6.419.4.1 `decaf::lang::Pointer<Properties> decaf::util::Properties::defaults`
`[protected]`

Default list used to answer for any keys not found in the properties list, can be filled in by another implementation of this class.

The documentation for this class was generated from the following file:

- `src/main/decaf/util/Properties.h`

6.420 decaf::util::logging::PropertiesChangeListener Class - Reference

Defines the interface that classes can use to listen for change events on **Properties** (p. 2200).

```
#include <src/main/decaf/util/logging/PropertiesChangeListener.h>
```

Public Member Functions

- virtual **~PropertiesChangeListener** ()
- virtual void **onPropertiesReset** ()=0
*Indicates that the **Properties** (p. 2200) have all been reset and should be considered to be back to their default values.*
- virtual void **onPropertyChanged** (const std::string &name, const std::string &oldValue, const std::string &newValue)=0
Change Event, called when a property is changed, includes the name of the property that was changed along with it old and new values.

6.420.1 Detailed Description

Defines the interface that classes can use to listen for change events on **Properties** (p. 2200).

Since

1.0

6.420.2 Constructor & Destructor Documentation

6.420.2.1 virtual decaf::util::logging::PropertiesChangeListener::~PropertiesChangeListener () [inline, virtual]

6.420.3 Member Function Documentation

6.420.3.1 virtual void decaf::util::logging::PropertiesChangeListener::onPropertiesReset () [pure virtual]

Indicates that the **Properties** (p. 2200) have all been reset and should be considered to be back to their default values.

6.420.3.2 virtual void decaf::util::logging::PropertiesChangeListener::onPropertyChanged (const std::string & name, const std::string & oldValue, const std::string & newValue) [pure virtual]

Change Event, called when a property is changed, includes the name of the property that was changed along with it old and new values.

Parameters

<i>name</i>	The name of the Property that changed.
<i>oldValue</i>	The old Value of the Property.
<i>newValue</i>	The new Value of the Property.

The documentation for this class was generated from the following file:

- src/main/decaf/util/logging/**PropertiesChangeListener.h**

6.421 decaf::net::ProtocolException Class Reference

```
#include <src/main/decaf/net/ProtocolException.h>
```

Inheritance diagram for decaf::net::ProtocolException:

Public Member Functions

- **ProtocolException** () throw ()
Default Constructor.
- **ProtocolException** (const **Exception** &ex) throw ()
Conversion Constructor from some other Exception.
- **ProtocolException** (const **ProtocolException** &ex) throw ()
Copy Constructor.
- **ProtocolException** (const char *file, const int lineNumber, const std::exception *cause, const char *msg,...) throw ()
Constructor - Initializes the file name and line number where this message occurred.
- **ProtocolException** (const std::exception *cause) throw ()
Constructor.
- **ProtocolException** (const char *file, const int lineNumber, const char *msg,...) throw ()
Constructor - Initializes the file name and line number where this message occurred.
- virtual **ProtocolException** * clone () const
Clones this exception.
- virtual ~**ProtocolException** () throw ()

6.421.1 Constructor & Destructor Documentation

6.421.1.1 decaf::net::ProtocolException::ProtocolException () throw ()
[inline]

Default Constructor.

6.421.1.2 **decaf::net::ProtocolException::ProtocolException (const Exception & ex) throw () [inline]**

Conversion Constructor from some other Exception.

Parameters

<i>ex</i>	An exception that should become this type of Exception
-----------	--

6.421.1.3 **decaf::net::ProtocolException::ProtocolException (const ProtocolException & ex) throw () [inline]**

Copy Constructor.

Parameters

<i>ex</i>	An exception that should become this type of Exception
-----------	--

6.421.1.4 **decaf::net::ProtocolException::ProtocolException (const char * file, const int lineNumber, const std::exception * cause, const char * msg, ...) throw () [inline]**

Constructor - Initializes the file name and line number where this message occurred.

Sets the message to report, using an optional list of arguments to parse into the message

Parameters

<i>file</i>	The file name where exception occurs
<i>lineNumber</i>	The line number where the exception occurred.
<i>cause</i>	The exception that was the cause for this one to be thrown.
<i>msg</i>	The message to report
<i>...</i>	list of primitives that are formatted into the message

6.421.1.5 **decaf::net::ProtocolException::ProtocolException (const std::exception * cause) throw () [inline]**

Constructor.

Parameters

<i>cause</i>	Pointer to the exception that caused this one to be thrown, the object is cloned caller retains ownership.
--------------	--

6.421.1.6 **decaf::net::ProtocolException::ProtocolException** (const char * *file*, const int *lineNumber*, const char * *msg*, ...) throw () [inline]

Constructor - Initializes the file name and line number where this message occurred.

Sets the message to report, using an optional list of arguments to parse into the message

Parameters

<i>file</i>	The file name where exception occurs
<i>lineNumber</i>	The line number where the exception occurred.
<i>msg</i>	The message to report
...	list of primitives that are formatted into the message

6.421.1.7 **virtual decaf::net::ProtocolException::~~ProtocolException** () throw () [inline, virtual]

6.421.2 Member Function Documentation

6.421.2.1 **virtual ProtocolException* decaf::net::ProtocolException::clone** () const [inline, virtual]

Clones this exception.

This is useful for cases where you need to preserve the type of the original exception as well as the message. All subclasses should override.

Returns

a new Exception instance that is a copy of this Exception object.

Reimplemented from **decaf::io::IOException** (p. 1547).

The documentation for this class was generated from the following file:

- src/main/decaf/net/**ProtocolException.h**

6.422 decaf::security::PublicKey Class Reference

A public key.

```
#include <src/main/decaf/security/PublicKey.h>
```

Inheritance diagram for decaf::security::PublicKey:

Public Member Functions

- virtual `~PublicKey ()`

6.422.1 Detailed Description

A public key.

This interface contains no methods or constants. It merely serves to group (and provide type safety for) all public key interfaces.

6.422.2 Constructor & Destructor Documentation

6.422.2.1 `virtual decaf::security::PublicKey::~~PublicKey () [inline, virtual]`

The documentation for this class was generated from the following file:

- `src/main/decaf/security/PublicKey.h`

6.423 decaf::io::PushbackInputStream Class Reference

A **PushbackInputStream** (p. 2214) adds functionality to another input stream, namely the ability to "push back" or "unread" one byte.

```
#include <src/main/decaf/io/PushbackInputStream.h>
```

Inheritance diagram for `decaf::io::PushbackInputStream`:

Public Member Functions

- **PushbackInputStream (InputStream *stream, bool own=false)**
*Creates a **PushbackInputStream** (p. 2214) and saves its argument, the input stream in, for later use.*
- **PushbackInputStream (InputStream *stream, int bufSize, bool own=false)**
*Creates a **PushbackInputStream** (p. 2214) and saves its argument, the input stream in, for later use.*
- virtual `~PushbackInputStream ()`
- void **unread** (unsigned char value)
Pushes back the given byte, the byte is copied to the front of the pushback buffer, future calls to read start reading from the beginning of these pushed back byte.
- void **unread** (const unsigned char *buffer, int size)
Pushes back the given array of bytes, the bytes are copied to the front of the pushback buffer, future calls to read start reading from the beginning of these pushed back bytes.

- void **unread** (const unsigned char *buffer, int size, int offset, int length)

Pushes back the given array of bytes, the bytes are copied to the front of the pushback buffer, future calls to read start reading from the beginning of these pushed back bytes.

- virtual int **available** () const

Indicates the number of bytes available.

The default implementation of this methods returns 0. Classes that override this method may return the total number of bytes that are currently available for reading and others may simply return a value of one indicating that there is some data available. The caller should view the result of this method as an absolute.

The default implementation of this method returns zero.

Returns

the number of bytes available on this input stream.

Exceptions

IOException (p. 1545)	<i>if an I/O error occurs.</i>
------------------------------	--------------------------------

- virtual long long **skip** (long long num)

Skips over and discards n bytes of data from this input stream.

The skip method may, for a variety of reasons, end up skipping over some smaller number of bytes, possibly 0. This may result from any of a number of conditions; reaching end of file before n bytes have been skipped is only one possibility. The actual number of bytes skipped is returned.

*The skip method of **InputStream** (p. 1464) creates a byte array and then repeatedly reads into it until num bytes have been read or the end of the stream has been reached. Subclasses are encouraged to provide a more efficient implementation of this method.*

Parameters

num	<i>The number of bytes to skip.</i>
-----	-------------------------------------

Returns

total bytes skipped

Exceptions

IOException (p. 1545)	<i>if an I/O error occurs.</i>
UnsupportedOperationException	<i>if the concrete stream class does not support skipping bytes.</i>

- virtual void **mark** (int readLimit)

*Does nothing except throw an **IOException** (p. 1545).*

- virtual void **reset** ()

*Does nothing except throw an **IOException** (p. 1545).*

- virtual bool **markSupported** () const

*Does nothing except throw an **IOException** (p. 1545).*

Protected Member Functions

- virtual int **doReadByte** ()
- virtual int **doReadArrayBounded** (unsigned char *buffer, int size, int offset, int length)

6.423.1 Detailed Description

A **PushbackInputStream** (p. 2214) adds functionality to another input stream, namely the ability to "push back" or "unread" one byte.

This is useful in situations where it is convenient for a fragment of code to read an indefinite number of data bytes that are delimited by a particular byte value; after reading the terminating byte, the code fragment can "unread" it, so that the next read operation on the input stream will reread the byte that was pushed back. For example, bytes representing the characters constituting an identifier might be terminated by a byte representing an operator character; a method whose job is to read just an identifier can read until it sees the operator and then push the operator back to be re-read.

Since

1.0

6.423.2 Constructor & Destructor Documentation

6.423.2.1 `decaf::io::PushbackInputStream::PushbackInputStream (InputStream * stream, bool own = false)`

Creates a **PushbackInputStream** (p. 2214) and saves its argument, the input stream in, for later use.

Initially, there is no pushed-back byte.

Parameters

<i>stream</i>	The InputStream (p. 1464) instance to wrap.
<i>Boolean</i>	value indicating if this FilterInputStream (p. 1334) owns the wrapped stream.

6.423.2.2 `decaf::io::PushbackInputStream::PushbackInputStream (InputStream * stream, int bufSize, bool own = false)`

Creates a **PushbackInputStream** (p. 2214) and saves its argument, the input stream in, for later use.

Initially, there is no pushed-back byte.

Parameters

<i>stream</i>	The InputStream (p. 1464) instance to wrap.
<i>bufSize</i>	The number of byte to allocate for pushback into this stream.
<i>Boolean</i>	value indicating if this FilterInputStream (p. 1334) owns the wrapped stream.

Exceptions

<i>IllegalArgumentException</i>	if the bufSize argument is < zero.
---------------------------------	------------------------------------

6.423.2.3 virtual **decaf::io::PushbackInputStream::~~PushbackInputStream** ()
[virtual]

6.423.3 Member Function Documentation

6.423.3.1 virtual int **decaf::io::PushbackInputStream::available** () const
[virtual]

Indicates the number of bytes available.

The default implementation of this methods returns 0. Classes that override this method may return the total number of bytes that are currently available for reading and others may simply return a value of one indicating that there is some data available. The caller should view the result of this method as an absolute.

The default implementation of this method returns zero.

Returns

the number of bytes available on this input stream.

Exceptions

<i>IOException</i> (p. 1545)	if an I/O error occurs.
---------------------------------	-------------------------

Returns the sum of the number of pushed back bytes if any and the amount of bytes available in the underlying stream via a call to available.

Reimplemented from **decaf::io::FilterInputStream** (p. 1337).

6.423.3.2 virtual int **decaf::io::PushbackInputStream::doReadArrayBounded** (
unsigned char * *buffer*, int *size*, int *offset*, int *length*) [protected,
virtual]

Reimplemented from **decaf::io::FilterInputStream** (p. 1338).

6.423.3.3 virtual int **decaf::io::PushbackInputStream::doReadByte** ()
[protected, virtual]

Reimplemented from **decaf::io::FilterInputStream** (p. 1338).

6.423.3.4 `virtual void decaf::io::PushbackInputStream::mark (int readLimit)`
`[virtual]`

Does nothing except throw an **IOException** (p. 1545).

Marks the current position in the stream. A subsequent call to the `reset` method repositions this stream at the last marked position so that subsequent reads re-read the same bytes.

If a stream instance reports that marks are supported then the stream will ensure that the same bytes can be read again after the `reset` method is called so long the `readLimit` is not reached.

Calling `mark` on a closed stream instance should have no effect.

The default implementation of this method does nothing.

Parameters

<i>readLimit</i>	The max bytes read before marked position is invalid.
------------------	---

Reimplemented from **decaf::io::FilterInputStream** (p. 1338).

6.423.3.5 `virtual bool decaf::io::PushbackInputStream::markSupported () const`
`[inline, virtual]`

Does nothing except throw an **IOException** (p. 1545).

Determines if this input stream supports the `mark` and `reset` methods.

Whether or not `mark` and `reset` are supported is an invariant property of a particular input stream instance.

The default implementation of this method returns `false`.

Returns

`true` if this stream instance supports marks

Reimplemented from **decaf::io::FilterInputStream** (p. 1339).

6.423.3.6 `virtual void decaf::io::PushbackInputStream::reset ()` `[virtual]`

Does nothing except throw an **IOException** (p. 1545).

Repositions this stream to the position at the time the `mark` method was last called on this input stream.

If the method `markSupported` returns `true`, then: * If the method `mark` has not been called since the stream was created, or the number of bytes read from the stream since `mark` was last called is larger than the argument to `mark` at that last call, then an **IOException** (p. 1545) might be thrown. * If such an **IOException** (p. 1545) is not thrown, then the stream is reset to a state such that all the bytes read since the most recent

call to mark (or since the start of the file, if mark has not been called) will be resupplied to subsequent callers of the read method, followed by any bytes that otherwise would have been the next input data as of the time of the call to reset.

If the method markSupported returns false, then: * The call to reset may throw an **IOException** (p. 1545). * If an **IOException** (p. 1545) is not thrown, then the stream is reset to a fixed state that depends on the particular type of the input stream and how it was created. The bytes that will be supplied to subsequent callers of the read method depend on the particular type of the input stream.

The default implementation of this method throws an **IOException** (p. 1545).

Exceptions

IOException (p. 1545)	if an I/O error occurs.
---------------------------------	-------------------------

Reimplemented from **decaf::io::FilterInputStream** (p. 1339).

6.423.3.7 virtual long long decaf::io::PushbackInputStream::skip (long long num)
[virtual]

Skips over and discards n bytes of data from this input stream.

The skip method may, for a variety of reasons, end up skipping over some smaller number of bytes, possibly 0. This may result from any of a number of conditions; reaching end of file before n bytes have been skipped is only one possibility. The actual number of bytes skipped is returned.

The skip method of **InputStream** (p. 1464) creates a byte array and then repeatedly reads into it until num bytes have been read or the end of the stream has been reached. Subclasses are encouraged to provide a more efficient implementation of this method.

Parameters

<i>num</i>	The number of bytes to skip.
------------	------------------------------

Returns

total bytes skipped

Exceptions

IOException (p. 1545)	if an I/O error occurs.
<i>UnsupportedOperationException</i>	if the concrete stream class does not support skipping bytes.

This method first skips bytes in the local pushed back buffer before attempting to complete the request by calling the underlying stream skip method with the remainder of

bytes that needs to be skipped.

Reimplemented from **decaf::io::FilterInputStream** (p. 1340).

6.423.3.8 void decaf::io::PushbackInputStream::unread (unsigned char *value*)

Pushes back the given byte, the byte is copied to the front of the pushback buffer, future calls to read start reading from the beginning of these pushed back byte.

Parameters

<i>value</i>	The byte that is to be placed at the front of the push back buffer.
--------------	---

Exceptions

<i>IOException</i> (p. 1545)	if there is not enough space in the pushback buffer or this stream has already been closed.
--	---

6.423.3.9 void decaf::io::PushbackInputStream::unread (const unsigned char * *buffer*, int *size*)

Pushes back the given array of bytes, the bytes are copied to the front of the pushback buffer, future calls to read start reading from the beginning of these pushed back bytes.

Parameters

<i>buffer</i>	The bytes to copy to the front of push back buffer.
<i>size</i>	The size of the array to be copied.

Exceptions

<i>NullPointerException</i>	if the buffer passed is NULL.
<i>IndexOutOfBoundsException</i>	if the size value given is negative.
<i>IOException</i> (p. 1545)	if there is not enough space in the pushback buffer or this stream has already been closed.

6.423.3.10 void decaf::io::PushbackInputStream::unread (const unsigned char * *buffer*, int *size*, int *offset*, int *length*)

Pushes back the given array of bytes, the bytes are copied to the front of the pushback buffer, future calls to read start reading from the beginning of these pushed back bytes.

Parameters

<i>buffer</i>	The bytes to copy to the front of push back buffer.
<i>size</i>	The size of the array to be copied.

<i>offset</i>	The position in the buffer to start copying from.
<i>length</i>	The number of bytes to push back from the passed buffer.

Exceptions

<i>NullPointerException</i>	if the buffer passed is NULL.
<i>IndexOutOfBoundsException</i>	if the offset + length is greater than the buffer size.
IOException (p. 1545)	if there is not enough space in the pushback buffer or this stream has already been closed.

The documentation for this class was generated from the following file:

- src/main/decaf/io/**PushbackInputStream.h**

6.424 cms::Queue Class Reference

An interface encapsulating a provider-specific queue name.

```
#include <src/main/cms/Queue.h>
```

Inheritance diagram for cms::Queue:

Public Member Functions

- virtual \sim **Queue** () throw ()
- virtual std::string **getQueueName** () const =0

Gets the name of this queue.

6.424.1 Detailed Description

An interface encapsulating a provider-specific queue name.

Messages sent to a **Queue** (p. 2221) are sent to a Single Subscriber on that **Queue** (p. 2221) **Destination** (p. 1210). This allows for Queues to be used as load balances implementing a SEDA based architecture. The length of time that a Provider will store a **Message** (p. 1839) in a **Queue** (p. 2221) is not defined by the CMS API, consult your Provider documentation for this information.

Since

1.0

6.424.2 Constructor & Destructor Documentation

6.424.2.1 `virtual cms::Queue::~~Queue () throw ()` `[virtual]`

6.424.3 Member Function Documentation

6.424.3.1 `virtual std::string cms::Queue::getQueueName () const` `[pure virtual]`

Gets the name of this queue.

Returns

The queue name.

Exceptions

<i>CMSException</i> (p. 826)	- If an internal error occurs.
--	--------------------------------

Implemented in `activemq::commands::ActiveMQQueue` (p. 325).

The documentation for this class was generated from the following file:

- `src/main/cms/Queue.h`

6.425 decaf::util::Queue< E > Class Template Reference

A kind of collection provides advanced operations than other basic collections, such as insertion, extraction, and inspection.

```
#include <src/main/decaf/util/Queue.h>
```

Inheritance diagram for `decaf::util::Queue< E >`:

Public Member Functions

- `virtual ~Queue ()`
- `virtual bool offer (const E &value)=0`
Inserts the specified element into the queue provided that the condition allows such an operation.
- `virtual bool poll (E &result)=0`
Gets and removes the element in the head of the queue.
- `virtual E remove ()=0`
Gets and removes the element in the head of the queue.

- virtual bool **peek** (E &result) const =0
Gets but not removes the element in the head of the queue.
- virtual E **element** () const =0
Gets but not removes the element in the head of the queue.

6.425.1 Detailed Description

template<typename E>class decaf::util::Queue< E >

A kind of collection provides advanced operations than other basic collections, such as insertion, extraction, and inspection.

Generally, a queue orders its elements by means of first-in-first-out. While priority queue orders its elements according to a comparator specified or the elements' natural order. Furthermore, a stack orders its elements last-in-first out.

Queue (p. 2222) does not provide blocking queue methods, which will block until the operation of the method is allowed. BlockingQueue interface defines such methods.

Unlike the Java **Queue** (p. 2222) interface the methods of this class cannot return null to indicate that a **Queue** (p. 2222) is empty since null has no meaning for elements such as classes, structs and primitive types and cannot be used in a meaningful way to check for an empty queue. Methods that would have returned null in the Java - **Queue** (p. 2222) interface have been altered to return a boolean value indicating if the operation succeeded and take single argument that is a reference to the location where the returned value is to be assigned. This implies that elements in the **Queue** (p. 2222) must be *assignable* in order to utilize these methods.

Since

1.0

6.425.2 Constructor & Destructor Documentation

6.425.2.1 template<typename E> virtual decaf::util::Queue< E >::~~Queue ()
[inline, virtual]

6.425.3 Member Function Documentation

6.425.3.1 template<typename E> virtual E decaf::util::Queue< E >::element () const
[pure virtual]

Gets but not removes the element in the head of the queue.

Throws a **NoSuchElementException** (p. 1984) if there is no element in the queue.

Returns

the element in the head of the queue.

Exceptions

<i>NoSuchElementException</i> (p. 1984)	if there is no element in the queue.
---	--------------------------------------

Implemented in **decaf::util::LinkedList< E >** (p. 1645), **decaf::util::LinkedList< Pointer< Transport > >** (p. 1645), **decaf::util::LinkedList< cms::MessageConsumer * >** (p. 1645), **decaf::util::LinkedList< CompositeTask * >** (p. 1645), **decaf::util::LinkedList< URI >** (p. 1645), **decaf::util::LinkedList< Pointer< MessageDispatch > >** (p. 1645), **decaf::util::LinkedList< Pointer< DestinationInfo > >** (p. 1645), **decaf::util::LinkedList< PrimitiveValueNode >** (p. 1645), **decaf::util::LinkedList< Pointer< Command > >** (p. 1645), **decaf::util::LinkedList< Pointer< BackupTransport > >** (p. 1645), **decaf::util::LinkedList< cms::MessageProducer * >** (p. 1645), **decaf::util::LinkedList< cms::Destination * >** (p. 1645), **decaf::util::LinkedList< cms::Session * >** (p. 1645), **decaf::util::LinkedList< cms::Connection * >** (p. 1645), and **decaf::util::AbstractQueue< E >** (p. 142).

6.425.3.2 `template<typename E> virtual bool decaf::util::Queue< E >::offer (const E & value) [pure virtual]`

Inserts the specified element into the queue provided that the condition allows such an operation.

The method is generally preferable to the `collection.add(E)`, since the latter might throw an exception if the operation fails.

Parameters

<i>value</i>	the specified element to insert into the queue.
--------------	---

Returns

true if the operation succeeds and false if it fails.

Exceptions

<i>NullPointerException</i>	if the Queue (p. 2222) implementation does not allow Null values to be inserted into the Queue (p. 2222).
<i>IllegalArgument-Exception</i>	if some property of the specified element prevents it from being added to this queue

Implemented in **decaf::util::concurrent::LinkedBlockingQueue< E >** (p. 1627), **decaf::util::LinkedList< E >** (p. 1649), **decaf::util::LinkedList< Pointer< Transport > >** (p. 1649), **decaf::util::LinkedList< cms::MessageConsumer * >** (p. 1649), **decaf::util::LinkedList< CompositeTask * >** (p. 1649), **decaf::util::LinkedList< URI >** (p. 1649), **decaf::util::LinkedList< Pointer< MessageDispatch > >** (p. 1649), **decaf::util::LinkedList< Pointer< DestinationInfo > >** (p. 1649),

decaf::util::LinkedList< PrimitiveValueNode > (p. 1649), **decaf::util::LinkedList< Pointer< Command > >** (p. 1649), **decaf::util::LinkedList< Pointer< BackupTransport > >** (p. 1649), **decaf::util::LinkedList< cms::MessageProducer * >** (p. 1649), **decaf::util::LinkedList< cms::Destination * >** (p. 1649), **decaf::util::LinkedList< cms::Session * >** (p. 1649), **decaf::util::LinkedList< cms::Connection * >** (p. 1649), **decaf::util::PriorityQueue< E >** (p. 2167), and **decaf::util::concurrent::SynchronousQueue< E >** (p. 2662).

Referenced by **decaf::util::AbstractQueue< E >::add()**.

6.425.3.3 `template<typename E> virtual bool decaf::util::Queue< E >::peek (E & result) const [pure virtual]`

Gets but not removes the element in the head of the queue.

The result if successful is assigned to the result parameter.

Parameters

<i>result</i>	Reference to an instance of the contained type to assigned the removed value to.
---------------	--

Returns

true if the element at the head of the queue was removed and assigned to the result parameter.

Implemented in **decaf::util::concurrent::LinkedBlockingQueue< E >** (p. 1628), **decaf::util::LinkedList< E >** (p. 1651), **decaf::util::LinkedList< Pointer< Transport > >** (p. 1651), **decaf::util::LinkedList< cms::MessageConsumer * >** (p. 1651), **decaf::util::LinkedList< CompositeTask * >** (p. 1651), **decaf::util::LinkedList< URI >** (p. 1651), **decaf::util::LinkedList< Pointer< MessageDispatch > >** (p. 1651), **decaf::util::LinkedList< Pointer< DestinationInfo > >** (p. 1651), **decaf::util::LinkedList< PrimitiveValueNode >** (p. 1651), **decaf::util::LinkedList< Pointer< Command > >** (p. 1651), **decaf::util::LinkedList< Pointer< BackupTransport > >** (p. 1651), **decaf::util::LinkedList< cms::MessageProducer * >** (p. 1651), **decaf::util::LinkedList< cms::Destination * >** (p. 1651), **decaf::util::LinkedList< cms::Session * >** (p. 1651), **decaf::util::LinkedList< cms::Connection * >** (p. 1651), and **decaf::util::PriorityQueue< E >** (p. 2168).

Referenced by **decaf::util::AbstractQueue< E >::element()**.

6.425.3.4 `template<typename E> virtual bool decaf::util::Queue< E >::poll (E & result) [pure virtual]`

Gets and removes the element in the head of the queue.

If the operation succeeds the value of the element at the head of the **Queue** (p. 2222) is assigned to the result parameter and the method returns true. If the operation fails the method returns false and the value of the result parameter is undefined.

Parameters

<i>result</i>	Reference to an instance of the contained type to assigned the removed value to.
---------------	--

Returns

true if the element at the head of the queue was removed and assigned to the result parameter.

Implemented in `decaf::util::concurrent::LinkedBlockingQueue< E >` (p. 1629), `decaf::util::LinkedList< E >` (p. 1652), `decaf::util::LinkedList< Pointer< Transport > >` (p. 1652), `decaf::util::LinkedList< cms::MessageConsumer * >` (p. 1652), `decaf::util::LinkedList< CompositeTask * >` (p. 1652), `decaf::util::LinkedList< URI >` (p. 1652), `decaf::util::LinkedList< Pointer< MessageDispatch > >` (p. 1652), `decaf::util::LinkedList< Pointer< DestinationInfo > >` (p. 1652), `decaf::util::LinkedList< PrimitiveValueNode >` (p. 1652), `decaf::util::LinkedList< Pointer< Command > >` (p. 1652), `decaf::util::LinkedList< Pointer< BackupTransport > >` (p. 1652), `decaf::util::LinkedList< cms::MessageProducer * >` (p. 1652), `decaf::util::LinkedList< cms::Destination * >` (p. 1652), `decaf::util::LinkedList< cms::Session * >` (p. 1652), `decaf::util::LinkedList< cms::Connection * >` (p. 1652), `decaf::util::PriorityQueue< E >` (p. 2169), and `decaf::util::concurrent::SynchronousQueue< E >` (p. 2663).

Referenced by `decaf::util::AbstractQueue< E >::clear()`, and `decaf::util::AbstractQueue< E >::remove()`.

6.425.3.5 `template<typename E> virtual E decaf::util::Queue< E >::remove ()`
[pure virtual]

Gets and removes the element in the head of the queue.

Throws a **NoSuchElementException** (p. 1984) if there is no element in the queue.

Returns

the element in the head of the queue.

Exceptions

NoSuchElementException (p. 1984)	if there is no element in the queue.
--	--------------------------------------

Implemented in `decaf::util::LinkedList< E >` (p. 1655), `decaf::util::LinkedList< Pointer< Transport > >` (p. 1655), `decaf::util::LinkedList< cms::MessageConsumer * >` (p. 1655), `decaf::util::LinkedList< CompositeTask * >` (p. 1655), `decaf::util::LinkedList< URI >` (p. 1655), `decaf::util::LinkedList< Pointer< MessageDispatch > >` (p. 1655), `decaf::util::LinkedList< Pointer< DestinationInfo > >` (p. 1655), `decaf::util::LinkedList< PrimitiveValueNode >` (p. 1655),

decaf::util::LinkedList< **Pointer**< **Command** > > (p. 1655), **decaf::util::LinkedList**< **Pointer**< **BackupTransport** > > (p. 1655), **decaf::util::LinkedList**< **cms::MessageProducer** * > (p. 1655), **decaf::util::LinkedList**< **cms::Destination** * > (p. 1655), **decaf::util::LinkedList**< **cms::Session** * > (p. 1655), **decaf::util::LinkedList**< **cms::Connection** * > (p. 1655), **decaf::util::PriorityQueue**< **E** > (p. 2169), and **decaf::util::AbstractQueue**< **E** > (p. 142).

The documentation for this class was generated from the following file:

- src/main/decaf/util/**Queue.h**

6.426 cms::QueueBrowser Class Reference

This class implements in interface for browsing the messages in a **Queue** (p. 2221) without removing them.

```
#include <src/main/cms/QueueBrowser.h>
```

Inheritance diagram for cms::QueueBrowser:

Public Member Functions

- virtual **~QueueBrowser** () throw ()
- virtual const **Queue** * **getQueue** () const =0
- virtual std::string **getMessageSelector** () const =0
- virtual **cms::MessageEnumeration** * **getEnumeration** ()=0

*Gets a pointer to an Enumeration object for browsing the Messages currently in the **Queue** (p. 2221) in the order that a client would receive them.*

6.426.1 Detailed Description

This class implements in interface for browsing the messages in a **Queue** (p. 2221) without removing them.

To browse the contents of the **Queue** (p. 2221) the client calls the `getEnumeration` method to retrieve a new instance of a **Queue** (p. 2221) Enumerator. The client then calls the `hasMoreMessages` method of the Enumeration, if it returns true the client can safely call the `nextMessage` method of the Enumeration instance.

```
Enumeration* enumeration = queueBrowser->getEnumeration() (p. 2228);
```

```
while( enumeration->hasMoreMessages() ) { cms::Message (p. 1839)* message =
enumeration->nextMessage();
```

```
// ... Do something with the Message (p. 1839).
```

```
delete message; }
```

Since

1.1

6.426.2 Constructor & Destructor Documentation

6.426.2.1 `virtual cms::QueueBrowser::~~QueueBrowser () throw ()` [virtual]

6.426.3 Member Function Documentation

6.426.3.1 `virtual cms::MessageEnumeration* cms::QueueBrowser::get-Enumeration ()` [pure virtual]

Gets a pointer to an Enumeration object for browsing the Messages currently in the **Queue** (p. 2221) in the order that a client would receive them.

The pointer returned is owned by the browser and should not be deleted by the client application.

Returns

a pointer to a **Queue** (p. 2221) Enumeration, this Pointer is owned by the **Queue-Browser** (p. 2227) and should not be deleted by the client.

Exceptions

CMSEException (p. 826)	if an internal error occurs.
----------------------------------	------------------------------

Implemented in **activemq::core::ActiveMQQueueBrowser** (p. 327).

6.426.3.2 `virtual std::string cms::QueueBrowser::getMessageSelector () const` [pure virtual]

Returns

the MessageSelector that is used on when this browser was created or empty string if no selector was present.

Exceptions

CMSEException (p. 826)	if an internal error occurs.
----------------------------------	------------------------------

Implemented in **activemq::core::ActiveMQQueueBrowser** (p. 327).

6.426.3.3 `virtual const Queue* cms::QueueBrowser::getQueue () const` `[pure virtual]`

Returns

the **Queue** (p. 2221) that this browser is listening on.

Exceptions

<i>CMSException</i> (p. 826)	if an internal error occurs.
--	------------------------------

Implemented in **activemq::core::ActiveMQQueueBrowser** (p. 328).

The documentation for this class was generated from the following file:

- `src/main/cms/QueueBrowser.h`

6.427 decaf::util::Random Class Reference

Random (p. 2229) Value Generator which is used to generate a stream of pseudorandom numbers.

```
#include <src/main/decaf/util/Random.h>
```

Inheritance diagram for `decaf::util::Random`:

Public Member Functions

- **Random** ()
Construct a random generator with the current time of day in milliseconds as the initial state.
- **Random** (unsigned long long seed)
*Construct a random generator with the given *seed* as the initial state.*
- virtual **~Random** ()
- bool **nextBoolean** ()
Answers the next pseudo-random, uniformly distributed boolean value generated by this generator.
- double **nextDouble** ()
Generates a normally distributed random double number between 0.0 inclusively and 1.0 exclusively.
- float **nextFloat** ()
Generates a normally distributed random float number between 0.0 inclusively and 1.0 exclusively.
- double **nextGaussian** ()

Pseudo-randomly generates (approximately) a normally distributed `double` value with mean 0.0 and a standard deviation value of 1.0 using the polar method of G.

- `int nextInt ()`

Generates a uniformly distributed 32-bit `int` value from the this random number sequence.

- `int nextInt (int n)`

Returns to the caller a new pseudo-random integer value which is uniformly distributed between 0 (inclusively) and the value of `n` (exclusively).

- `long long nextLong ()`

Generates a uniformly distributed 64-bit `int` value from the this random number sequence.

- `virtual void nextBytes (std::vector< unsigned char > &buf)`

Modifies the byte array by a random sequence of bytes generated by this random number generator.

- `virtual void nextBytes (unsigned char *buf, int size)`

Modifies the byte array by a random sequence of bytes generated by this random number generator.

- `virtual void setSeed (unsigned long long seed)`

*Modifies the seed using linear congruential formula presented in *The Art of Computer Programming, Volume 2, Section 3.2.1.**

Protected Member Functions

- `virtual int next (int bits)`

Answers a pseudo-random uniformly distributed `int` value of the number of bits specified by the argument `bits` as described by Donald E.

6.427.1 Detailed Description

Random (p. 2229) Value Generator which is used to generate a stream of pseudorandom numbers.

The algorithms implemented by class **Random** (p. 2229) use a protected utility method that on each invocation can supply up to 32 pseudorandomly generated bits.

Since

1.0

6.427.2 Constructor & Destructor Documentation

6.427.2.1 `decaf::util::Random::Random ()`

Construct a random generator with the current time of day in milliseconds as the initial state.

See also

setSeed (p. 2235)

6.427.2.2 decaf::util::Random::Random (unsigned long long *seed*)

Construct a random generator with the given *seed* as the initial state.

Parameters

<i>seed</i>	the seed that will determine the initial state of this random number generator
-------------	--

See also

setSeed (p. 2235)

6.427.2.3 virtual decaf::util::Random::~~Random () [virtual]

6.427.3 Member Function Documentation

6.427.3.1 virtual int decaf::util::Random::next (int *bits*) [protected, virtual]

Answers a pseudo-random uniformly distributed `int` value of the number of bits specified by the argument *bits* as described by Donald E.

Knuth in *The Art of Computer Programming, Volume 2: Seminumerical Algorithms*, section 3.2.1.

Returns

`int` a pseudo-random generated `int` number

Parameters

<i>bits</i>	number of bits of the returned value
-------------	--------------------------------------

See also

nextBytes (p. 2232)

nextDouble (p. 2233)

nextFloat (p. 2233)

nextInt() (p. 2234)

nextInt(int) (p. 2234)

nextGaussian (p. 2233)

nextLong (p. 2234)

Reimplemented in **decaf::security::SecureRandom** (p. 2323).

6.427.3.2 `bool decaf::util::Random::nextBoolean ()`

Answers the next pseudo-random, uniformly distributed boolean value generated by this generator.

Returns

boolean a pseudo-random, uniformly distributed boolean value

6.427.3.3 `virtual void decaf::util::Random::nextBytes (std::vector< unsigned char > & buf) [virtual]`

Modifies the byte array by a random sequence of bytes generated by this random number generator.

Parameters

<i>buf</i>	non-null array to contain the new random bytes
------------	--

See also

next (p. 2231)

Reimplemented in **decaf::security::SecureRandom** (p. 2324).

6.427.3.4 `virtual void decaf::util::Random::nextBytes (unsigned char * buf, int size) [virtual]`

Modifies the byte array by a random sequence of bytes generated by this random number generator.

Parameters

<i>buf</i>	non-null array to contain the new random bytes
------------	--

See also

next (p. 2231)

Exceptions

<i>NullPointerException</i>	if <i>buf</i> is NULL
<i>IllegalArgumentException</i>	if <i>size</i> is negative

Reimplemented in **decaf::security::SecureRandom** (p. 2324).

6.427.3.5 double decaf::util::Random::nextDouble ()

Generates a normally distributed random double number between 0.0 inclusively and 1.0 exclusively.

Returns

double

See also

nextFloat (p. 2233)

6.427.3.6 float decaf::util::Random::nextFloat ()

Generates a normally distributed random float number between 0.0 inclusively and 1.0 exclusively.

Returns

float a random float number between 0.0 and 1.0

See also

nextDouble (p. 2233)

6.427.3.7 double decaf::util::Random::nextGaussian ()

Pseudo-randomly generates (approximately) a normally distributed `double` value with mean 0.0 and a standard deviation value of 1.0 using the *polar method* of G.

E. P. Box, M. E. Muller, and G. Marsaglia, as described by Donald E. Knuth in *The Art of Computer Programming, Volume 2: Seminumerical Algorithms*, section 3.4.1, subsection C, algorithm P

Returns

double

See also

nextDouble (p. 2233)

6.427.3.8 int decaf::util::Random::nextInt ()

Generates a uniformly distributed 32-bit `int` value from the this random number sequence.

Returns

`int` uniformly distributed `int` value

See also

next (p. 2231)

nextLong (p. 2234)

6.427.3.9 int decaf::util::Random::nextInt (int n)

Returns to the caller a new pseudo-random integer value which is uniformly distributed between 0 (inclusively) and the value of `n` (exclusively).

Parameters

<code>n</code>	The <code>int</code> value that defines the max value of the return.
----------------	--

Returns

the next pseudo random `int` value.

Exceptions

<i>IllegalArgumentException</i>	if <code>n</code> is less than or equal to zero.
---------------------------------	--

6.427.3.10 long long decaf::util::Random::nextLong ()

Generates a uniformly distributed 64-bit `int` value from the this random number sequence.

Returns

64-bit `int` random number

See also

next (p. 2231)

nextInt() (p. 2234)

nextInt(int) (p. 2234)

6.427.3.11 virtual void **decaf::util::Random::setSeed** (unsigned long long *seed*)
[virtual]

Modifies the seed using linear congruential formula presented in *The Art of Computer Programming, Volume 2*, Section 3.2.1.

Parameters

<i>seed</i>	the seed that alters the state of the random number generator
-------------	---

See also

next (p. 2231)
Random() (p. 2230)
#Random(long)

Reimplemented in **decaf::security::SecureRandom** (p. 2324).

The documentation for this class was generated from the following file:

- src/main/decaf/util/**Random.h**

6.428 decaf::lang::Readable Class Reference

A **Readable** (p. 2235) is a source of characters.

```
#include <src/main/decaf/lang/Readable.h>
```

Inheritance diagram for decaf::lang::Readable:

Public Member Functions

- virtual **~Readable** ()
- virtual int **read** (**decaf::nio::CharBuffer** *charBuffer)=0
Attempts to read characters into the specified character buffer.

6.428.1 Detailed Description

A **Readable** (p. 2235) is a source of characters.

Characters from a **Readable** (p. 2235) are made available to callers of the read method via a CharBuffer.

Since

1.0

6.428.2 Constructor & Destructor Documentation

6.428.2.1 `virtual decaf::lang::Readable::~Readable () [inline, virtual]`

6.428.3 Member Function Documentation

6.428.3.1 `virtual int decaf::lang::Readable::read (decaf::nio::CharBuffer * charBuffer) [pure virtual]`

Attempts to read characters into the specified character buffer.

The buffer is used as a repository of characters as-is: the only changes made are the results of a put operation. No flipping or rewinding of the buffer is performed.

Parameters

<i>charBuffer</i>	The Buffer to read Characters into.
-------------------	-------------------------------------

Returns

The number of char values added to the buffer, or -1 if this source of characters is at its end

Exceptions

<i>IOException</i>	if an I/O error occurs.
<i>NullPointerException</i>	if buffer is NULL.
<i>ReadOnlyBuffer-Exception</i>	if <i>charBuffer</i> is a read only buffer.

Implemented in **decaf::io::Reader** (p. 2242).

The documentation for this class was generated from the following file:

- `src/main/decaf/lang/Readable.h`

6.429 activemq::transport::inactivity::ReadChecker Class Reference

Runnable class that is used by the {.

```
#include <src/main/activemq/transport/inactivity/Read-Checker.h>
```

Inheritance diagram for `activemq::transport::inactivity::ReadChecker`:

Public Member Functions

- **ReadChecker** (**InactivityMonitor** *parent)
- virtual ~**ReadChecker** ()
- virtual void **run** ()

Run method - called by the Thread class in the context of the thread.

6.429.1 Detailed Description

Runnable class that is used by the {.

See also

InactivityMonitor (p. 1425)} class the check for timeouts related to **transport** (p. 71) reads.

Since

3.1

6.429.2 Constructor & Destructor Documentation

6.429.2.1 **activemq::transport::inactivity::ReadChecker::ReadChecker** (**InactivityMonitor** *
parent)

6.429.2.2 virtual **activemq::transport::inactivity::ReadChecker::~~ReadChecker** ()
[virtual]

6.429.3 Member Function Documentation

6.429.3.1 virtual void **activemq::transport::inactivity::ReadChecker::run** ()
[virtual]

Run method - called by the Thread class in the context of the thread.

Implements **decaf::lang::Runnable** (p. 2312).

The documentation for this class was generated from the following file:

- src/main/activemq/transport/inactivity/**ReadChecker.h**

6.430 decaf::io::Reader Class Reference

```
#include <src/main/decaf/io/Reader.h>
```

Inheritance diagram for decaf::io::Reader:

Public Member Functions

- virtual **~Reader** ()
- virtual void **mark** (int readAheadLimit)
Marks the present position in the stream.
- virtual bool **markSupported** () const
*Tells whether this stream supports the **mark()** (p. 2239) operation.*
- virtual bool **ready** () const
Tells whether this stream is ready to be read.
- virtual void **reset** ()
Resets the stream.
- virtual long long **skip** (long long count)
Skips characters.
- virtual int **read** (std::vector< char > &buffer)
Reads characters into an array.
- virtual int **read** (char *buffer, int size)
Reads characters into an array, the method will attempt to read as much data as the size of the array.
- virtual int **read** (char *buffer, int size, int offset, int length)
Reads characters into a portion of an array.
- virtual int **read** ()
Reads a single character.
- virtual int **read** (decaf::nio::CharBuffer *charBuffer)
Attempts to read characters into the specified character buffer.

Protected Member Functions

- **Reader** ()
- virtual int **doReadArrayBounded** (char *buffer, int size, int offset, int length)=0
Override this method to customize the functionality of the method read(unsigned char buffer, int size, int offset, int length).*
- virtual int **doReadVector** (std::vector< char > &buffer)
Override this method to customize the functionality of the method read(std::vector<char> & buffer) (p. 2240).
- virtual int **doReadArray** (char *buffer, int length)
Override this method to customize the functionality of the method read(char buffer, std::size_t length).*
- virtual int **doReadChar** ()
Override this method to customize the functionality of the method read() (p. 2242).
- virtual int **doReadCharBuffer** (decaf::nio::CharBuffer *charBuffer)
Override this method to customize the functionality of the method read(CharBuffer charBuffer).*

6.430.1 Constructor & Destructor Documentation

6.430.1.1 `decaf::io::Reader::Reader ()` [protected]

6.430.1.2 `virtual decaf::io::Reader::~~Reader ()` [virtual]

6.430.2 Member Function Documentation

6.430.2.1 `virtual int decaf::io::Reader::doReadArray (char * buffer, int length)`
[protected, virtual]

Override this method to customize the functionality of the method `read(char* buffer, std::size_t length)`.

6.430.2.2 `virtual int decaf::io::Reader::doReadArrayBounded (char * buffer, int size, int offset, int length)` [protected, pure virtual]

Override this method to customize the functionality of the method `read(unsigned char* buffer, int size, int offset, int length)`.

All subclasses must override this method to provide the basic **Reader** (p. 2237) functionality.

Implemented in **decaf::io::InputStreamReader** (p. 1476).

6.430.2.3 `virtual int decaf::io::Reader::doReadChar ()` [protected, virtual]

Override this method to customize the functionality of the method **read()** (p. 2242).

6.430.2.4 `virtual int decaf::io::Reader::doReadCharBuffer (decaf::nio::CharBuffer * charBuffer)` [protected, virtual]

Override this method to customize the functionality of the method `read(CharBuffer* charBuffer)`.

6.430.2.5 `virtual int decaf::io::Reader::doReadVector (std::vector< char > & buffer)`
[protected, virtual]

Override this method to customize the functionality of the method **read(std::vector<char>& buffer)** (p. 2240).

6.430.2.6 `virtual void decaf::io::Reader::mark (int readAheadLimit)` [virtual]

Marks the present position in the stream.

Subsequent calls to **reset()** (p. 2243) will attempt to reposition the stream to this point. Not all character-input streams support the **mark()** (p. 2239) operation.

Parameters

<i>readAhead-Limit</i>	Limit on the number of characters that may be read while still preserving the mark. After reading this many characters, attempting to reset the stream may fail.
------------------------	--

Exceptions

<i>IOException</i> (p. 1545)	if an I/O error occurs, or the stream does not support mark.
--	--

6.430.2.7 `virtual bool decaf::io::Reader::markSupported () const [inline, virtual]`

Tells whether this stream supports the **mark()** (p. 2239) operation.

The default implementation always returns false. Subclasses should override this method.

Returns

true if and only if this stream supports the mark operation.

6.430.2.8 `virtual int decaf::io::Reader::read (std::vector< char > & buffer) [virtual]`

Reads characters into an array.

This method will block until some input is available, an I/O error occurs, or the end of the stream is reached.

Parameters

<i>buffer</i>	The buffer to read characters into.
---------------	-------------------------------------

Returns

The number of characters read, or -1 if the end of the stream has been reached

Exceptions

<i>IOException</i> (p. 1545)	thrown if an I/O error occurs.
--	--------------------------------

6.430.2.9 virtual int decaf::io::Reader::read (char * *buffer*, int *size*) [virtual]

Reads characters into an array, the method will attempt to read as much data as the size of the array.

This method will block until some input is available, an I/O error occurs, or the end of the stream is reached.

Parameters

<i>buffer</i>	The target char buffer.
<i>size</i>	The size in bytes of the target buffer.

Returns

The number of bytes read or -1 if the end of stream is reached.

Exceptions

<i>IOException</i> (p. 1545)	thrown if an I/O error occurs.
<i>NullPointerException</i>	if buffer is NULL.

6.430.2.10 virtual int decaf::io::Reader::read (char * *buffer*, int *size*, int *offset*, int *length*) [virtual]

Reads characters into a portion of an array.

This method will block until some input is available, an I/O error occurs, or the end of the stream is reached.

Parameters

<i>buffer</i>	The target char buffer.
<i>size</i>	The size in bytes of the target buffer.
<i>offset</i>	The position in the buffer to start filling.
<i>length</i>	The maximum number of bytes to read.

Returns

The number of bytes read or -1 if the end of stream is reached.

Exceptions

<i>IOException</i> (p. 1545)	thrown if an I/O error occurs.
<i>NullPointerException</i>	if buffer is NULL.
<i>IndexOutOfBoundsException</i>	if the offset + length is greater than the array size.

6.430.2.11 `virtual int decaf::io::Reader::read ()` [virtual]

Reads a single character.

This method will block until a character is available, an I/O error occurs, or the end of the stream is reached.

Subclasses that intend to support efficient single-character input should override this method.

Returns

The character read, as an integer in the range 0 to 65535 (0x00-0xffff), or -1 if the end of the stream has been reached.

Exceptions

<i>IOException</i> (p. 1545)	thrown if an I/O error occurs.
--	--------------------------------

6.430.2.12 `virtual int decaf::io::Reader::read (decaf::nio::CharBuffer * charBuffer)`
[virtual]

Attempts to read characters into the specified character buffer.

The buffer is used as a repository of characters as-is: the only changes made are the results of a put operation. No flipping or rewinding of the buffer is performed.

Parameters

<i>charBuffer</i>	The Buffer to read Characters into.
-------------------	-------------------------------------

Returns

The number of char values added to the buffer, or -1 if this source of characters is at its end

Exceptions

<i>IOException</i> (p. 1545)	if an I/O error occurs.
<i>NullPointerException</i>	if buffer is NULL.
<i>ReadOnlyBuffer-Exception</i>	if charBuffer is a read only buffer.

Implements **decaf::lang::Readable** (p. 2236).

6.430.2.13 `virtual bool decaf::io::Reader::ready () const` `[virtual]`

Tells whether this stream is ready to be read.

Returns

True if the next **read()** (p. 2242) is guaranteed not to block for input, false otherwise.
Note that returning false does not guarantee that the next read will block.

Exceptions

<i>IOException</i> (p. 1545)	if an I/O error occurs.
--	-------------------------

Reimplemented in **decaf::io::InputStreamReader** (p. 1477).

6.430.2.14 `virtual void decaf::io::Reader::reset ()` `[virtual]`

Resets the stream.

If the stream has been marked, then attempt to reposition it at the mark. If the stream has not been marked, then attempt to reset it in some way appropriate to the particular stream, for example by repositioning it to its starting point. Not all character-input streams support the **reset()** (p. 2243) operation, and some support **reset()** (p. 2243) without supporting **mark()** (p. 2239).

Exceptions

<i>IOException</i> (p. 1545)	if an I/O error occurs.
--	-------------------------

6.430.2.15 `virtual long long decaf::io::Reader::skip (long long count)` `[virtual]`

Skips characters.

This method will block until some characters are available, an I/O error occurs, or the end of the stream is reached.

Parameters

<i>count</i>	The number of character to skip.
--------------	----------------------------------

Returns

the number of Character actually skipped.

Exceptions

<i>IOException</i> if an I/O error occurs. (p. 1545)
--

The documentation for this class was generated from the following file:

- src/main/decaf/io/**Reader.h**

6.431 decaf::nio::ReadOnlyBufferException Class Reference

```
#include <src/main/decaf/nio/ReadOnlyBufferException.h>
```

Inheritance diagram for decaf::nio::ReadOnlyBufferException:

Public Member Functions

- **ReadOnlyBufferException** () throw ()
Default Constructor.
- **ReadOnlyBufferException** (const lang::Exception &ex) throw ()
Copy Constructor.
- **ReadOnlyBufferException** (const ReadOnlyBufferException &ex) throw ()
Copy Constructor.
- **ReadOnlyBufferException** (const char *file, const int lineNumber, const std::exception *cause, const char *msg,...) throw ()
Constructor - Initializes the file name and line number where this message occurred.
- **ReadOnlyBufferException** (const std::exception *cause) throw ()
Constructor.
- **ReadOnlyBufferException** (const char *file, const int lineNumber, const char *msg,...) throw ()
Constructor.
- virtual **ReadOnlyBufferException * clone** () const
Clones this exception.
- virtual ~**ReadOnlyBufferException** () throw ()

6.431.1 Constructor & Destructor Documentation

6.431.1.1 decaf::nio::ReadOnlyBufferException::ReadOnlyBufferException ()
throw () [inline]

Default Constructor.

6.431.1.2 **decaf::nio::ReadOnlyBufferException::ReadOnlyBufferException (**
const lang::Exception & ex) throw () `[inline]`

Copy Constructor.

Parameters

<i>ex</i>	the exception to copy
-----------	-----------------------

6.431.1.3 **decaf::nio::ReadOnlyBufferException::ReadOnlyBufferException (**
const ReadOnlyBufferException & ex) throw () `[inline]`

Copy Constructor.

Parameters

<i>ex</i>	the exception to copy, which is an instance of this type
-----------	--

6.431.1.4 **decaf::nio::ReadOnlyBufferException::ReadOnlyBufferException (**
const char * file, const int lineNumber, const std::exception * cause, const char * msg, ...) throw () `[inline]`

Constructor - Initializes the file name and line number where this message occurred.

Sets the message to report, using an optional list of arguments to parse into the message

Parameters

<i>file</i>	The file name where exception occurs
<i>lineNumber</i>	The line number where the exception occurred.
<i>cause</i>	The exception that was the cause for this one to be thrown.
<i>msg</i>	The message to report
<i>...</i>	list of primitives that are formatted into the message

6.431.1.5 **decaf::nio::ReadOnlyBufferException::ReadOnlyBufferException (**
const std::exception * cause) throw () `[inline]`

Constructor.

Parameters

<i>cause</i>	Pointer to the exception that caused this one to be thrown, the object is cloned caller retains ownership.
--------------	--

6.431.1.6 **decaf::nio::ReadOnlyBufferException::ReadOnlyBufferException**
 (const char * *file*, const int *lineNumber*, const char * *msg*, ...) throw ()
 [inline]

Constructor.

Parameters

<i>file</i>	The file name where exception occurs
<i>lineNumber</i>	The line number where the exception occurred.
<i>msg</i>	The message to report
...	list of primitives that are formatted into the message

6.431.1.7 **virtual decaf::nio::ReadOnlyBufferException::~~ReadOnlyBufferException** () throw () [inline, virtual]

6.431.2 Member Function Documentation

6.431.2.1 **virtual ReadOnlyBufferException* decaf::nio::ReadOnlyBufferException::clone** () const [inline, virtual]

Clones this exception.

This is useful for cases where you need to preserve the type of the original exception as well as the message. All subclasses should override.

Reimplemented from **decaf::lang::exceptions::UnsupportedOperationException** (p. 2827).

The documentation for this class was generated from the following file:

- src/main/decaf/nio/**ReadOnlyBufferException.h**

6.432 decaf::util::concurrent::locks::ReadWriteLock Class - Reference

A **ReadWriteLock** (p. 2246) maintains a pair of associated locks, one for read-only operations and one for writing.

```
#include <src/main/decaf/util/concurrent/locks/ReadWriteLock.h>
```

Public Member Functions

- virtual **~ReadWriteLock** ()

- virtual **Lock & readLock** ()=0
Returns the lock used for reading.
- virtual **Lock & writeLock** ()=0
Returns the lock used for writing.

6.432.1 Detailed Description

A **ReadWriteLock** (p. 2246) maintains a pair of associated locks, one for read-only operations and one for writing.

The read lock may be held simultaneously by multiple reader threads, so long as there are no writers. The write lock is exclusive.

All **ReadWriteLock** (p. 2246) implementations must guarantee that the memory synchronization effects of writeLock operations (as specified in the **Lock** (p. 1684) interface) also hold with respect to the associated readLock. That is, a thread successfully acquiring the read lock will see all updates made upon previous release of the write lock.

A read-write lock allows for a greater level of concurrency in accessing shared data than that permitted by a mutual exclusion lock. It exploits the fact that while only a single thread at a time (a writer thread) can modify the shared data, in many cases any number of threads can concurrently read the data (hence reader threads). In theory, the increase in concurrency permitted by the use of a read-write lock will lead to performance improvements over the use of a mutual exclusion lock. In practice this increase in concurrency will only be fully realized on a multi-processor, and then only if the access patterns for the shared data are suitable.

Whether or not a read-write lock will improve performance over the use of a mutual exclusion lock depends on the frequency that the data is read compared to being modified, the duration of the read and write operations, and the contention for the data - that is, the number of threads that will try to read or write the data at the same time. For example, a collection that is initially populated with data and thereafter infrequently modified, while being frequently searched (such as a directory of some kind) is an ideal candidate for the use of a read-write lock. However, if updates become frequent then the data spends most of its time being exclusively locked and there is little, if any increase in concurrency. Further, if the read operations are too short the overhead of the read-write lock implementation (which is inherently more complex than a mutual exclusion lock) can dominate the execution cost, particularly as many read-write lock implementations still serialize all threads through a small section of code. Ultimately, only profiling and measurement will establish whether the use of a read-write lock is suitable for your application.

Although the basic operation of a read-write lock is straight-forward, there are many policy decisions that an implementation must make, which may affect the effectiveness of the read-write lock in a given application. Examples of these policies include:

- * Determining whether to grant the read lock or the write lock, when both readers and writers are waiting, at the time that a writer releases the write lock. Writer preference is common, as writes are expected to be short and infrequent. Reader preference is less common as it can lead to lengthy delays for a write if the readers are frequent and long-

lived as expected. Fair, or "in-order" implementations are also possible. * Determining whether readers that request the read lock while a reader is active and a writer is waiting, are granted the read lock. Preference to the reader can delay the writer indefinitely, while preference to the writer can reduce the potential for concurrency. * Determining whether the locks are reentrant: can a thread with the write lock reacquire it? Can it acquire a read lock while holding the write lock? Is the read lock itself reentrant? * Can the write lock be downgraded to a read lock without allowing an intervening writer? Can a read lock be upgraded to a write lock, in preference to other waiting readers or writers?

You should consider all of these things when evaluating the suitability of a given implementation for your application.

Since

1.0

6.432.2 Constructor & Destructor Documentation

6.432.2.1 `virtual decaf::util::concurrent::locks::ReadWriteLock::~~ReadWriteLock () [inline, virtual]`

6.432.3 Member Function Documentation

6.432.3.1 `virtual Lock& decaf::util::concurrent::locks::ReadWriteLock::readLock () [pure virtual]`

Returns the lock used for reading.

Returns

the lock used for reading.

6.432.3.2 `virtual Lock& decaf::util::concurrent::locks::ReadWriteLock::writeLock () [pure virtual]`

Returns the lock used for writing.

Returns

the lock used for writing.

The documentation for this class was generated from the following file:

- `src/main/decaf/util/concurrent/locks/ReadWriteLock.h`

6.433 activemq::cmsutil::CmsTemplate::ReceiveExecutor Class Reference

```
#include <src/main/activemq/cmsutil/CmsTemplate.h>
```

Inheritance diagram for activemq::cmsutil::CmsTemplate::ReceiveExecutor:

Public Member Functions

- **ReceiveExecutor** (**CmsTemplate** *parent, **cms::Destination** *destination, const std::string &selector, bool noLocal)
- virtual ~**ReceiveExecutor** () throw ()
- virtual void **doInCms** (**cms::Session** *session)
Execute any number of operations against the supplied CMS session.
- virtual **cms::Destination** * **getDestination** (**cms::Session** *session AMQCPP-_UNUSED)
- **cms::Message** * **getMessage** ()

Protected Attributes

- **cms::Destination** * destination
- std::string selector
- bool noLocal
- **cms::Message** * message
- **CmsTemplate** * parent

6.433.1 Constructor & Destructor Documentation

6.433.1.1 **activemq::cmsutil::CmsTemplate::ReceiveExecutor::ReceiveExecutor** (**CmsTemplate** * parent, **cms::Destination** * destination, const std::string & selector, bool noLocal) [inline]

6.433.1.2 virtual **activemq::cmsutil::CmsTemplate::ReceiveExecutor::~ReceiveExecutor** () throw () [inline, virtual]

6.433.2 Member Function Documentation

6.433.2.1 virtual void **activemq::cmsutil::CmsTemplate::ReceiveExecutor::doInCms** (**cms::Session** * session) [virtual]

Execute any number of operations against the supplied CMS session.

Parameters

<i>session</i>	the CMS Session
----------------	-----------------

Exceptions

<i>CMSException</i>	if thrown by CMS API methods
---------------------	------------------------------

Implements **activemq::cmsutil::SessionCallback** (p. 2377).

6.433.2.2 **virtual cms::Destination* activemq::cmsutil::CmsTemplate::ReceiveExecutor::getDestination (cms::Session *session *AMQCPP_UNUSED*)**
[inline, virtual]

6.433.2.3 **cms::Message* activemq::cmsutil::CmsTemplate::ReceiveExecutor::getMessage ()** [inline]

6.433.3 Field Documentation

6.433.3.1 **cms::Destination* activemq::cmsutil::CmsTemplate::ReceiveExecutor::destination** [protected]

6.433.3.2 **cms::Message* activemq::cmsutil::CmsTemplate::ReceiveExecutor::message** [protected]

6.433.3.3 **bool activemq::cmsutil::CmsTemplate::ReceiveExecutor::noLocal**
[protected]

6.433.3.4 **CmsTemplate* activemq::cmsutil::CmsTemplate::ReceiveExecutor::parent** [protected]

6.433.3.5 **std::string activemq::cmsutil::CmsTemplate::ReceiveExecutor::selector**
[protected]

The documentation for this class was generated from the following file:

- src/main/activemq/cmsutil/**CmsTemplate.h**

6.434 **activemq::core::RedeliveryPolicy Class Reference**

Interface for a **RedeliveryPolicy** (p. 2250) object that controls how message Redelivery is handled in ActiveMQ-CPP when a transaction is rolled back.

```
#include <src/main/activemq/core/RedeliveryPolicy.h>
```

Inheritance diagram for **activemq::core::RedeliveryPolicy**:

Public Member Functions

- virtual `~RedeliveryPolicy ()`
- virtual double `getBackOffMultiplier ()` const =0
- virtual void `setBackOffMultiplier (double value)=0`
Sets the Back-Off Multiplier for Message Redelivery.
- virtual short `getCollisionAvoidancePercent ()` const =0
- virtual void `setCollisionAvoidancePercent (short value)=0`
- virtual long long `getInitialRedeliveryDelay ()` const =0
Gets the initial time that redelivery of messages is delayed.
- virtual void `setInitialRedeliveryDelay (long long value)=0`
Sets the initial time that redelivery will be delayed.
- virtual long long `getRedeliveryDelay ()` const =0
Gets the time that redelivery of messages is delayed.
- virtual void `setRedeliveryDelay (long long value)=0`
Sets the time that redelivery will be delayed.
- virtual int `getMaximumRedeliveries ()` const =0
Gets the Maximum number of allowed redeliveries for a message before it will be discarded by the consumer.
- virtual void `setMaximumRedeliveries (int maximumRedeliveries)=0`
Sets the Maximum allowable redeliveries for a Message.
- virtual long long `getNextRedeliveryDelay (long long previousDelay)=0`
Given the last used redelivery delay calculate the next value of the delay based on the settings in this Policy instance.
- virtual bool `isUseCollisionAvoidance ()` const =0
- virtual void `setUseCollisionAvoidance (bool value)=0`
- virtual bool `isUseExponentialBackOff ()` const =0
- virtual void `setUseExponentialBackOff (bool value)=0`
- virtual `RedeliveryPolicy * clone ()` const =0
Create a copy of this Policy and return it.
- virtual void `configure (const decaf::util::Properties &properties)`
Checks the supplied properties object for properties matching the configurable settings of this class.

Static Public Attributes

- static const long long `NO_MAXIMUM_REDELIVERIES`

Protected Member Functions

- `RedeliveryPolicy ()`

6.434.1 Detailed Description

Interface for a **RedeliveryPolicy** (p. 2250) object that controls how message Redelivery is handled in ActiveMQ-CPP when a transaction is rolled back.

Since

3.2.0

6.434.2 Constructor & Destructor Documentation

6.434.2.1 `activemq::core::RedeliveryPolicy::RedeliveryPolicy ()` [protected]

6.434.2.2 `virtual activemq::core::RedeliveryPolicy::~~RedeliveryPolicy ()`
[virtual]

6.434.3 Member Function Documentation

6.434.3.1 `virtual RedeliveryPolicy* activemq::core::RedeliveryPolicy::clone ()`
`const` [pure virtual]

Create a copy of this Policy and return it.

Returns

pointer to a new **RedeliveryPolicy** (p. 2250) that is a copy of this one.

Implemented in `activemq::core::policies::DefaultRedeliveryPolicy` (p. 1151).

6.434.3.2 `virtual void activemq::core::RedeliveryPolicy::configure (const`
`decaf::util::Properties & properties)` [virtual]

Checks the supplied properties object for properties matching the configurable settings of this class.

The default implementation looks for properties named with the prefix `cms.RedeliveryPolicy.XXX` where XXX is the name of a property with a public setter method. For instance `cms.RedeliveryPolicy.useExponentialBackOff` will be used to set the value of the use exponential back off toggle.

Subclasses can override this method to add more configuration options or to exclude certain parameters from being set via the properties object.

Parameters

<i>properties</i>	The Properties object used to configure this object.
-------------------	--

Exceptions

<i>NumberFormatException</i>	if a property that is numeric cannot be converted
<i>IllegalArgument-Exception</i>	if a property can't be converted to the correct type.

6.434.3.3 **virtual double** **activemq::core::RedeliveryPolicy::getBackOffMultiplier ()**
const [pure virtual]

Returns

The value of the Back-Off Multiplier for Message Redelivery.

Implemented in **activemq::core::policies::DefaultRedeliveryPolicy** (p. 1151).

6.434.3.4 **virtual short** **activemq::core::RedeliveryPolicy::getCollisionAvoidancePercent ()** **const** [pure virtual]

Returns

the currently set Collision Avoidance percentage.

Implemented in **activemq::core::policies::DefaultRedeliveryPolicy** (p. 1151).

6.434.3.5 **virtual long long** **activemq::core::RedeliveryPolicy::getInitialRedeliveryDelay ()** **const** [pure virtual]

Gets the initial time that redelivery of messages is delayed.

Returns

the time in milliseconds that redelivery is delayed initially.

Implemented in **activemq::core::policies::DefaultRedeliveryPolicy** (p. 1152).

6.434.3.6 **virtual int** **activemq::core::RedeliveryPolicy::getMaximumRedeliveries ()** **const** [pure virtual]

Gets the Maximum number of allowed redeliveries for a message before it will be discarded by the consumer.

Returns

maximum allowed redeliveries for a message.

Implemented in **activemq::core::policies::DefaultRedeliveryPolicy** (p. 1152).

6.434.3.7 `virtual long long activemq::core::RedeliveryPolicy::getNextRedeliveryDelay (long long previousDelay)` [pure virtual]

Given the last used redelivery delay calculate the next value of the delay based on the settings in this Policy instance.

Parameters

<i>previous-Delay</i>	The last delay that was used between message redeliveries.
-----------------------	--

Returns

the new delay to use before attempting another redelivery.

Implemented in **activemq::core::policies::DefaultRedeliveryPolicy** (p. 1152).

6.434.3.8 `virtual long long activemq::core::RedeliveryPolicy::getRedeliveryDelay ()`
`const` [pure virtual]

Gets the time that redelivery of messages is delayed.

Returns

the time in milliseconds that redelivery is delayed.

Implemented in **activemq::core::policies::DefaultRedeliveryPolicy** (p. 1152).

6.434.3.9 `virtual bool activemq::core::RedeliveryPolicy::isUseCollisionAvoidance ()`
`const` [pure virtual]

Returns

whether or not collision avoidance is enabled for this Policy.

Implemented in **activemq::core::policies::DefaultRedeliveryPolicy** (p. 1153).

6.434.3.10 `virtual bool activemq::core::RedeliveryPolicy::isUseExponentialBackOff ()`
`const` [pure virtual]

Returns

whether or not the exponential back off option is enabled.

Implemented in **activemq::core::policies::DefaultRedeliveryPolicy** (p. 1153).

6.434.3.11 virtual void **activemq::core::RedeliveryPolicy::setBackOffMultiplier** (double *value*) [pure virtual]

Sets the Back-Off Multiplier for Message Redelivery.

Parameters

<i>value</i>	The new value for the back-off multiplier.
--------------	--

Implemented in **activemq::core::policies::DefaultRedeliveryPolicy** (p. 1153).

6.434.3.12 virtual void **activemq::core::RedeliveryPolicy::setCollisionAvoidancePercent** (short *value*) [pure virtual]

Parameters

<i>value</i>	The collision avoidance percentage setting.
--------------	---

Implemented in **activemq::core::policies::DefaultRedeliveryPolicy** (p. 1153).

6.434.3.13 virtual void **activemq::core::RedeliveryPolicy::setInitialRedeliveryDelay** (long long *value*) [pure virtual]

Sets the initial time that redelivery will be delayed.

Parameters

<i>value</i>	Time in Milliseconds to wait before starting redelivery.
--------------	--

Implemented in **activemq::core::policies::DefaultRedeliveryPolicy** (p. 1154).

6.434.3.14 virtual void **activemq::core::RedeliveryPolicy::setMaximumRedeliveries** (int *maximumRedeliveries*) [pure virtual]

Sets the Maximum allowable redeliveries for a Message.

Parameters

<i>maximum-Redeliveries</i>	The maximum number of times that a message will be redelivered.
-----------------------------	---

Implemented in **activemq::core::policies::DefaultRedeliveryPolicy** (p. 1154).

6.434.3.15 virtual void **activemq::core::RedeliveryPolicy::setRedeliveryDelay** (long *value*) [pure virtual]

Sets the time that redelivery will be delayed.

Parameters

<i>value</i>	Time in Milliseconds to wait before the next redelivery.
--------------	--

Implemented in **activemq::core::policies::DefaultRedeliveryPolicy** (p. 1154).

6.434.3.16 `virtual void activemq::core::RedeliveryPolicy::setUseCollisionAvoidance (bool value) [pure virtual]`

Parameters

<i>value</i>	Enable or Disable collision avoidance for this Policy.
--------------	--

Implemented in **activemq::core::policies::DefaultRedeliveryPolicy** (p. 1154).

6.434.3.17 `virtual void activemq::core::RedeliveryPolicy::setUseExponentialBackOff (bool value) [pure virtual]`

Parameters

<i>value</i>	Enable or Disable the exponential back off multiplier option.
--------------	---

Implemented in **activemq::core::policies::DefaultRedeliveryPolicy** (p. 1155).

6.434.4 Field Documentation

6.434.4.1 `const long long activemq::core::RedeliveryPolicy::NO_MAXIMUM_REDELIVERIES [static]`

The documentation for this class was generated from the following file:

- `src/main/activemq/core/RedeliveryPolicy.h`

6.435 decaf::util::concurrent::locks::ReentrantLock Class Reference

A reentrant mutual exclusion **Lock** (p. 1684) with extended capabilities.

```
#include <src/main/decaf/util/concurrent/locks/ReentrantLock.h>
```

Inheritance diagram for `decaf::util::concurrent::locks::ReentrantLock`:

Public Member Functions

- **ReentrantLock** ()

- virtual `~ReentrantLock ()`
- virtual void `lock ()`
Acquires the lock.
- virtual void `lockInterruptibly ()`
Acquires the lock unless the current thread is interrupted.
- virtual bool `tryLock ()`
Acquires the lock only if it is not held by another thread at the time of invocation.
- virtual bool `tryLock (long long time, const TimeUnit &unit)`
Acquires the lock if it is not held by another thread within the given waiting time and the current thread has not been interrupted.
- virtual void `unlock ()`
Attempts to release this lock.
- virtual `Condition * newCondition ()`
*Returns a **Condition** (p. 921) instance for use with this **Lock** (p. 1684) instance.*
- int `getHoldCount ()` const
Queries the number of holds on this lock by the current thread.
- bool `isHeldByCurrentThread ()` const
Queries if this lock is held by the current thread.
- bool `isLocked ()` const
Queries if this lock is held by any thread.
- bool `isFair ()` const
Returns true if this lock has fairness set true.
- std::string `toString ()` const
Returns a string identifying this lock, as well as its lock state.

6.435.1 Detailed Description

A reentrant mutual exclusion **Lock** (p. 1684) with extended capabilities.

A **ReentrantLock** (p. 2256) is owned by the thread last successfully locking, but not yet unlocking it. A thread invoking `lock` will return, successfully acquiring the lock, when the lock is not owned by another thread. The method will return immediately if the current thread already owns the lock. This can be checked using methods `isHeldByCurrentThread()` (p. 2259), and `getHoldCount()` (p. 2258).

The constructor for this class accepts an optional fairness parameter. When set true, under contention, locks favor granting access to the longest-waiting thread. Otherwise this lock does not guarantee any particular access order. Programs using fair locks accessed by many threads may display lower overall throughput (i.e., are slower; often much slower) than those using the default setting, but have smaller variances in times to obtain locks and guarantee lack of starvation. Note however, that fairness of locks does not guarantee fairness of thread scheduling. Thus, one of many threads using a fair lock may obtain it multiple times in succession while other active threads are not progressing and not currently holding the lock. Also note that the untimed `tryLock` method does not honor the fairness setting. It will succeed if the lock is available even if other threads are waiting.

It is recommended practice to always immediately follow a call to lock with a try block, most typically in a before/after construction such as:

```
class X { private:
```

```
    ReentrantLock (p. 2256) lock; // ...
```

```
public:
```

```
void m() { lock.lock(); // block until condition holds
```

```
try { // ... method body } finally { lock.unlock() } } }
```

In addition to implementing the **Lock** (p. 1684) interface, this class defines methods is-Locked and getLockQueueLength, as well as some associated protected access methods that may be useful for instrumentation and monitoring.

Since

1.0

6.435.2 Constructor & Destructor Documentation

6.435.2.1 `decaf::util::concurrent::locks::ReentrantLock::ReentrantLock ()`

6.435.2.2 `virtual decaf::util::concurrent::locks::ReentrantLock::~~ReentrantLock () [virtual]`

6.435.3 Member Function Documentation

6.435.3.1 `int decaf::util::concurrent::locks::ReentrantLock::getHoldCount () const`

Queries the number of holds on this lock by the current thread.

A thread has a hold on a lock for each lock action that is not matched by an unlock action.

The hold count information is typically only used for testing and debugging purposes. For example, if a certain section of code should not be entered with the lock already held then we can assert that fact:

```
class X { private:
```

```
    ReentrantLock (p. 2256) lock; // ...
```

```
public:
```

```
void m() { assert( lock.getHoldCount() == 0 ); lock.lock(); try { // ... method body }
catch(...) { lock.unlock(); } } }
```

Returns

the number of holds on this lock by the current thread, or zero if this lock is not held by the current thread

6.435.3.2 bool decaf::util::concurrent::locks::ReentrantLock::isFair () const

Returns true if this lock has fairness set true.

Returns

true if this lock has fairness set true

6.435.3.3 bool decaf::util::concurrent::locks::ReentrantLock::isHeldByCurrentThread () const

Queries if this lock is held by the current thread.

This method is typically used for debugging and testing. For example, a method that should only be called while a lock is held can assert that this is the case:

```
class X { private: ReentrantLock (p. 2256) lock; // ...  
public: void m() { assert( lock.isHeldByCurrentThread() ); // ... method body } }
```

It can also be used to ensure that a reentrant lock is used in a non-reentrant manner, for example:

```
class X { private: ReentrantLock (p. 2256) lock; // ...  
public: void m() { assert !lock.isHeldByCurrentThread() (p. 2259); lock.lock(); try { // ...  
method body } finally { lock.unlock(); } } }
```

Returns

true if current thread holds this lock and false otherwise

6.435.3.4 bool decaf::util::concurrent::locks::ReentrantLock::isLocked () const

Queries if this lock is held by any thread.

This method is designed for use in monitoring of the system state, not for synchronization control.

Returns

true if any thread holds this lock and false otherwise

6.435.3.5 virtual void decaf::util::concurrent::locks::ReentrantLock::lock () [virtual]

Acquires the lock.

Acquires the lock if it is not held by another thread and returns immediately, setting the lock hold count to one.

If the current thread already holds the lock then the hold count is incremented by one and the method returns immediately.

If the lock is held by another thread then the current thread becomes disabled for thread scheduling purposes and lies dormant until the lock has been acquired, at which time the lock hold count is set to one.

Exceptions

<i>RuntimeException</i>	if an error occurs while acquiring the lock.
-------------------------	--

Implements **decaf::util::concurrent::locks::Lock** (p. 1686).

6.435.3.6 **virtual void decaf::util::concurrent::locks::ReentrantLock::lock-Interruptibly ()** [virtual]

Acquires the lock unless the current thread is interrupted.

Acquires the lock if it is not held by another thread and returns immediately, setting the lock hold count to one.

If the current thread already holds this lock then the hold count is incremented by one and the method returns immediately.

If the lock is held by another thread then the current thread becomes disabled for thread scheduling purposes and lies dormant until one of two things happens:

* The lock is acquired by the current thread; or * Some other thread interrupts the current thread.

If the lock is acquired by the current thread then the lock hold count is set to one.

If the current thread:

* has its interrupted status set on entry to this method; or * is interrupted while acquiring the lock,

then InterruptedException is thrown and the current thread's interrupted status is cleared.

In this implementation, as this method is an explicit interruption point, preference is given to responding to the interrupt over normal or reentrant acquisition of the lock.

Exceptions

<i>RuntimeException</i>	if an error occurs while acquiring the lock.
-------------------------	--

<i>InterruptedException</i>	if the current thread is interrupted while acquiring the lock (and interruption of lock acquisition is supported).
-----------------------------	--

Implements **decaf::util::concurrent::locks::Lock** (p. 1686).

6.435.3.7 virtual **Condition*** decaf::util::concurrent::locks::ReentrantLock::new-Condition () [virtual]

Returns a **Condition** (p. 921) instance for use with this **Lock** (p. 1684) instance.

The returned **Condition** (p. 921) instance supports the same usages as do the **Mutex** (p. 1960) Class' methods (wait, notify, and notifyAll).

* If this lock is not held when any of the **Condition** (p. 921) waiting or signalling methods are called, then an `IllegalMonitorStateException` is thrown. * When the condition waiting methods are called the lock is released and, before they return, the lock is reacquired and the lock hold count restored to what it was when the method was called. * If a thread is interrupted while waiting then the wait will terminate, an `InterruptedException` will be thrown, and the thread's interrupted status will be cleared. * Waiting threads are signaled in FIFO order. * The ordering of lock reacquisition for threads returning from waiting methods is the same as for threads initially acquiring the lock, which is in the default case not specified, but for fair locks favors those threads that have been waiting the longest.

Exceptions

<i>RuntimeException</i>	if an error occurs while creating the Condition (p. 921).
<i>UnsupportedOperationException</i>	if this Lock (p. 1684) implementation does not support conditions

Implements **decaf::util::concurrent::locks::Lock** (p. 1687).

6.435.3.8 std::string decaf::util::concurrent::locks::ReentrantLock::toString () const

Returns a string identifying this lock, as well as its lock state.

The state, in brackets, includes either the String "Unlocked" or the String "Locked by" followed by the name of the owning thread.

Returns

a string identifying this lock, as well as its lock state

6.435.3.9 virtual bool decaf::util::concurrent::locks::ReentrantLock::tryLock () [virtual]

Acquires the lock only if it is not held by another thread at the time of invocation.

Acquires the lock if it is not held by another thread and returns immediately with the value true, setting the lock hold count to one. Even when this lock has been set to use a fair ordering policy, a call to **tryLock()** (p. 2261) will immediately acquire the lock if it is available, whether or not other threads are currently waiting for the lock. This "barging" behavior can be useful in certain circumstances, even though it breaks fairness. If you

want to honor the fairness setting for this lock, then use `tryLock(0, TimeUnit.SECONDS (p. 2764))` which is almost equivalent (it also detects interruption).

If the current thread already holds this lock then the hold count is incremented by one and the method returns true.

If the lock is held by another thread then this method will return immediately with the value false.

Returns

true if the lock was acquired and false otherwise

Exceptions

<i>RuntimeException</i>	if an error occurs while acquiring the lock.
-------------------------	--

Implements **decaf::util::concurrent::locks::Lock** (p. 1687).

6.435.3.10 `virtual bool decaf::util::concurrent::locks::ReentrantLock::tryLock (long long time, const TimeUnit & unit) [virtual]`

Acquires the lock if it is not held by another thread within the given waiting time and the current thread has not been interrupted.

Acquires the lock if it is not held by another thread and returns immediately with the value true, setting the lock hold count to one. If this lock has been set to use a fair ordering policy then an available lock will not be acquired if any other threads are waiting for the lock. This is in contrast to the **tryLock()** (p. 2261) method. If you want a timed `tryLock` that does permit barging on a fair lock then combine the timed and un-timed forms together:

```
if (lock.tryLock() || lock.tryLock(timeout, unit) ) { ... }
```

If the current thread already holds this lock then the hold count is incremented by one and the method returns true.

If the lock is held by another thread then the current thread becomes disabled for thread scheduling purposes and lies dormant until one of three things happens:

- * The lock is acquired by the current thread; or
- * Some other thread interrupts the current thread; or
- * The specified waiting time elapses

If the lock is acquired then the value true is returned and the lock hold count is set to one.

If the current thread:

- * has its interrupted status set on entry to this method; or
- * is interrupted while acquiring the lock,

then `InterruptedException` is thrown and the current thread's interrupted status is cleared.

If the specified waiting time elapses then the value false is returned. If the time is less

6.436 decaf::util::concurrent::RejectedExecutionException Class Reference 2269

than or equal to zero, the method will not wait at all.

In this implementation, as this method is an explicit interruption point, preference is given to responding to the interrupt over normal or reentrant acquisition of the lock, and over reporting the elapse of the waiting time.

Parameters

<i>time</i>	the maximum time to wait for the lock
<i>unit</i>	the time unit of the time argument

Returns

true if the lock was acquired and false if the waiting time elapsed before the lock was acquired

Exceptions

<i>RuntimeException</i>	if an error occurs while acquiring the lock.
<i>InterruptedException</i>	if the current thread is interrupted while acquiring the lock (and interruption of lock acquisition is supported)

Implements **decaf::util::concurrent::locks::Lock** (p. 1688).

6.435.3.11 virtual void **decaf::util::concurrent::locks::ReentrantLock::unlock** ()
[virtual]

Attempts to release this lock.

If the current thread is the holder of this lock then the hold count is decremented. If the hold count is now zero then the lock is released. If the current thread is not the holder of this lock then *IllegalMonitorStateException* is thrown.

Exceptions

<i>RuntimeException</i>	if an error occurs while acquiring the lock.
-------------------------	--

Implements **decaf::util::concurrent::locks::Lock** (p. 1689).

The documentation for this class was generated from the following file:

- src/main/decaf/util/concurrent/locks/**ReentrantLock.h**

6.436 decaf::util::concurrent::RejectedExecutionException Class - Reference

```
#include <src/main/decaf/util/concurrent/RejectedExecution-Exception.h>
```

Inheritance diagram for `decaf::util::concurrent::RejectedExecutionException`:

Public Member Functions

- **RejectedExecutionException** () throw ()
Default Constructor.
- **RejectedExecutionException** (const **Exception** &ex) throw ()
Conversion Constructor from some other Exception.
- **RejectedExecutionException** (const **RejectedExecutionException** &ex) throw ()
Copy Constructor.
- **RejectedExecutionException** (const std::exception ***cause**) throw ()
Constructor.
- **RejectedExecutionException** (const char *file, const int lineNumber, const char *msg,...) throw ()
Constructor - Initializes the file name and line number where this message occurred.
- **RejectedExecutionException** (const char *file, const int lineNumber, const std::exception ***cause**, const char *msg,...) throw ()
Constructor - Initializes the file name and line number where this message occurred.
- virtual **RejectedExecutionException** * **clone** () const
Clones this exception.
- virtual ~**RejectedExecutionException** () throw ()

6.436.1 Constructor & Destructor Documentation

6.436.1.1 **decaf::util::concurrent::RejectedExecutionException::RejectedExecutionException** () throw () `[inline]`

Default Constructor.

6.436.1.2 **decaf::util::concurrent::RejectedExecutionException::RejectedExecutionException** (const **Exception** & ex) throw () `[inline]`

Conversion Constructor from some other Exception.

Parameters

ex	An exception that should become this type of Exception
-----------	--

6.436 decaf::util::concurrent::RejectedExecutionException Class Reference 2271

6.436.1.3 **decaf::util::concurrent::RejectedExecutionException::Rejected-
ExecutionException** (const RejectedExecutionException & *ex*) throw ()
[inline]

Copy Constructor.

Parameters

<i>ex</i>	- The Exception to copy in this new instance.
-----------	---

6.436.1.4 **decaf::util::concurrent::RejectedExecutionException::Rejected-
ExecutionException** (const std::exception * *cause*) throw ()
[inline]

Constructor.

Parameters

<i>cause</i>	Pointer to the exception that caused this one to be thrown, the object is cloned caller retains ownership.
--------------	--

6.436.1.5 **decaf::util::concurrent::RejectedExecutionException::Rejected-
ExecutionException** (const char * *file*, const int *lineNumber*, const char * *msg*,
...) throw () [inline]

Constructor - Initializes the file name and line number where this message occurred.

Sets the message to report, using an optional list of arguments to parse into the message

Parameters

<i>file</i>	- The file name where exception occurs
<i>lineNumber</i>	- The line number where the exception occurred.
<i>msg</i>	- The message to report
...	- list of primitives that are formatted into the message

6.436.1.6 **decaf::util::concurrent::RejectedExecutionException::Rejected-
ExecutionException** (const char * *file*, const int *lineNumber*, const
std::exception * *cause*, const char * *msg*, ...) throw () [inline]

Constructor - Initializes the file name and line number where this message occurred.

Sets the message to report, using an optional list of arguments to parse into the message

Parameters

<i>file</i>	- The file name where exception occurs
<i>lineNumber</i>	- The line number where the exception occurred.
<i>cause</i>	- The exception that was the cause for this one to be thrown.
<i>msg</i>	- The message to report
<i>...</i>	- list of primitives that are formatted into the message

6.436.1.7 `virtual decaf::util::concurrent::RejectedExecutionException::~~RejectedExecutionException () throw () [inline, virtual]`

6.436.2 Member Function Documentation

6.436.2.1 `virtual RejectedExecutionException* decaf::util::concurrent::RejectedExecutionException::clone () const [inline, virtual]`

Clones this exception.

This is useful for cases where you need to preserve the type of the original exception as well as the message. All subclasses should override.

Returns

A new instance this exception type with a copy the current state.

Reimplemented from **decaf::lang::Exception** (p. 1282).

The documentation for this class was generated from the following file:

- `src/main/decaf/util/concurrent/RejectedExecutionException.h`

6.437 decaf::util::concurrent::RejectedExecutionHandler Class - Reference

A handler for tasks that cannot be executed by a **ThreadPoolExecutor** (p. 2715).

```
#include <src/main/decaf/util/concurrent/RejectedExecutionHandler.h>
```

Inheritance diagram for `decaf::util::concurrent::RejectedExecutionHandler`:

Public Member Functions

- **RejectedExecutionHandler** ()

6.437 decaf::util::concurrent::RejectedExecutionHandler Class Reference 2273

- virtual `~RejectedExecutionHandler()`
- virtual void `rejectedExecution (decaf::lang::Runnable *r, ThreadPoolExecutor *executer)=0`

*Method that may be invoked by a **ThreadPoolExecutor** (p. 2715) when **execute** (p. 2724) cannot accept a task.*

6.437.1 Detailed Description

A handler for tasks that cannot be executed by a **ThreadPoolExecutor** (p. 2715).

Since

1.0

6.437.2 Constructor & Destructor Documentation

6.437.2.1 `decaf::util::concurrent::RejectedExecutionHandler::RejectedExecutionHandler()`

6.437.2.2 `virtual decaf::util::concurrent::RejectedExecutionHandler::~~RejectedExecutionHandler()` [virtual]

6.437.3 Member Function Documentation

6.437.3.1 `virtual void decaf::util::concurrent::RejectedExecutionHandler::rejectedExecution (decaf::lang::Runnable * r, ThreadPoolExecutor * executer)`
[pure virtual]

Method that may be invoked by a **ThreadPoolExecutor** (p. 2715) when **execute** (p. 2724) cannot accept a task.

This may occur when no more threads or queue slots are available because their bounds would be exceeded, or upon shutdown of the **Executor** (p. 1297).

In the absence of other alternatives, the method may throw an **RejectedExecutionException** (p. 2263), which will be propagated to the caller of **execute** (p. 2724).

Parameters

<i>r</i>	The pointer to the runnable task requested to be executed.
<i>executer</i>	The pointer to the executor attempting to execute this task.

Exceptions

RejectedExecutionException (p. 2263)	if there is no remedy.
--	------------------------

Implemented in **decaf::util::concurrent::ThreadPoolExecutor::CallerRunsPolicy::DiscardOldestPolicy** (p. 1225).

The documentation for this class was generated from the following file:

- src/main/decaf/util/concurrent/**RejectedExecutionHandler.h**

6.438 activemq::commands::RemoveInfo Class Reference

```
#include <src/main/activemq/commands/RemoveInfo.h>
```

Inheritance diagram for activemq::commands::RemoveInfo:

Public Member Functions

- **RemoveInfo** ()
- virtual **~RemoveInfo** ()
- virtual unsigned char **getDataStructureType** () const
*Get the **DataStructure** (p. 1133) Type as defined in CommandTypes.h.*
- virtual **RemoveInfo** * **cloneDataStructure** () const
Clone this object and return a new instance that the caller now owns, this will be an exact copy of this one.
- virtual void **copyDataStructure** (const **DataStructure** *src)
Copy the contents of the passed object into this objects members, overwriting any existing data.
- virtual std::string **toString** () const
*Returns a string containing the information for this **DataStructure** (p. 1133) such as its type and value of its elements.*
- virtual bool **equals** (const **DataStructure** *value) const
*Compares the **DataStructure** (p. 1133) passed in to this one, and returns if they are equivalent.*
- virtual const **Pointer** < **DataStructure** > & **getObjectId** () const
- virtual **Pointer**< **DataStructure** > & **getObjectId** ()
- virtual void **setObjectId** (const **Pointer**< **DataStructure** > &objectId)
- virtual long long **getLastDeliveredSequenceId** () const
- virtual void **setLastDeliveredSequenceId** (long long lastDeliveredSequenceId)
- virtual bool **isRemoveInfo** () const
- virtual **Pointer**< **Command** > **visit** (activemq::state::CommandVisitor *visitor)
Allows a Visitor to visit this command and return a response to the command based on the command type being visited.

Static Public Attributes

- static const unsigned char **ID_REMOVEINFO** = 12

Protected Attributes

- `Pointer< DataStructure >` **objectId**
- long long **lastDeliveredSequenceId**

6.438.1 Constructor & Destructor Documentation

6.438.1.1 `activemq::commands::RemoveInfo::RemoveInfo ()`

6.438.1.2 `virtual activemq::commands::RemoveInfo::~~RemoveInfo ()`
[virtual]

6.438.2 Member Function Documentation

6.438.2.1 `virtual RemoveInfo* activemq::commands::RemoveInfo::cloneData-
Structure () const` [virtual]

Clone this obbject and return a new instance that the caller now owns, this will be an exact copy of this one.

Returns

new copy of this object.

Implements `activemq::commands::DataStructure` (p. 1133).

6.438.2.2 `virtual void activemq::commands::RemoveInfo::copyDataStructure (`
`const DataStructure * src)` [virtual]

Copy the contents of the passed object into this objects members, overwriting any existing data.

Returns

src - Source Object

Reimplemented from `activemq::commands::BaseCommand` (p. 493).

6.438.2.3 `virtual bool activemq::commands::RemoveInfo::equals (const`
`DataStructure * value) const` [virtual]

Compares the `DataStructure` (p. 1133) passed in to this one, and returns if they are equivalent.

Equivalent here means that they are of the same type, and that each element of the objects are the same.

Returns

true if **DataStructure** (p. 1133)'s are Equal.

Reimplemented from **activemq::commands::BaseCommand** (p. 494).

6.438.2.4 **virtual unsigned char activemq::commands::RemoveInfo::getDataStructureType () const** [virtual]

Get the **DataStructure** (p. 1133) Type as defined in CommandTypes.h.

Returns

The type of the data structure

Implements **activemq::commands::DataStructure** (p. 1137).

6.438.2.5 **virtual long long activemq::commands::RemoveInfo::getLastDeliveredSequenceld () const** [virtual]

6.438.2.6 **virtual const Pointer<DataStructure>& activemq::commands::RemoveInfo::getObjectId () const** [virtual]

Referenced by **activemq::state::CommandVisitorAdapter::processRemoveInfo()**.

6.438.2.7 **virtual Pointer<DataStructure>& activemq::commands::RemoveInfo::getObjectId ()** [virtual]

6.438.2.8 **virtual bool activemq::commands::RemoveInfo::isRemoveInfo () const** [inline, virtual]

Returns

an answer of true to the **isRemoveInfo()** (p. 2270) query.

Reimplemented from **activemq::commands::BaseCommand** (p. 496).

6.438.2.9 **virtual void activemq::commands::RemoveInfo::setLastDeliveredSequenceld (long long lastDeliveredSequenceld)** [virtual]

6.438.2.10 **virtual void activemq::commands::RemoveInfo::setObjectId (const Pointer< DataStructure > & objectId)** [virtual]

6.439

activemq::wireformat::openwire::marshal::generated::RemoveInfoMarshaller

Class Reference

2277

6.438.2.11 `virtual std::string activemq::commands::RemoveInfo::toString () const`
[virtual]

Returns a string containing the information for this **DataSet** (p. 1133) such as its type and value of its elements.

Returns

formatted string useful for debugging.

Reimplemented from **activemq::commands::BaseCommand** (p. 498).

6.438.2.12 `virtual Pointer<Command> activemq::commands::RemoveInfo::visit (`
`activemq::state::CommandVisitor * visitor)` [virtual]

Allows a Visitor to visit this command and return a response to the command based on the command type being visited.

The command will call the proper processXXX method in the visitor.

Returns

a **Response** (p. 2298) to the visitor being called or NULL if no response.

Implements **activemq::commands::Command** (p. 871).

6.438.3 Field Documentation

6.438.3.1 `const unsigned char activemq::commands::RemoveInfo::ID_REMOVEINFO`
`= 12` [static]

6.438.3.2 `long long activemq::commands::RemoveInfo::lastDeliveredSequenceId`
[protected]

6.438.3.3 `Pointer<DataSet> activemq::commands::RemoveInfo::objectId`
[protected]

The documentation for this class was generated from the following file:

- src/main/activemq/commands/**RemoveInfo.h**

6.439 activemq::wireformat::openwire::marshal::generated::-

RemoveInfoMarshaller Class Reference

Marshaling code for Open Wire Format for **RemoveInfoMarshaller** (p. 2271).

```
#include <src/main/activemq/wireformat/openwire/marshal/generated/-
RemoveInfoMarshaller.h>
```

Inheritance diagram for `activemq::wireformat::openwire::marshal::generated::RemoveInfoMarshaller`:

Public Member Functions

- **RemoveInfoMarshaller** ()
- virtual **~RemoveInfoMarshaller** ()
- virtual **commands::DataStructure * createObject** () const
Creates a new instance of the class that this class is a marshaling director for.
- virtual unsigned char **getDataStructureType** () const
Gets the DataStructureType that this class marshals/unmarshals.
- virtual void **tightUnmarshal** (**OpenWireFormat *wireFormat**, **commands::DataStructure *dataStructure**, **decaf::io::DataInputStream *dataIn**, **utils::BooleanStream *bs**)
Tight Un-marhsal to the given stream.
- virtual int **tightMarshal1** (**OpenWireFormat *wireFormat**, **commands::DataStructure *dataStructure**, **utils::BooleanStream *bs**)
Tight Marhsal to the given stream.
- virtual void **tightMarshal2** (**OpenWireFormat *wireFormat**, **commands::DataStructure *dataStructure**, **decaf::io::DataOutputStream *dataOut**, **utils::BooleanStream *bs**)
Tight Marhsal to the given stream.
- virtual void **looseUnmarshal** (**OpenWireFormat *wireFormat**, **commands::DataStructure *dataStructure**, **decaf::io::DataInputStream *dataIn**)
Loose Un-marhsal to the given stream.
- virtual void **looseMarshal** (**OpenWireFormat *wireFormat**, **commands::DataStructure *dataStructure**, **decaf::io::DataOutputStream *dataOut**)
Tight Marhsal to the given stream.

6.439.1 Detailed Description

Marshaling code for Open Wire Format for **RemoveInfoMarshaller** (p. 2271).

NOTE!: This file is auto generated - do not modify! if you need to make a change, please see the Java Classes in the `activemq-openwire-generator` module

6.439.2 Constructor & Destructor Documentation

- 6.439.2.1 **activemq::wireformat::openwire::marshal::generated::RemoveInfoMarshaller::RemoveInfoMarshaller** ()
[inline]

6.439

activemq::wireformat::openwire::marshal::generated::RemoveInfoMarshaller

Class Reference

2279

6.439.2.2 `virtual activemq::wireformat::openwire::marshal::generated::RemoveInfoMarshaller::~~RemoveInfoMarshaller () [inline, virtual]`

6.439.3 Member Function Documentation

6.439.3.1 `virtual commands::DataStructure* activemq::wireformat::openwire::marshal::generated::RemoveInfoMarshaller::createObject () const [virtual]`

Creates a new instance of the class that this class is a marshaling director for.

Returns

newly allocated Command

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1120).

6.439.3.2 `virtual unsigned char activemq::wireformat::openwire::marshal::generated::RemoveInfoMarshaller::getDataStructureType () const [virtual]`

Gets the DataStructureType that this class marshals/unmarshals.

Returns

byte Id of this classes DataStructureType

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1122).

6.439.3.3 `virtual void activemq::wireformat::openwire::marshal::generated::RemoveInfoMarshaller::looseMarshal (OpenWireFormat * format, commands::DataStructure * command, decaf::io::DataOutputStream * ds) [virtual]`

Tight Marshal to the given stream.

Parameters

<i>format</i>	- The OpenWireFormat properties
<i>command</i>	- the object to Marshal
<i>ds</i>	- DataOutputStream to marshal to

Exceptions

<i>IOException</i>	if an error occurs.
--------------------	---------------------

Reimplemented from **activemq::wireformat::openwire::marshal::generated::Base-CommandMarshaller** (p. 500).

6.439.3.4 `virtual void activemq::wireformat::openwire::marshal::generated::RemoveInfoMarshaller::looseUnmarshal (OpenWireFormat * format, commands::DataStructure * command, decaf::io::DataInputStream * dis)`
`[virtual]`

Loose Un-marhsal to the given stream.

Parameters

<i>format</i>	- The OpenWireFormat properties
<i>command</i>	- the object to Un-Marshal
<i>dis</i>	- the DataInputStream to Un-Marshal from

Exceptions

<i>IOException</i>	if an error occurs.
--------------------	---------------------

Reimplemented from **activemq::wireformat::openwire::marshal::generated::Base-CommandMarshaller** (p. 501).

6.439.3.5 `virtual int activemq::wireformat::openwire::marshal::generated::RemoveInfoMarshaller::tightMarshal1 (OpenWireFormat * format, commands::DataStructure * command, utils::BooleanStream * bs)`
`[virtual]`

Tight Marhsal to the given stream.

Parameters

<i>format</i>	The OpenWireFormat properties
<i>command</i>	The object to Marshal
<i>bs</i>	The boolean stream to marshal to.

Exceptions

<i>IOException</i>	if an error occurs.
--------------------	---------------------

Reimplemented from **activemq::wireformat::openwire::marshal::generated::Base-CommandMarshaller** (p. 503).

6.439

activemq::wireformat::openwire::marshal::generated::RemoveInfoMarshaller

Class Reference

2281

6.439.3.6 virtual void **activemq::wireformat::openwire::marshal::generated::RemoveInfoMarshaller::tightMarshal2** (**OpenWireFormat** * *format*, **commands::DataStructure** * *command*, **decaf::io::DataOutputStream** * *ds*, **utils::BooleanStream** * *bs*) [virtual]

Tight Marhsal to the given stream.

Parameters

<i>format</i>	- The OpenWireFormat properties
<i>command</i>	- the object to Marshal
<i>ds</i>	- the DataOutputStream to Marshal to
<i>bs</i>	- boolean stream to marshal to.

Exceptions

<i>IOException</i>	if an error occurs.
--------------------	---------------------

Reimplemented from **activemq::wireformat::openwire::marshal::generated::Base-CommandMarshaller** (p. 504).

6.439.3.7 virtual void **activemq::wireformat::openwire::marshal::generated::RemoveInfoMarshaller::tightUnmarshal** (**OpenWireFormat** * *format*, **commands::DataStructure** * *command*, **decaf::io::DataInputStream** * *dis*, **utils::BooleanStream** * *bs*) [virtual]

Tight Un-marhsal to the given stream.

Parameters

<i>format</i>	- The OpenWireFormat properties
<i>command</i>	- the object to Un-Marshal
<i>dis</i>	- the DataInputStream to Un-Marshal from
<i>bs</i>	- boolean stream to unmarshal from.

Exceptions

<i>IOException</i>	if an error occurs.
--------------------	---------------------

Reimplemented from **activemq::wireformat::openwire::marshal::generated::Base-CommandMarshaller** (p. 505).

The documentation for this class was generated from the following file:

- src/main/activemq/wireformat/openwire/marshal/generated/**RemoveInfoMarshaller.h**

6.440 activemq::commands::RemoveSubscriptionInfo Class - Reference

```
#include <src/main/activemq/commands/RemoveSubscription-Info.h>
```

Inheritance diagram for activemq::commands::RemoveSubscriptionInfo:

Public Member Functions

- **RemoveSubscriptionInfo** ()
- virtual **~RemoveSubscriptionInfo** ()
- virtual unsigned char **getDataStructureType** () const
*Get the **DataStructure** (p. 1133) Type as defined in CommandTypes.h.*
- virtual **RemoveSubscriptionInfo** * **cloneDataStructure** () const
Clone this obbject and return a new instance that the caller now owns, this will be an exact copy of this one.
- virtual void **copyDataStructure** (const **DataStructure** *src)
Copy the contents of the passed object into this objects members, overwriting any existing data.
- virtual std::string **toString** () const
*Returns a string containing the information for this **DataStructure** (p. 1133) such as its type and value of its elements.*
- virtual bool **equals** (const **DataStructure** *value) const
*Compares the **DataStructure** (p. 1133) passed in to this one, and returns if they are equivalent.*
- virtual const **Pointer** < **ConnectionId** > & **getConnectionId** () const
- virtual **Pointer**< **ConnectionId** > & **getConnectionId** ()
- virtual void **setConnectionId** (const **Pointer**< **ConnectionId** > &connectionId)
- virtual const std::string & **getSubscriptionName** () const
- virtual std::string & **getSubscriptionName** ()
- virtual void **setSubscriptionName** (const std::string &subscriptionName)
- virtual const std::string & **getClientId** () const
- virtual std::string & **getClientId** ()
- virtual void **setClientId** (const std::string &clientId)
- virtual bool **isRemoveSubscriptionInfo** () const
- virtual **Pointer**< **Command** > **visit** (activemq::state::CommandVisitor *visitor)
Allows a Visitor to visit this command and return a response to the command based on the command type being visited.

Static Public Attributes

- static const unsigned char **ID_REMOVESUBSCRIPTIONINFO** = 9

Protected Attributes

- **Pointer**< **ConnectionId** > **connectionId**
- std::string **subscriptionName**
- std::string **clientId**

6.440.1 Constructor & Destructor Documentation

6.440.1.1 **activemq::commands::RemoveSubscriptionInfo::RemoveSubscriptionInfo** ()

6.440.1.2 **virtual activemq::commands::RemoveSubscriptionInfo::~~RemoveSubscriptionInfo** () [virtual]

6.440.2 Member Function Documentation

6.440.2.1 **virtual RemoveSubscriptionInfo* activemq::commands::RemoveSubscriptionInfo::cloneDataStructure** () const [virtual]

Clone this obbject and return a new instance that the caller now owns, this will be an exact copy of this one.

Returns

new copy of this object.

Implements **activemq::commands::DataStructure** (p. 1133).

6.440.2.2 **virtual void activemq::commands::RemoveSubscriptionInfo::copyDataStructure** (const **DataStructure** * *src*) [virtual]

Copy the contents of the passed object into this objects members, overwriting any existing data.

Returns

src - Source Object

Reimplemented from **activemq::commands::BaseCommand** (p. 493).

6.440.2.3 **virtual bool activemq::commands::RemoveSubscriptionInfo::equals** (const **DataStructure** * *value*) const [virtual]

Compares the **DataStructure** (p. 1133) passed in to this one, and returns if they are equivalent.

Equivalent here means that they are of the same type, and that each element of the objects are the same.

Returns

true if **DataStructure** (p. 1133)'s are Equal.

Reimplemented from **activemq::commands::BaseCommand** (p. 494).

6.440.2.4 `virtual const std::string& activemq::commands::RemoveSubscriptionInfo::getClientId () const [virtual]`

6.440.2.5 `virtual std::string& activemq::commands::RemoveSubscriptionInfo::getClientId () [virtual]`

6.440.2.6 `virtual const Pointer<ConnectionId>& activemq::commands::RemoveSubscriptionInfo::getConnectionId () const [virtual]`

6.440.2.7 `virtual Pointer<ConnectionId>& activemq::commands::RemoveSubscriptionInfo::getConnectionId () [virtual]`

6.440.2.8 `virtual unsigned char activemq::commands::RemoveSubscriptionInfo::getDataStructureType () const [virtual]`

Get the **DataStructure** (p. 1133) Type as defined in CommandTypes.h.

Returns

The type of the data structure

Implements **activemq::commands::DataStructure** (p. 1137).

6.440.2.9 `virtual const std::string& activemq::commands::RemoveSubscriptionInfo::getSubscriptionName () const [virtual]`

6.440.2.10 `virtual std::string& activemq::commands::RemoveSubscriptionInfo::getSubscriptionName () [virtual]`

6.440.2.11 `virtual bool activemq::commands::RemoveSubscriptionInfo::isRemoveSubscriptionInfo () const [inline, virtual]`

Returns

an answer of true to the **isRemoveSubscriptionInfo()** (p. 2278) query.

Reimplemented from **activemq::commands::BaseCommand** (p. 497).

6.440.2.12 `virtual void activemq::commands::RemoveSubscriptionInfo::setClientId (const std::string & clientId) [virtual]`

6.440.2.13 `virtual void activemq::commands::RemoveSubscriptionInfo::set-ConnectionId (const Pointer< ConnectionId > & connectionId) [virtual]`

6.440.2.14 `virtual void activemq::commands::RemoveSubscriptionInfo::setSubscriptionName (const std::string & subscriptionName) [virtual]`

6.440.2.15 `virtual std::string activemq::commands::RemoveSubscriptionInfo::toString () const [virtual]`

Returns a string containing the information for this **DataSet** (p. 1133) such as its type and value of its elements.

Returns

formatted string useful for debugging.

Reimplemented from **activemq::commands::BaseCommand** (p. 498).

6.440.2.16 `virtual Pointer<Command> activemq::commands::RemoveSubscriptionInfo::visit (activemq::state::CommandVisitor * visitor) [virtual]`

Allows a Visitor to visit this command and return a response to the command based on the command type being visited.

The command will call the proper processXXX method in the visitor.

Returns

a **Response** (p. 2298) to the visitor being called or NULL if no response.

Implements **activemq::commands::Command** (p. 871).

6.440.3 Field Documentation

6.440.3.1 `std::string activemq::commands::RemoveSubscriptionInfo::clientId [protected]`

6.440.3.2 `Pointer<ConnectionId> activemq::commands::RemoveSubscriptionInfo::connectionId [protected]`

6.440.3.3 `const unsigned char activemq::commands::RemoveSubscriptionInfo::ID_REMOVE SUBSCRIPTION INFO = 9 [static]`

6.440.3.4 `std::string activemq::commands::RemoveSubscriptionInfo::subscription-Name` [protected]

The documentation for this class was generated from the following file:

- `src/main/activemq/commands/RemoveSubscriptionInfo.h`

6.441 `activemq::wireformat::openwire::marshal::generated::RemoveSubscriptionInfoMarshaller` Class Reference

Marshaling code for Open Wire Format for `RemoveSubscriptionInfoMarshaller` (p. 2280).

```
#include <src/main/activemq/wireformat/openwire/marshal/generated/-
RemoveSubscriptionInfoMarshaller.h>
```

Inheritance diagram for `activemq::wireformat::openwire::marshal::generated::RemoveSubscriptionInfoMarshaller`:

Public Member Functions

- `RemoveSubscriptionInfoMarshaller ()`
- `virtual ~RemoveSubscriptionInfoMarshaller ()`
- `virtual commands::DataStructure * createObject () const`
Creates a new instance of the class that this class is a marshaling director for.
- `virtual unsigned char getDataStructureType () const`
Gets the DataStructureType that this class marshals/unmarshals.
- `virtual void tightUnmarshal (OpenWireFormat *wireFormat, commands::DataStructure *dataStructure, decaf::io::DataInputStream *dataIn, utils::BooleanStream *bs)`
Tight Un-marhsal to the given stream.
- `virtual int tightMarshal1 (OpenWireFormat *wireFormat, commands::DataStructure *dataStructure, utils::BooleanStream *bs)`
Tight Marhsal to the given stream.
- `virtual void tightMarshal2 (OpenWireFormat *wireFormat, commands::DataStructure *dataStructure, decaf::io::DataOutputStream *dataOut, utils::BooleanStream *bs)`
Tight Marhsal to the given stream.
- `virtual void looseUnmarshal (OpenWireFormat *wireFormat, commands::DataStructure *dataStructure, decaf::io::DataInputStream *dataIn)`
Loose Un-marhsal to the given stream.
- `virtual void looseMarshal (OpenWireFormat *wireFormat, commands::DataStructure *dataStructure, decaf::io::DataOutputStream *dataOut)`
Tight Marhsal to the given stream.

6.441 activemq::wireformat::openwire::marshal::generated::RemoveSubscriptionInfoMarshaller Class

Reference

2287

6.441.1 Detailed Description

Marshaling code for Open Wire Format for **RemoveSubscriptionInfoMarshaller** (p.2280).

NOTE!: This file is auto generated - do not modify! if you need to make a change, please see the Java Classes in the activemq-openwire-generator module

6.441.2 Constructor & Destructor Documentation

6.441.2.1 **activemq::wireformat::openwire::marshal::generated::RemoveSubscriptionInfoMarshaller::RemoveSubscriptionInfoMarshaller** ()
[inline]

6.441.2.2 **virtual activemq::wireformat::openwire::marshal::generated::RemoveSubscriptionInfoMarshaller::~~RemoveSubscriptionInfoMarshaller** ()
[inline, virtual]

6.441.3 Member Function Documentation

6.441.3.1 **virtual commands::DataStructure* activemq::wireformat::openwire::marshal::generated::RemoveSubscriptionInfoMarshaller::createObject** () const [virtual]

Creates a new instance of the class that this class is a marshaling director for.

Returns

newly allocated Command

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p.1120).

6.441.3.2 **virtual unsigned char activemq::wireformat::openwire::marshal::generated::RemoveSubscriptionInfoMarshaller::getDataStructureType** () const
[virtual]

Gets the DataStructureType that this class marshals/unmarshals.

Returns

byte Id of this classes DataStructureType

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p.1122).

6.441.3.3 `virtual void activemq::wireformat::openwire::marshal::generated-
::RemoveSubscriptionInfoMarshaller::looseMarshal (`
`OpenWireFormat * format, commands::DataStructure * command,`
`decaf::io::DataOutputStream * ds) [virtual]`

Tight Marshal to the given stream.

Parameters

<i>format</i>	- The OpenWireFormat properties
<i>command</i>	- the object to Marshal
<i>ds</i>	- DataOutputStream to marshal to

Exceptions

<i>IOException</i>	if an error occurs.
--------------------	---------------------

Reimplemented from **activemq::wireformat::openwire::marshal::generated::Base-
CommandMarshaller** (p. 500).

6.441.3.4 `virtual void activemq::wireformat::openwire::marshal::generated-
::RemoveSubscriptionInfoMarshaller::looseUnmarshal (`
`OpenWireFormat * format, commands::DataStructure * command,`
`decaf::io::DataInputStream * dis) [virtual]`

Loose Un-marshal to the given stream.

Parameters

<i>format</i>	- The OpenWireFormat properties
<i>command</i>	- the object to Un-Marshal
<i>dis</i>	- the DataInputStream to Un-Marshal from

Exceptions

<i>IOException</i>	if an error occurs.
--------------------	---------------------

Reimplemented from **activemq::wireformat::openwire::marshal::generated::Base-
CommandMarshaller** (p. 501).

6.441.3.5 `virtual int activemq::wireformat::openwire::marshal::generated::Remove-
SubscriptionInfoMarshaller::tightMarshal1 (OpenWireFormat * format,`
`commands::DataStructure * command, utils::BooleanStream * bs)`
`[virtual]`

Tight Marshal to the given stream.

6.441 activemq::wireformat::openwire::marshal::generated::RemoveSubscriptionInfoMarshaller Class

Reference

2289

Parameters

<i>format</i>	The OpenwireFormat properties
<i>command</i>	The object to Marshal
<i>bs</i>	The boolean stream to marshal to.

Exceptions

<i>IOException</i>	if an error occurs.
--------------------	---------------------

Reimplemented from **activemq::wireformat::openwire::marshal::generated::BaseCommandMarshaller** (p. 503).

```
6.441.3.6 virtual void activemq::wireformat::openwire::marshal::generated-  
::RemoveSubscriptionInfoMarshaller::tightMarshal2 (  
    OpenWireFormat * format, commands::DataStructure * command,  
    decaf::io::DataOutputStream * ds, utils::BooleanStream * bs )  
[virtual]
```

Tight Marhsal to the given stream.

Parameters

<i>format</i>	- The OpenwireFormat properties
<i>command</i>	- the object to Marshal
<i>ds</i>	- the DataOutputStream to Marshal to
<i>bs</i>	- boolean stream to marshal to.

Exceptions

<i>IOException</i>	if an error occurs.
--------------------	---------------------

Reimplemented from **activemq::wireformat::openwire::marshal::generated::BaseCommandMarshaller** (p. 504).

```
6.441.3.7 virtual void activemq::wireformat::openwire::marshal::generated-  
::RemoveSubscriptionInfoMarshaller::tightUnmarshal (  
    OpenWireFormat * format, commands::DataStructure * command,  
    decaf::io::DataInputStream * dis, utils::BooleanStream * bs )  
[virtual]
```

Tight Un-marhsal to the given stream.

Parameters

<i>format</i>	- The OpenwireFormat properties
<i>command</i>	- the object to Un-Marshal
<i>dis</i>	- the DataInputStream to Un-Marshal from
<i>bs</i>	- boolean stream to unmarshal from.

Exceptions

<i>IOException</i> if an error occurs.
--

Reimplemented from **activemq::wireformat::openwire::marshal::generated::Base-CommandMarshaller** (p. 505).

The documentation for this class was generated from the following file:

- `src/main/activemq/wireformat/openwire/marshal/generated/RemoveSubscription-InfoMarshaller.h`

6.442 **activemq::commands::ReplayCommand** Class Reference

```
#include <src/main/activemq/commands/ReplayCommand.h>
```

Inheritance diagram for **activemq::commands::ReplayCommand**:

Public Member Functions

- **ReplayCommand** ()
- virtual **~ReplayCommand** ()
- virtual unsigned char **getDataStructureType** () const
*Get the **DataStructure** (p. 1133) Type as defined in CommandTypes.h.*
- virtual **ReplayCommand** * **cloneDataStructure** () const
Clone this object and return a new instance that the caller now owns, this will be an exact copy of this one.
- virtual void **copyDataStructure** (const **DataStructure** *src)
Copy the contents of the passed object into this objects members, overwriting any existing data.
- virtual std::string **toString** () const
*Returns a string containing the information for this **DataStructure** (p. 1133) such as its type and value of its elements.*
- virtual bool **equals** (const **DataStructure** *value) const
*Compares the **DataStructure** (p. 1133) passed in to this one, and returns if they are equivalent.*
- virtual int **getFirstNakNumber** () const
- virtual void **setFirstNakNumber** (int firstNakNumber)
- virtual int **getLastNakNumber** () const
- virtual void **setLastNakNumber** (int lastNakNumber)
- virtual **Pointer**< **Command** > **visit** (activemq::state::CommandVisitor *visitor)
Allows a Visitor to visit this command and return a response to the command based on the command type being visited.

Static Public Attributes

- static const unsigned char **ID_REPLAYCOMMAND** = 65

Protected Attributes

- int **firstNakNumber**
- int **lastNakNumber**

6.442.1 Constructor & Destructor Documentation

6.442.1.1 `activemq::commands::ReplayCommand::ReplayCommand ()`

6.442.1.2 `virtual activemq::commands::ReplayCommand::~~ReplayCommand ()`
[virtual]

6.442.2 Member Function Documentation

6.442.2.1 `virtual ReplayCommand* activemq::commands::ReplayCommand::cloneDataStructure () const`
[virtual]

Clone this object and return a new instance that the caller now owns, this will be an exact copy of this one.

Returns

new copy of this object.

Implements **activemq::commands::DataStructure** (p. 1133).

6.442.2.2 `virtual void activemq::commands::ReplayCommand::copyDataStructure (const DataStructure * src)` [virtual]

Copy the contents of the passed object into this objects members, overwriting any existing data.

Returns

src - Source Object

Reimplemented from **activemq::commands::BaseCommand** (p. 493).

6.442.2.3 `virtual bool activemq::commands::ReplayCommand::equals (const DataStructure * value) const` [virtual]

Compares the **DataStructure** (p. 1133) passed in to this one, and returns if they are equivalent.

Equivalent here means that they are of the same type, and that each element of the objects are the same.

Returns

true if **DataStructure** (p. 1133)'s are Equal.

Reimplemented from **activemq::commands::BaseCommand** (p. 494).

6.442.2.4 `virtual unsigned char activemq::commands::ReplayCommand::getDataStructureType () const [virtual]`

Get the **DataStructure** (p. 1133) Type as defined in CommandTypes.h.

Returns

The type of the data structure

Implements **activemq::commands::DataStructure** (p. 1137).

6.442.2.5 `virtual int activemq::commands::ReplayCommand::getFirstNakNumber () const [virtual]`

6.442.2.6 `virtual int activemq::commands::ReplayCommand::getLastNakNumber () const [virtual]`

6.442.2.7 `virtual void activemq::commands::ReplayCommand::setFirstNakNumber (int firstNakNumber) [virtual]`

6.442.2.8 `virtual void activemq::commands::ReplayCommand::setLastNakNumber (int lastNakNumber) [virtual]`

6.442.2.9 `virtual std::string activemq::commands::ReplayCommand::toString () const [virtual]`

Returns a string containing the information for this **DataStructure** (p. 1133) such as its type and value of its elements.

Returns

formatted string useful for debugging.

Reimplemented from **activemq::commands::BaseCommand** (p. 498).

6.442.2.10 `virtual Pointer<Command> activemq::commands::ReplayCommand::visit (activemq::state::CommandVisitor * visitor) [virtual]`

Allows a Visitor to visit this command and return a response to the command based on the command type being visited.

6.443 activemq::wireformat::openwire::marshal::generated::ReplayCommandMarshaller Class

Reference

2293

The command will call the proper processXXX method in the visitor.

Returns

a **Response** (p. 2298) to the visitor being called or NULL if no response.

Implements **activemq::commands::Command** (p. 871).

6.442.3 Field Documentation

6.442.3.1 **int activemq::commands::ReplayCommand::firstNakNumber**
[protected]

6.442.3.2 **const unsigned char activemq::commands::ReplayCommand::ID_REPLAYCOMMAND = 65** [static]

6.442.3.3 **int activemq::commands::ReplayCommand::lastNakNumber**
[protected]

The documentation for this class was generated from the following file:

- src/main/activemq/commands/**ReplayCommand.h**

6.443 activemq::wireformat::openwire::marshal::generated::ReplayCommandMarshaller Class Reference

Marshaling code for Open Wire Format for **ReplayCommandMarshaller** (p. 2287).

```
#include <src/main/activemq/wireformat/openwire/marshal/generated/-  
ReplayCommandMarshaller.h>
```

Inheritance diagram for activemq::wireformat::openwire::marshal::generated::ReplayCommandMarshaller:

Public Member Functions

- **ReplayCommandMarshaller ()**
- virtual **~ReplayCommandMarshaller ()**
- virtual **commands::DataStructure * createObject ()** const
Creates a new instance of the class that this class is a marshaling director for.
- virtual unsigned char **getDataStructureType ()** const
Gets the DataStructureType that this class marshals/unmarshals.
- virtual void **tightUnmarshal (OpenWireFormat *wireFormat, commands::DataStructure *dataStructure, decaf::io::DataInputStream *dataIn, utils::BooleanStream *bs)**

Tight Un-marhsal to the given stream.

- virtual int **tightMarshal1** (**OpenWireFormat** *wireFormat, **commands::Data-Structure** *dataStructure, **utils::BooleanStream** *bs)

Tight Marhsal to the given stream.

- virtual void **tightMarshal2** (**OpenWireFormat** *wireFormat, **commands::Data-Structure** *dataStructure, **decaf::io::DataOutputStream** *dataOut, **utils::BooleanStream** *bs)

Tight Marhsal to the given stream.

- virtual void **looseUnmarshal** (**OpenWireFormat** *wireFormat, **commands::Data-Structure** *dataStructure, **decaf::io::DataInputStream** *dataIn)

Loose Un-marhsal to the given stream.

- virtual void **looseMarshal** (**OpenWireFormat** *wireFormat, **commands::Data-Structure** *dataStructure, **decaf::io::DataOutputStream** *dataOut)

Tight Marhsal to the given stream.

6.443.1 Detailed Description

Marshaling code for Open Wire Format for **ReplayCommandMarshaller** (p. 2287).

NOTE!: This file is auto generated - do not modify! if you need to make a change, please see the Java Classes in the activemq-openwire-generator module

6.443.2 Constructor & Destructor Documentation

6.443.2.1 **activemq::wireformat::openwire::marshal::generated::ReplayCommandMarshaller::ReplayCommandMarshaller ()**
[inline]

6.443.2.2 **virtual activemq::wireformat::openwire::marshal::generated::ReplayCommandMarshaller::~~ReplayCommandMarshaller ()** [inline, virtual]

6.443.3 Member Function Documentation

6.443.3.1 **virtual commands::DataStructure* activemq::wireformat::openwire::marshal::generated::ReplayCommandMarshaller::createObject ()**
const [virtual]

Creates a new instance of the class that this class is a marshaling director for.

Returns

newly allocated Command

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1120).

6.443 activemq::wireformat::openwire::marshal::generated::ReplayCommandMarshaller Class

Reference

2295

```
6.443.3.2 virtual unsigned char activemq::wireformat::openwire::marshal::generated::ReplayCommandMarshaller::getDataStructureType ( ) const  
[virtual]
```

Gets the DataStructureType that this class marshals/unmarshals.

Returns

byte Id of this classes DataStructureType

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1122).

```
6.443.3.3 virtual void activemq::wireformat::openwire::marshal::generated::ReplayCommandMarshaller::looseMarshal ( OpenWireFormat * format,  
commands::DataStructure * command, decaf::io::DataOutputStream * ds  
) [virtual]
```

Tight Marhsal to the given stream.

Parameters

<i>format</i>	- The OpenwireFormat properties
<i>command</i>	- the object to Marshal
<i>ds</i>	- DataOutputStream to marshal to

Exceptions

<i>IOException</i>	if an error occurs.
--------------------	---------------------

Reimplemented from **activemq::wireformat::openwire::marshal::generated::BaseCommandMarshaller** (p. 500).

```
6.443.3.4 virtual void activemq::wireformat::openwire::marshal::generated::ReplayCommandMarshaller::looseUnmarshal ( OpenWireFormat * format,  
commands::DataStructure * command, decaf::io::DataInputStream * dis )  
[virtual]
```

Loose Un-marhsal to the given stream.

Parameters

<i>format</i>	- The OpenwireFormat properties
<i>command</i>	- the object to Un-Marshal
<i>dis</i>	- the DataInputStream to Un-Marshal from

Exceptions

<i>IOException</i>	if an error occurs.
--------------------	---------------------

Reimplemented from **activemq::wireformat::openwire::marshal::generated::Base-CommandMarshaller** (p. 501).

6.443.3.5 **virtual int activemq::wireformat::openwire::marshal::generated::Replay-CommandMarshaller::tightMarshal1 (OpenWireFormat * *format*, commands::DataStructure * *command*, utils::BooleanStream * *bs*)**
[virtual]

Tight Marshal to the given stream.

Parameters

<i>format</i>	The OpenWireFormat properties
<i>command</i>	The object to Marshal
<i>bs</i>	The boolean stream to marshal to.

Exceptions

<i>IOException</i>	if an error occurs.
--------------------	---------------------

Reimplemented from **activemq::wireformat::openwire::marshal::generated::Base-CommandMarshaller** (p. 503).

6.443.3.6 **virtual void activemq::wireformat::openwire::marshal::generated::ReplayCommandMarshaller::tightMarshal2 (OpenWireFormat * *format*, commands::DataStructure * *command*, decaf::io::DataOutputStream * *ds*, utils::BooleanStream * *bs*)** [virtual]

Tight Marshal to the given stream.

Parameters

<i>format</i>	- The OpenWireFormat properties
<i>command</i>	- the object to Marshal
<i>ds</i>	- the DataOutputStream to Marshal to
<i>bs</i>	- boolean stream to marshal to.

Exceptions

<i>IOException</i>	if an error occurs.
--------------------	---------------------

Reimplemented from **activemq::wireformat::openwire::marshal::generated::Base-CommandMarshaller** (p. 504).

6.444 activemq::cmsutil::CmsTemplate::ResolveProducerExecutor Class Reference

2297

6.443.3.7 virtual void **activemq::wireformat::openwire::marshal::generated::ReplayCommandMarshaller::tightUnmarshal** (**OpenWireFormat** * *format*, **commands::DataStructure** * *command*, **decaf::io::DataInputStream** * *dis*, **utils::BooleanStream** * *bs*) [virtual]

Tight Un-marhsal to the given stream.

Parameters

<i>format</i>	- The OpenWireFormat properties
<i>command</i>	- the object to Un-Marshal
<i>dis</i>	- the DataInputStream to Un-Marshal from
<i>bs</i>	- boolean stream to unmarshal from.

Exceptions

<i>IOException</i>	if an error occurs.
--------------------	---------------------

Reimplemented from **activemq::wireformat::openwire::marshal::generated::BaseCommandMarshaller** (p. 505).

The documentation for this class was generated from the following file:

- src/main/activemq/wireformat/openwire/marshal/generated/**ReplayCommandMarshaller.h**

6.444 activemq::cmsutil::CmsTemplate::ResolveProducerExecutor Class Reference

```
#include <src/main/activemq/cmsutil/CmsTemplate.h>
```

Inheritance diagram for **activemq::cmsutil::CmsTemplate::ResolveProducerExecutor**:

Public Member Functions

- **ResolveProducerExecutor** (**ProducerCallback** **action*, **CmsTemplate** **parent*, const std::string &*destinationName*)
- virtual ~**ResolveProducerExecutor** () throw ()
- virtual **cms::Destination** * **getDestination** (**cms::Session** **session*)

6.444.1 Constructor & Destructor Documentation

6.444.1.1 `activemq::cmsutil::CmsTemplate::ResolveProducerExecutor::ResolveProducer-
Executor (ProducerCallback * action, CmsTemplate * parent, const
std::string & destinationName) [inline]`

6.444.1.2 `virtual activemq::cmsutil::CmsTemplate::ResolveProducer-
Executor::~~ResolveProducerExecutor () throw () [inline,
virtual]`

6.444.2 Member Function Documentation

6.444.2.1 `virtual cms::Destination* activemq::cmsutil::CmsTemplate::Resolve-
ProducerExecutor::getDestination (cms::Session * session)
[virtual]`

The documentation for this class was generated from the following file:

- `src/main/activemq/cmsutil/CmsTemplate.h`

6.445 `activemq::cmsutil::CmsTemplate::ResolveReceiveExecutor` Class Reference

```
#include <src/main/activemq/cmsutil/CmsTemplate.h>
```

Inheritance diagram for `activemq::cmsutil::CmsTemplate::ResolveReceiveExecutor`:

Public Member Functions

- **ResolveReceiveExecutor** (`CmsTemplate *parent`, `const std::string &selector`, `bool noLocal`, `const std::string &destinationName`)
- virtual **~ResolveReceiveExecutor** () throw ()
- virtual `cms::Destination * getDestination (cms::Session *session)`

6.445.1 Constructor & Destructor Documentation

6.445.1.1 `activemq::cmsutil::CmsTemplate::ResolveReceiveExecutor::ResolveReceiveExecutor
(CmsTemplate * parent, const std::string & selector, bool noLocal, const
std::string & destinationName) [inline]`

6.445.1.2 `virtual activemq::cmsutil::CmsTemplate::ResolveReceive-
Executor::~~ResolveReceiveExecutor () throw () [inline,
virtual]`

6.445.2 Member Function Documentation

6.445.2.1 virtual cms::Destination* activemq::cmsutil::CmsTemplate::ResolveReceiveExecutor::getDestination (cms::Session * session)
[virtual]

The documentation for this class was generated from the following file:

- src/main/activemq/cmsutil/CmsTemplate.h

6.446 decaf::internal::util::Resource Class Reference

Interface for all Managed Resources in Decaf, these objects are added to the Runtime System and are destroyed at shutdown.

```
#include <src/main/decaf/internal/util/Resource.h>
```

Inheritance diagram for decaf::internal::util::Resource:

Public Member Functions

- virtual ~Resource ()

6.446.1 Detailed Description

Interface for all Managed Resources in Decaf, these objects are added to the Runtime System and are destroyed at shutdown.

Since

1.0

6.446.2 Constructor & Destructor Documentation

6.446.2.1 virtual decaf::internal::util::Resource::~~Resource () [virtual]

The documentation for this class was generated from the following file:

- src/main/decaf/internal/util/Resource.h

6.447 activemq::cmsutil::ResourceLifecycleManager Class Reference

Manages the lifecycle of a set of CMS resources.

```
#include <src/main/activemq/cmsutil/ResourceLifecycle-
Manager.h>
```

Public Member Functions

- **ResourceLifecycleManager** ()
- virtual **~ResourceLifecycleManager** ()
Destructor - calls `destroy`
- void **addConnection** (**cms::Connection** *connection)
Adds a connection so that its life will be managed by this object.
- void **addSession** (**cms::Session** *session)
Adds a session so that its life will be managed by this object.
- void **addDestination** (**cms::Destination** *dest)
Adds a destination so that its life will be managed by this object.
- void **addMessageProducer** (**cms::MessageProducer** *producer)
Adds a message producer so that its life will be managed by this object.
- void **addMessageConsumer** (**cms::MessageConsumer** *consumer)
Adds a message consumer so that its life will be managed by this object.
- void **destroy** ()
Closes and destroys the contained CMS resources.
- void **releaseAll** ()
Releases all of the contained resources so that this object will no longer control their lifetimes.

Protected Member Functions

- **ResourceLifecycleManager** (const **ResourceLifecycleManager** &)
- **ResourceLifecycleManager** & **operator=** (const **ResourceLifecycleManager** &)

6.447.1 Detailed Description

Manages the lifecycle of a set of CMS resources.

A call to `destroy` will close and destroy all of the contained resources in the appropriate manner.

6.447.2 Constructor & Destructor Documentation

6.447.2.1 **activemq::cmsutil::ResourceLifecycleManager::ResourceLifecycleManager** (const **ResourceLifecycleManager** &)
[protected]

6.447.2.2 **activemq::cmsutil::ResourceLifecycleManager::ResourceLifecycleManager ()**

6.447.2.3 **virtual activemq::cmsutil::ResourceLifecycleManager::~~ResourceLifecycleManager ()** [virtual]

Destructor - calls `destroy`

6.447.3 Member Function Documentation

6.447.3.1 **void activemq::cmsutil::ResourceLifecycleManager::addConnection (cms::Connection * *connection*)**

Adds a connection so that its life will be managed by this object.

Parameters

<i>connection</i>	the object to be managed
-------------------	--------------------------

Exceptions

<i>CMSEException</i>	if an error occurs while performing this operation.
----------------------	---

6.447.3.2 **void activemq::cmsutil::ResourceLifecycleManager::addDestination (cms::Destination * *dest*)**

Adds a destination so that its life will be managed by this object.

Parameters

<i>dest</i>	the object to be managed
-------------	--------------------------

Exceptions

<i>CMSEException</i>	if an error occurs while performing this operation.
----------------------	---

6.447.3.3 **void activemq::cmsutil::ResourceLifecycleManager::addMessageConsumer (cms::MessageConsumer * *consumer*)**

Adds a message consumer so that its life will be managed by this object.

Parameters

<i>consumer</i>	the object to be managed
-----------------	--------------------------

Exceptions

<i>CMSEException</i>	if an error occurs while performing this operation.
----------------------	---

6.447.3.4 **void activemq::cmsutil::ResourceLifecycleManager::addMessageProducer (cms::MessageProducer * *producer*)**

Adds a message producer so that its life will be managed by this object.

Parameters

<i>producer</i>	the object to be managed
-----------------	--------------------------

Exceptions

<i>CMSEException</i>	if an error occurs while performing this operation.
----------------------	---

6.447.3.5 **void activemq::cmsutil::ResourceLifecycleManager::addSession (cms::Session * *session*)**

Adds a session so that its life will be managed by this object.

Parameters

<i>session</i>	the object to be managed
----------------	--------------------------

Exceptions

<i>CMSEException</i>	if an error occurs while performing this operation.
----------------------	---

6.447.3.6 **void activemq::cmsutil::ResourceLifecycleManager::destroy ()**

Closes and destroys the contained CMS resources.

Exceptions

<i>cms::CMS-Exception</i> (p. 826)	thrown if an error occurs.
--	----------------------------

6.447.3.7 **ResourceLifecycleManager& activemq::cmsutil::ResourceLifecycleManager::operator= (const ResourceLifecycleManager &)**
[protected]

6.447.3.8 void activemq::cmsutil::ResourceLifecycleManager::releaseAll ()

Releases all of the contained resources so that this object will no longer control their lifetimes.

The documentation for this class was generated from the following file:

- src/main/activemq/cmsutil/**ResourceLifecycleManager.h**

6.448 decaf::internal::util::ResourceLifecycleManager Class - Reference

```
#include <src/main/decaf/internal/util/ResourceLifecycle-
Manager.h>
```

Public Member Functions

- **ResourceLifecycleManager** ()
- virtual **~ResourceLifecycleManager** ()
- virtual void **addResource** (**Resource** *value)

Protected Member Functions

- virtual void **destroyResources** ()

6.448.1 Detailed Description

Since

1.0

6.448.2 Constructor & Destructor Documentation

6.448.2.1 decaf::internal::util::ResourceLifecycleManager::ResourceLifecycleManager ()

6.448.2.2 virtual decaf::internal::util::ResourceLifecycleManager::~~ResourceLifecycleManager () [virtual]

6.448.3 Member Function Documentation

6.448.3.1 virtual void decaf::internal::util::ResourceLifecycleManager::addResource (**Resource** * *value*) [virtual]

6.448.3.2 virtual void **decaf::internal::util::ResourceLifecycleManager::destroyResources** () [protected, virtual]

The documentation for this class was generated from the following file:

- src/main/decaf/internal/util/**ResourceLifecycleManager.h**

6.449 activemq::commands::Response Class Reference

```
#include <src/main/activemq/commands/Response.h>
```

Inheritance diagram for activemq::commands::Response:

Public Member Functions

- **Response** ()
- virtual **~Response** ()
- virtual unsigned char **getDataStructureType** () const
*Get the **DataStructure** (p. 1133) Type as defined in CommandTypes.h.*
- virtual **Response * cloneDataStructure** () const
Clone this object and return a new instance that the caller now owns, this will be an exact copy of this one.
- virtual void **copyDataStructure** (const **DataStructure** *src)
Copy the contents of the passed object into this objects members, overwriting any existing data.
- virtual std::string **toString** () const
*Returns a string containing the information for this **DataStructure** (p. 1133) such as its type and value of its elements.*
- virtual bool **equals** (const **DataStructure** *value) const
*Compares the **DataStructure** (p. 1133) passed in to this one, and returns if they are equivalent.*
- virtual int **getCorrelationId** () const
- virtual void **setCorrelationId** (int correlationId)
- virtual bool **isResponse** () const
- virtual **Pointer< Command > visit** (activemq::state::CommandVisitor *visitor)
Allows a Visitor to visit this command and return a response to the command based on the command type being visited.

Static Public Attributes

- static const unsigned char **ID_RESPONSE** = 30

Protected Attributes

- int **correlationId**

6.449.1 Constructor & Destructor Documentation

6.449.1.1 **activemq::commands::Response::Response** ()

6.449.1.2 **virtual activemq::commands::Response::~~Response** () [virtual]

6.449.2 Member Function Documentation

6.449.2.1 **virtual Response* activemq::commands::Response::cloneDataStructure**
() const [virtual]

Clone this object and return a new instance that the caller now owns, this will be an exact copy of this one.

Returns

new copy of this object.

Implements **activemq::commands::DataStructure** (p. 1133).

Reimplemented in **activemq::commands::DataArrayResponse** (p. 1070), **activemq::commands::DataResponse** (p. 1113), **activemq::commands::ExceptionResponse** (p. 1289), and **activemq::commands::IntegerResponse** (p. 1517).

6.449.2.2 **virtual void activemq::commands::Response::copyDataStructure** (const **DataStructure* src**) [virtual]

Copy the contents of the passed object into this objects members, overwriting any existing data.

Returns

src - Source Object

Reimplemented from **activemq::commands::BaseCommand** (p. 493).

Reimplemented in **activemq::commands::DataArrayResponse** (p. 1070), **activemq::commands::DataResponse** (p. 1113), **activemq::commands::ExceptionResponse** (p. 1289), and **activemq::commands::IntegerResponse** (p. 1517).

6.449.2.3 **virtual bool activemq::commands::Response::equals** (const **DataStructure* value**) const [virtual]

Compares the **DataStructure** (p. 1133) passed in to this one, and returns if they are equivalent.

Equivalent here means that they are of the same type, and that each element of the objects are the same.

Returns

true if **DataStructure** (p. 1133)'s are Equal.

Reimplemented from **activemq::commands::BaseCommand** (p. 494).

Reimplemented in **activemq::commands::DataArrayResponse** (p. 1070), **activemq::commands::DataResponse** (p. 1114), **activemq::commands::ExceptionResponse** (p. 1289), and **activemq::commands::IntegerResponse** (p. 1518).

6.449.2.4 `virtual int activemq::commands::Response::getCorrelationId () const`
[virtual]

6.449.2.5 `virtual unsigned char activemq::commands::Response::getDataStructure-`
`Type () const` [virtual]

Get the **DataStructure** (p. 1133) Type as defined in CommandTypes.h.

Returns

The type of the data structure

Implements **activemq::commands::DataStructure** (p. 1137).

Reimplemented in **activemq::commands::DataArrayResponse** (p. 1071), **activemq::commands::DataResponse** (p. 1114), **activemq::commands::ExceptionResponse** (p. 1289), and **activemq::commands::IntegerResponse** (p. 1518).

6.449.2.6 `virtual bool activemq::commands::Response::isResponse () const`
[inline, virtual]

Returns

an answer of true to the **isResponse()** (p. 2300) query.

Reimplemented from **activemq::commands::BaseCommand** (p. 497).

6.449.2.7 `virtual void activemq::commands::Response::setCorrelationId (int`
`correlationId)` [virtual]

6.449.2.8 `virtual std::string activemq::commands::Response::toString () const`
[virtual]

Returns a string containing the information for this **DataStructure** (p. 1133) such as its type and value of its elements.

Returns

formatted string useful for debugging.

Reimplemented from `activemq::commands::BaseCommand` (p. 498).

Reimplemented in `activemq::commands::DataArrayResponse` (p. 1071), `activemq::commands::DataResponse` (p. 1114), `activemq::commands::ExceptionResponse` (p. 1290), and `activemq::commands::IntegerResponse` (p. 1518).

6.449.2.9 `virtual Pointer<Command> activemq::commands::Response::visit (activemq::state::CommandVisitor * visitor)` [virtual]

Allows a Visitor to visit this command and return a response to the command based on the command type being visited.

The command will call the proper processXXX method in the visitor.

Returns

a **Response** (p. 2298) to the visitor being called or NULL if no response.

Implements `activemq::commands::Command` (p. 871).

6.449.3 Field Documentation

6.449.3.1 `int activemq::commands::Response::correlationId` [protected]

6.449.3.2 `const unsigned char activemq::commands::Response::ID_RESPONSE = 30` [static]

The documentation for this class was generated from the following file:

- `src/main/activemq/commands/Response.h`

6.450 `activemq::transport::mock::ResponseBuilder` Class Reference

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

```
#include <src/main/activemq/transport/mock/ResponseBuilder.h>
```

Inheritance diagram for `activemq::transport::mock::ResponseBuilder`:

Public Member Functions

- virtual **~ResponseBuilder** ()
- virtual **Pointer< Response > buildResponse** (const **Pointer< Command >** &command)=0
Given a Command, check if it requires a response and return the appropriate - Response that the Broker would send for this Command.
- virtual void **buildIncomingCommands** (const **Pointer< Command >** &command, **decaf::util::LinkedList< Pointer< Command >** &queue)=0
*When called the **ResponseBuilder** (p. 2301) must construct all the Responses or - Asynchronous commands that would be sent to this client by the Broker upon receipt of the passed command.*

6.450.1 Detailed Description

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

6.450.2 Constructor & Destructor Documentation

- 6.450.2.1 virtual **activemq::transport::mock::ResponseBuilder::~ResponseBuilder** () [inline, virtual]

6.450.3 Member Function Documentation

- 6.450.3.1 virtual void **activemq::transport::mock::ResponseBuilder::buildIncomingCommands** (const **Pointer< Command >** & command, **decaf::util::LinkedList< Pointer< Command >** & queue) [pure virtual]

When called the **ResponseBuilder** (p. 2301) must construct all the Responses or - Asynchronous commands that would be sent to this client by the Broker upon receipt of the passed command.

Parameters

<i>command</i>	- The Command being sent to the Broker.
<i>queue</i>	- Queue of Command sent back from the broker.

Implemented in **activemq::wireformat::openwire::OpenWireResponseBuilder** (p. 2065).

6.451 activemq::transport::correlator::ResponseCorrelator Class Reference 2309

6.450.3.2 virtual **Pointer**<**Response**> **activemq::transport::mock::ResponseBuilder::buildResponse** (const **Pointer**< **Command** > & *command*)
[pure virtual]

Given a **Command**, check if it requires a response and return the appropriate **Response** that the **Broker** would send for this **Command**.

Parameters

<i>command</i>	- The command to build a response for
----------------	---------------------------------------

Returns

A **Response** object pointer, or **NULL** if no response.

Implemented in **activemq::wireformat::openwire::OpenWireResponseBuilder** (p. 2066).

The documentation for this class was generated from the following file:

- src/main/activemq/transport/mock/**ResponseBuilder.h**

6.451 activemq::transport::correlator::ResponseCorrelator Class - Reference

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

```
#include <src/main/activemq/transport/correlator/ResponseCorrelator.h>
```

Inheritance diagram for **activemq::transport::correlator::ResponseCorrelator**:

Public Member Functions

- **ResponseCorrelator** (const **Pointer**< **Transport** > &next)
Constructor.
- virtual ~**ResponseCorrelator** ()
- virtual void **oneway** (const **Pointer**< **Command** > &command)
Sends a one-way command.
- virtual **Pointer**< **Response** > **request** (const **Pointer**< **Command** > &command)
Sends the given command to the broker and then waits for the response.
- virtual **Pointer**< **Response** > **request** (const **Pointer**< **Command** > &command, unsigned int timeout)

Sends the given command to the broker and then waits for the response.

- virtual void **start** ()

*Starts the **Transport** (p. 2790), the send methods of a **Transport** (p. 2790) will throw an exception if used before the **Transport** (p. 2790) is started.*

- virtual void **close** ()

Closes this object and deallocates the appropriate resources.

- virtual void **onCommand** (const **Pointer**< **Command** > &command)

This is called in the context of the nested transport's reading thread.

- virtual void **onTransportException** (**Transport** *source, const **decaf::lang::Exception** &ex)

Event handler for an exception from a command transport.

6.451.1 Detailed Description

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

Non-response messages are simply sent directly to the CommandListener. It owns the transport that it

6.451.2 Constructor & Destructor Documentation

6.451.2.1 **activemq::transport::correlator::ResponseCorrelator::ResponseCorrelator** (const **Pointer**< **Transport** > &next)

Constructor.

Parameters

<i>next</i>	the next transport in the chain
-------------	---------------------------------

6.451.2.2 **virtual activemq::transport::correlator::ResponseCorrelator::~ResponseCorrelator** () [virtual]

6.451.3 Member Function Documentation

6.451.3.1 **virtual void activemq::transport::correlator::ResponseCorrelator::close** () [virtual]

Closes this object and deallocates the appropriate resources.

The object is generally no longer usable after calling close.

6.451 activemq::transport::correlator::ResponseCorrelator Class Reference 2311

Exceptions

<i>IOException</i>	if an error occurs while closing.
--------------------	-----------------------------------

Reimplemented from **activemq::transport::TransportFilter** (p. 2802).

6.451.3.2 `virtual void activemq::transport::correlator::ResponseCorrelator::onCommand (const Pointer< Command > & command)`
[virtual]

This is called in the context of the nested transport's reading thread.

In the case of a response object, updates the request map and notifies those waiting on the response. Non-response messages are just delegated to the command listener.

Parameters

<i>command</i>	The received from the nested transport.
----------------	---

Reimplemented from **activemq::transport::TransportFilter** (p. 2805).

6.451.3.3 `virtual void activemq::transport::correlator::ResponseCorrelator::oneway (const Pointer< Command > & command)` [virtual]

Sends a one-way command.

Does not wait for any response from the broker.

Parameters

<i>command</i>	The command to be sent.
----------------	-------------------------

Exceptions

<i>IOException</i>	if an exception occurs during writing of the command.
<i>UnsupportedOperationException</i>	if this method is not implemented by this transport.

Reimplemented from **activemq::transport::TransportFilter** (p. 2806).

6.451.3.4 `virtual void activemq::transport::correlator::ResponseCorrelator::onTransportException (Transport * source, const decaf::lang::Exception & ex)` [virtual]

Event handler for an exception from a command transport.

Parameters

<i>source</i>	The source of the exception.
<i>ex</i>	The exception that was caught.

6.451.3.5 **virtual Pointer<Response> activemq::transport::correlator::Response-Correlator::request (const Pointer< Command > & *command*)**
 [virtual]

Sends the given command to the broker and then waits for the response.

Parameters

<i>command</i>	the command to be sent.
----------------	-------------------------

Returns

the response from the broker.

Exceptions

<i>IOException</i>	if an exception occurs during the read of the command.
<i>Unsupported-OperationException</i>	if this method is not implemented by this transport.

Reimplemented from **activemq::transport::TransportFilter** (p. 2807).

6.451.3.6 **virtual Pointer<Response> activemq::transport::correlator::Response-Correlator::request (const Pointer< Command > & *command*, unsigned int *timeout*)** [virtual]

Sends the given command to the broker and then waits for the response.

Parameters

<i>command</i>	The command to be sent.
<i>timeout</i>	The time to wait for this response.

Returns

the response from the broker.

Exceptions

<i>IOException</i>	if an exception occurs during the read of the command.
<i>Unsupported-OperationException</i>	if this method is not implemented by this transport.

6.452 activemq::wireformat::openwire::marshal::generated::ResponseMarshaller Class Reference 2313

Reimplemented from **activemq::transport::TransportFilter** (p. 2807).

6.451.3.7 virtual void **activemq::transport::correlator::ResponseCorrelator::start** (
) [virtual]

Starts the **Transport** (p. 2790), the send methods of a **Transport** (p. 2790) will throw an exception if used before the **Transport** (p. 2790) is started.

Exceptions

<i>IOException</i>	if and error occurs while starting the Transport (p. 2790).
--------------------	--

Reimplemented from **activemq::transport::TransportFilter** (p. 2808).

The documentation for this class was generated from the following file:

- src/main/activemq/transport/correlator/**ResponseCorrelator.h**

6.452 activemq::wireformat::openwire::marshal::generated::-ResponseMarshaller Class Reference

Marshaling code for Open Wire Format for **ResponseMarshaller** (p. 2307).

```
#include <src/main/activemq/wireformat/openwire/marshal/generated/-  
ResponseMarshaller.h>
```

Inheritance diagram for **activemq::wireformat::openwire::marshal::generated::-ResponseMarshaller**:

Public Member Functions

- **ResponseMarshaller** ()
- virtual **~ResponseMarshaller** ()
- virtual **commands::DataStructure * createObject** () const
Creates a new instance of the class that this class is a marshaling director for.
- virtual unsigned char **getDataStructureType** () const
Gets the DataStructureType that this class marshals/unmarshals.
- virtual void **tightUnmarshal** (**OpenWireFormat** *wireFormat, **commands::-DataStructure** *dataStructure, **decaf::io::DataInputStream** *dataIn, **utils::-BooleanStream** *bs)
Tight Un-marhsal to the given stream.
- virtual int **tightMarshal1** (**OpenWireFormat** *wireFormat, **commands::Data-Structure** *dataStructure, **utils::BooleanStream** *bs)
Tight Marhsal to the given stream.

- virtual void **tightMarshal2** (**OpenWireFormat** *wireFormat, **commands::Data-Structure** *dataStructure, **decaf::io::DataOutputStream** *dataOut, **utils::BooleanStream** *bs)
Tight Marshal to the given stream.
- virtual void **looseUnmarshal** (**OpenWireFormat** *wireFormat, **commands::Data-Structure** *dataStructure, **decaf::io::DataInputStream** *dataIn)
Loose Un-marshal to the given stream.
- virtual void **looseMarshal** (**OpenWireFormat** *wireFormat, **commands::Data-Structure** *dataStructure, **decaf::io::DataOutputStream** *dataOut)
Tight Marshal to the given stream.

6.452.1 Detailed Description

Marshaling code for Open Wire Format for **ResponseMarshaller** (p. 2307).

NOTE!: This file is auto generated - do not modify! if you need to make a change, please see the Java Classes in the activemq-openwire-generator module

6.452.2 Constructor & Destructor Documentation

- 6.452.2.1 **activemq::wireformat::openwire::marshal::generated::ResponseMarshaller::ResponseMarshaller** ()
[inline]
- 6.452.2.2 **virtual activemq::wireformat::openwire::marshal::generated::ResponseMarshaller::~~ResponseMarshaller** () [inline, virtual]

6.452.3 Member Function Documentation

- 6.452.3.1 **virtual commands::DataStructure* activemq::wireformat::openwire::marshal::generated::ResponseMarshaller::createObject** () const
[virtual]

Creates a new instance of the class that this class is a marshaling director for.

Returns

newly allocated Command

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1120).

Reimplemented in **activemq::wireformat::openwire::marshal::generated::DataArrayResponseMarshaller** (p. 1073), **activemq::wireformat::openwire::marshal::generated::DataResponseMarshaller** (p. 1116), **activemq::wireformat::openwire::marshal::generated::ExceptionResponseMarshaller** (p. 1292), and **activemq::wireformat::openwire::marshal::generated::IntegerResponseMarshaller** (p. 1520).

6.452.3.2 `virtual unsigned char activemq::wireformat::openwire::marshal::generated::ResponseMarshaller::getDataStructureType () const`
`[virtual]`

Gets the `DataStreamType` that this class marshals/unmarshals.

Returns

byte Id of this classes `DataStreamType`

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1122).

Reimplemented in `activemq::wireformat::openwire::marshal::generated::DataArrayResponseMarshaller` (p. 1073), `activemq::wireformat::openwire::marshal::generated::DataResponseMarshaller` (p. 1117), `activemq::wireformat::openwire::marshal::generated::ExceptionResponseMarshaller` (p. 1292), and `activemq::wireformat::openwire::marshal::generated::IntegerResponseMarshaller` (p. 1520).

6.452.3.3 `virtual void activemq::wireformat::openwire::marshal::generated::ResponseMarshaller::looseMarshal (OpenWireFormat * format, commands::DataStructure * command, decaf::io::DataOutputStream * ds)` `[virtual]`

Tight Marshal to the given stream.

Parameters

<i>format</i>	- The <code>OpenWireFormat</code> properties
<i>command</i>	- the object to Marshal
<i>ds</i>	- <code>DataOutputStream</code> to marshal to

Exceptions

<i>IOException</i>	if an error occurs.
--------------------	---------------------

Reimplemented from `activemq::wireformat::openwire::marshal::generated::BaseCommandMarshaller` (p. 500).

Reimplemented in `activemq::wireformat::openwire::marshal::generated::DataArrayResponseMarshaller` (p. 1073), `activemq::wireformat::openwire::marshal::generated::DataResponseMarshaller` (p. 1117), `activemq::wireformat::openwire::marshal::generated::ExceptionResponseMarshaller` (p. 1292), and `activemq::wireformat::openwire::marshal::generated::IntegerResponseMarshaller` (p. 1521).

6.452.3.4 `virtual void activemq::wireformat::openwire::marshal::generated-
::ResponseMarshaller::looseUnmarshal (OpenWireFormat * format,
commands::DataStructure * command, decaf::io::DataInputStream * dis)
[virtual]`

Loose Un-marhsal to the given stream.

Parameters

<i>format</i>	- The OpenWireFormat properties
<i>command</i>	- the object to Un-Marshal
<i>dis</i>	- the DataInputStream to Un-Marshal from

Exceptions

<i>IOException</i>	if an error occurs.
--------------------	---------------------

Reimplemented from **activemq::wireformat::openwire::marshal::generated::Base-CommandMarshaller** (p. 501).

Reimplemented in **activemq::wireformat::openwire::marshal::generated::DataArrayResponseMarshaller** (p. 1074), **activemq::wireformat::openwire::marshal::generated::DataResponseMarshaller** (p. 1117), **activemq::wireformat::openwire::marshal::generated::ExceptionResponseMarshaller** (p. 1293), and **activemq::wireformat::openwire::marshal::generated::IntegerResponseMarshaller** (p. 1521).

6.452.3.5 `virtual int activemq::wireformat::openwire::marshal::generated::-
ResponseMarshaller::tightMarshal1 (OpenWireFormat * format,
commands::DataStructure * command, utils::BooleanStream * bs)
[virtual]`

Tight Marhsal to the given stream.

Parameters

<i>format</i>	The OpenWireFormat properties
<i>command</i>	The object to Marshal
<i>bs</i>	The boolean stream to marshal to.

Exceptions

<i>IOException</i>	if an error occurs.
--------------------	---------------------

Reimplemented from **activemq::wireformat::openwire::marshal::generated::Base-CommandMarshaller** (p. 503).

Reimplemented in **activemq::wireformat::openwire::marshal::generated::DataArrayResponseMarshaller** (p. 1074), **activemq::wireformat::openwire-**

6.452 activemq::wireformat::openwire::marshal::generated::ResponseMarshaller Class Reference 2317

activemq::wireformat::openwire::marshal::generated::DataResponseMarshaller (p. 1118), **activemq::wireformat::openwire::marshal::generated::ExceptionResponseMarshaller** (p. 1293), and **activemq::wireformat::openwire::marshal::generated::IntegerResponseMarshaller** (p. 1521).

6.452.3.6 virtual void **activemq::wireformat::openwire::marshal::generated::ResponseMarshaller::tightMarshal2** (**OpenWireFormat** * *format*, **commands::DataStructure** * *command*, **decaf::io::DataOutputStream** * *ds*, **utils::BooleanStream** * *bs*) [virtual]

Tight Marshal to the given stream.

Parameters

<i>format</i>	- The OpenWireFormat properties
<i>command</i>	- the object to Marshal
<i>ds</i>	- the DataOutputStream to Marshal to
<i>bs</i>	- boolean stream to marshal to.

Exceptions

<i>IOException</i>	if an error occurs.
--------------------	---------------------

Reimplemented from **activemq::wireformat::openwire::marshal::generated::BaseCommandMarshaller** (p. 504).

Reimplemented in **activemq::wireformat::openwire::marshal::generated::DataArrayResponseMarshaller** (p. 1075), **activemq::wireformat::openwire::marshal::generated::DataResponseMarshaller** (p. 1118), **activemq::wireformat::openwire::marshal::generated::ExceptionResponseMarshaller** (p. 1293), and **activemq::wireformat::openwire::marshal::generated::IntegerResponseMarshaller** (p. 1522).

6.452.3.7 virtual void **activemq::wireformat::openwire::marshal::generated::ResponseMarshaller::tightUnmarshal** (**OpenWireFormat** * *format*, **commands::DataStructure** * *command*, **decaf::io::DataInputStream** * *dis*, **utils::BooleanStream** * *bs*) [virtual]

Tight Un-marshal to the given stream.

Parameters

<i>format</i>	- The OpenWireFormat properties
<i>command</i>	- the object to Un-Marshal
<i>dis</i>	- the DataInputStream to Un-Marshal from
<i>bs</i>	- boolean stream to unmarshal from.

Exceptions

<i>IOException</i> if an error occurs.
--

Reimplemented from **activemq::wireformat::openwire::marshal::generated::BaseCommandMarshaller** (p. 505).

Reimplemented in **activemq::wireformat::openwire::marshal::generated::DataArrayResponseMarshaller** (p. 1075), **activemq::wireformat::openwire::marshal::generated::DataResponseMarshaller** (p. 1119), **activemq::wireformat::openwire::marshal::generated::ExceptionResponseMarshaller** (p. 1294), and **activemq::wireformat::openwire::marshal::generated::IntegerResponseMarshaller** (p. 1522).

The documentation for this class was generated from the following file:

- `src/main/activemq/wireformat/openwire/marshal/generated/ResponseMarshaller.h`

6.453 decaf::lang::Runnable Class Reference

Interface for a runnable object - defines a task that can be run by a thread.

```
#include <src/main/decaf/lang/Runnable.h>
```

Inheritance diagram for `decaf::lang::Runnable`:

Public Member Functions

- virtual **~Runnable** ()
- virtual void **run** ()=0

*Run method - called by the **Thread** (p. 2703) class in the context of the thread.*

6.453.1 Detailed Description

Interface for a runnable object - defines a task that can be run by a thread.

6.453.2 Constructor & Destructor Documentation

6.453.2.1 virtual `decaf::lang::Runnable::~~Runnable ()` [`inline`, `virtual`]

6.453.3 Member Function Documentation

6.453.3.1 virtual void `decaf::lang::Runnable::run ()` [`pure virtual`]

Run method - called by the **Thread** (p. 2703) class in the context of the thread.

Implemented in `activemq::transport::IOTransport` (p. 1554), `decaf::lang::Thread` (p. 2710), `activemq::threads::CompositeTaskRunner` (p. 895), `activemq::threads::DedicatedTaskRunner` (p. 1145), `activemq::transport::mock::InternalCommandListener` (p. 1528), `activemq::transport::inactivity::WriteChecker` (p. 2908), `activemq::transport::inactivity::ReadChecker` (p. 2237), and `activemq::threads::SchedulerTimerTask` (p. 2320).

Referenced by `decaf::util::concurrent::ThreadPoolExecutor::CallerRunsPolicy::rejectedExecution()`.

The documentation for this class was generated from the following file:

- `src/main/decaf/lang/Runnable.h`

6.454 decaf::lang::Runtime Class Reference

```
#include <src/main/decaf/lang/Runtime.h>
```

Inheritance diagram for `decaf::lang::Runtime`:

Public Member Functions

- virtual `~Runtime()`

Static Public Member Functions

- static `Runtime * getRuntime()`
*Gets the single instance of the Decaf **Runtime** (p. 2313) for this Process.*
- static void `initializeRuntime(int argc, char **argv)`
Initialize the Decaf Library passing it the args that were passed to the application at startup.
- static void `initializeRuntime()`
Initialize the Decaf Library.
- static void `shutdownRuntime()`
Shutdown the Decaf Library, this call should take places after all objects that were created from the Decaf library have been deallocated.

6.454.1 Constructor & Destructor Documentation

6.454.1.1 `virtual decaf::lang::Runtime::~~Runtime()` [`inline`, `virtual`]

6.454.2 Member Function Documentation

6.454.2.1 `static Runtime* decaf::lang::Runtime::getRuntime () [static]`

Gets the single instance of the Decaf **Runtime** (p. 2313) for this Process.

Returns

pointer to the single Decaf **Runtime** (p. 2313) instance that exists for this process

6.454.2.2 `static void decaf::lang::Runtime::initializeRuntime (int argc, char ** argv) [static]`

Initialize the Decaf Library passing it the args that were passed to the application at startup.

Parameters

<i>argc</i>	- The number of args passed
<i>argv</i>	- Array of char* values passed to the Process on start.

Exceptions

<i>runtime_error</i>	if the library is already initialized or an error occurs during initialization.
----------------------	---

6.454.2.3 `static void decaf::lang::Runtime::initializeRuntime () [static]`

Initialize the Decaf Library.

Exceptions

<i>runtime_error</i>	if the library is already initialized or an error occurs during initialization.
----------------------	---

6.454.2.4 `static void decaf::lang::Runtime::shutdownRuntime () [static]`

Shutdown the Decaf Library, this call should take places after all objects that were created from the Decaf library have been deallocated.

Exceptions

<i>runtime_error</i>	if the library has not already been initialized or an error occurs during shutdown.
----------------------	---

The documentation for this class was generated from the following file:

- `src/main/decaf/lang/Runtime.h`

6.455 decaf::lang::exceptions::RuntimeException Class Reference

```
#include <src/main/decaf/lang/exceptions/RuntimeException.h>
```

Inheritance diagram for decaf::lang::exceptions::RuntimeException:

Public Member Functions

- **RuntimeException** () throw ()
Default Constructor.
- **RuntimeException** (const **Exception** &ex) throw ()
Conversion Constructor from some other ActiveMQException.
- **RuntimeException** (const **RuntimeException** &ex) throw ()
Copy Constructor.
- **RuntimeException** (const char *file, const int lineNumber, const std::exception *cause, const char *msg,...) throw ()
Constructor - Initializes the file name and line number where this message occurred.
- **RuntimeException** (const std::exception *cause) throw ()
Constructor.
- **RuntimeException** (const char *file, const int lineNumber, const char *msg,...) throw ()
Constructor - Initializes the file name and line number where this message occurred.
- virtual **RuntimeException** * clone () const
Clones this exception.
- virtual ~**RuntimeException** () throw ()

6.455.1 Constructor & Destructor Documentation

6.455.1.1 decaf::lang::exceptions::RuntimeException::RuntimeException ()
throw () [inline]

Default Constructor.

6.455.1.2 decaf::lang::exceptions::RuntimeException::RuntimeException (const
Exception & ex) throw () [inline]

Conversion Constructor from some other ActiveMQException.

Parameters

ex	The Exception (p. 1279) whose data is to be copied into this one.
----	--

6.455.1.3 **decaf::lang::exceptions::RuntimeException::RuntimeException (const RuntimeException & ex) throw ()** `[inline]`

Copy Constructor.

Parameters

<i>ex</i>	The Exception (p. 1279) whose data is to be copied into this one.
-----------	--

6.455.1.4 **decaf::lang::exceptions::RuntimeException::RuntimeException (const char * file, const int lineNumber, const std::exception * cause, const char * msg, ...) throw ()** `[inline]`

Constructor - Initializes the file name and line number where this message occurred.

Sets the message to report, using an optional list of arguments to parse into the message

Parameters

<i>file</i>	The file name where exception occurs
<i>lineNumber</i>	The line number where the exception occurred.
<i>cause</i>	The exception that was the cause for this one to be thrown.
<i>msg</i>	The message to report
<i>...</i>	list of primitives that are formatted into the message

6.455.1.5 **decaf::lang::exceptions::RuntimeException::RuntimeException (const std::exception * cause) throw ()** `[inline]`

Constructor.

Parameters

<i>cause</i>	Pointer (p. 2083) to the exception that caused this one to be thrown, the object is cloned caller retains ownership.
--------------	---

6.455.1.6 **decaf::lang::exceptions::RuntimeException::RuntimeException (const char * file, const int lineNumber, const char * msg, ...) throw ()** `[inline]`

Constructor - Initializes the file name and line number where this message occurred.

Sets the message to report, using an optional list of arguments to parse into the message

Parameters

<i>file</i>	The file name where exception occurs
<i>lineNumber</i>	The line number where the exception occurred.

<i>msg</i>	The message to report
...	list of primitives that are formatted into the message

6.455.1.7 virtual **decaf::lang::exceptions::RuntimeException::~~RuntimeException**
() throw() [inline, virtual]

6.455.2 Member Function Documentation

6.455.2.1 virtual **RuntimeException*** **decaf::lang::exceptions-
::RuntimeException::clone** () const [inline,
virtual]

Clones this exception.

This is useful for cases where you need to preserve the type of the original exception as well as the message. All subclasses should override.

Returns

an new **Exception** (p. 1279) that is a copy of this one.

Reimplemented from **decaf::lang::Exception** (p. 1282).

Reimplemented in **decaf::util::NoSuchElementException** (p. 1986), and **decaf::util-
::ConcurrentModificationException** (p. 904).

The documentation for this class was generated from the following file:

- src/main/decaf/lang/exceptions/**RuntimeException.h**

6.456 activemq::threads::Scheduler Class Reference

Scheduler (p. 2317) class for use in executing Runnable Tasks either periodically or one time only with optional delay.

```
#include <src/main/activemq/threads/Scheduler.h>
```

Inheritance diagram for activemq::threads::Scheduler:

Public Member Functions

- **Scheduler** (const std::string &name)
- virtual **~Scheduler** ()
- void **executePeriodically** (decaf::lang::Runnable *task, long long period, bool ownsTask=true)

- void **schedualPeriodically** (decaf::lang::Runnable *task, long long period, bool ownsTask=true)
- void **cancel** (decaf::lang::Runnable *task)
- void **executeAfterDelay** (decaf::lang::Runnable *task, long long delay, bool ownsTask=true)
- void **shutdown** ()

Protected Member Functions

- virtual void **doStart** ()
Performs the actual start operation on the service, acquiring all the resources needed to run the service.
- virtual void **doStop** (activemq::util::ServiceStopper *stopper)
Performs the actual stop operation on the service, ensuring that all resources held are released, must be implemented in derived class.

6.456.1 Detailed Description

Scheduler (p. 2317) class for use in executing Runnable Tasks either periodically or one time only with optional delay.

Since

3.3.0

6.456.2 Constructor & Destructor Documentation

6.456.2.1 `activemq::threads::Scheduler::Scheduler (const std::string & name)`

6.456.2.2 `virtual activemq::threads::Scheduler::~~Scheduler ()` [virtual]

6.456.3 Member Function Documentation

6.456.3.1 `void activemq::threads::Scheduler::cancel (decaf::lang::Runnable * task)`

6.456.3.2 `virtual void activemq::threads::Scheduler::doStart ()` [protected, virtual]

Performs the actual start operation on the service, acquiring all the resources needed to run the service.

Must be implemented in derived class.

Implements **activemq::util::ServiceSupport** (p. 2360).

6.456.3.3 `virtual void activemq::threads::Scheduler::doStop (`
`activemq::util::ServiceStopper * stopper) [protected, virtual]`

Performs the actual stop operation on the service, ensuring that all resources held are released, must be implemented in derived class.

Implements `activemq::util::ServiceSupport` (p. 2360).

6.456.3.4 `void activemq::threads::Scheduler::executeAfterDelay (`
`decaf::lang::Runnable * task, long long delay, bool ownsTask = true)`

6.456.3.5 `void activemq::threads::Scheduler::executePeriodically (`
`decaf::lang::Runnable * task, long long period, bool ownsTask = true)`

6.456.3.6 `void activemq::threads::Scheduler::schedulingPeriodically (`
`decaf::lang::Runnable * task, long long period, bool ownsTask = true)`

6.456.3.7 `void activemq::threads::Scheduler::shutdown ()`

The documentation for this class was generated from the following file:

- `src/main/activemq/threads/Scheduler.h`

6.457 `activemq::threads::SchedulerTimerTask` Class Reference

Extension of the Decaf TimerTask that adds a Runnable instance which is the target of this task.

```
#include <src/main/activemq/threads/SchedulerTimerTask.h>
```

Inheritance diagram for `activemq::threads::SchedulerTimerTask`:

Public Member Functions

- `SchedulerTimerTask (decaf::lang::Runnable *task, bool ownsTask=true)`
- `virtual ~SchedulerTimerTask ()`
- `virtual void run ()`

Run method - called by the Thread class in the context of the thread.

6.457.1 Detailed Description

Extension of the Decaf TimerTask that adds a Runnable instance which is the target of this task.

Since

3.3.0

6.457.2 Constructor & Destructor Documentation

6.457.2.1 `activemq::threads::SchedulerTimerTask::SchedulerTimerTask (
 decaf::lang::Runnable * task, bool ownsTask = true)`

6.457.2.2 `virtual activemq::threads::SchedulerTimerTask::~~SchedulerTimerTask (
) [virtual]`

6.457.3 Member Function Documentation

6.457.3.1 `virtual void activemq::threads::SchedulerTimerTask::run ()
 [virtual]`

Run method - called by the Thread class in the context of the thread.

Implements **decaf::lang::Runnable** (p. 2312).

The documentation for this class was generated from the following file:

- `src/main/activemq/threads/SchedulerTimerTask.h`

6.458 decaf::security::SecureRandom Class Reference

```
#include <src/main/decaf/security/SecureRandom.h>
```

Inheritance diagram for `decaf::security::SecureRandom`:

Public Member Functions

- **SecureRandom** ()
Creates a new instance of a secure random number generator that implements the default random number algorithm.
- **SecureRandom** (const std::vector< unsigned char > &seed)
Creates a new instance of a secure random number generator that implements the default random number algorithm.
- **SecureRandom** (const unsigned char *seed, int size)
Creates a new instance of a secure random number generator that implements the default random number algorithm.
- virtual ~**SecureRandom** ()
- virtual void **nextBytes** (std::vector< unsigned char > &buf)
Modifies the byte array by a random sequence of bytes generated by this random number generator.

Parameters

buf	non-null array to contain the new random bytes
-----	--

See also

next (p. 2231)

- virtual void **nextBytes** (unsigned char *buf, int size)

Modifies the byte array by a random sequence of bytes generated by this random number generator.

Parameters

buf	non-null array to contain the new random bytes
-----	--

See also

next (p. 2231)

Exceptions

NullPointerException	if buff is NULL
IllegalArgumentException	if size is negative

- virtual void **setSeed** (unsigned long long seed)

Modifies the seed using linear congruential formula presented in The Art of Computer Programming, Volume 2, Section 3.2.1.

Parameters

seed	the seed that alters the state of the random number generator
------	---

See also

next (p. 2231)**Random()** (p. 2230)

#Random(long)

- virtual void **setSeed** (const std::vector< unsigned char > &seed)

Supplements or sets the seed of this secure random number generator, calls to this method never reduces randomness.

- virtual void **setSeed** (const unsigned char *seed, int size)

Supplements or sets the seed of this secure random number generator, calls to this method never reduces randomness.

Protected Member Functions

- virtual int **next** (int bits)

Answers a pseudo-random uniformly distributed int value of the number of bits specified by the argument bits as described by Donald E.

Knuth in The Art of Computer Programming, Volume 2: Seminumerical Algorithms, section 3.2.1.

Returns

int a pseudo-random generated int number

Parameters

bits	number of bits of the returned value
------	--------------------------------------

See also

nextBytes (p. 2232)
nextDouble (p. 2233)
nextFloat (p. 2233)
nextInt() (p. 2234)
nextInt(int) (p. 2234)
nextGaussian (p. 2233)
nextLong (p. 2234)

6.458.1 Detailed Description

Since

1.0

6.458.2 Constructor & Destructor Documentation

6.458.2.1 `decaf::security::SecureRandom::SecureRandom ()`

Creates a new instance of a secure random number generator that implements the default random number algorithm.

The **SecureRandom** (p. 2320) instance that is created with this constructor is unseeded and can be seeded by calling the `setSeed` method. Calls to `nextBytes` on an unseeded **SecureRandom** (p. 2320) result in the object seeding itself.

6.458.2.2 `decaf::security::SecureRandom::SecureRandom (const std::vector< unsigned char > & seed)`

Creates a new instance of a secure random number generator that implements the default random number algorithm.

The **SecureRandom** (p. 2320) instance created by this constructor is seeded using the passed byte array.

Parameters

<i>seed</i>	The seed bytes to use to seed this secure random number generator.
-------------	--

6.458.2.3 `decaf::security::SecureRandom::SecureRandom (const unsigned char * seed, int size)`

Creates a new instance of a secure random number generator that implements the default random number algorithm.

The **SecureRandom** (p. 2320) instance created by this constructor is seeded using the passed byte array.

Parameters

<i>seed</i>	The seed bytes to use to seed this secure random number generator.
<i>size</i>	The number of bytes in the seed buffer.

Exceptions

<i>NullPointerException</i>	if the seed buffer is NULL.
<i>IllegalArgument-Exception</i>	if the size value is negative.

6.458.2.4 `virtual decaf::security::SecureRandom::~SecureRandom ()`
[virtual]

6.458.3 Member Function Documentation

6.458.3.1 `virtual int decaf::security::SecureRandom::next (int bits)`
[protected, virtual]

Answers a pseudo-random uniformly distributed `int` value of the number of bits specified by the argument `bits` as described by Donald E.

Knuth in *The Art of Computer Programming, Volume 2: Seminumerical Algorithms*, section 3.2.1.

Returns

`int` a pseudo-random generated `int` number

Parameters

<i>bits</i>	number of bits of the returned value
-------------	--------------------------------------

See also

nextBytes (p. 2232)
nextDouble (p. 2233)
nextFloat (p. 2233)
nextInt() (p. 2234)
nextInt(int) (p. 2234)
nextGaussian (p. 2233)
nextLong (p. 2234)

Reimplemented from **decaf::util::Random** (p. 2231).

6.458.3.2 `virtual void decaf::security::SecureRandom::nextBytes (std::vector< unsigned char > & buf) [virtual]`

Modifies the byte array by a random sequence of bytes generated by this random number generator.

Parameters

<i>buf</i>	non-null array to contain the new random bytes
------------	--

See also

next (p. 2231)

Reimplemented from **decaf::util::Random** (p. 2232).

6.458.3.3 `virtual void decaf::security::SecureRandom::nextBytes (unsigned char * buf, int size) [virtual]`

Modifies the byte array by a random sequence of bytes generated by this random number generator.

Parameters

<i>buf</i>	non-null array to contain the new random bytes
------------	--

See also

next (p. 2231)

Exceptions

<i>NullPointerException</i>	if <i>buff</i> is NULL
<i>IllegalArgumentException</i>	if <i>size</i> is negative

Reimplemented from **decaf::util::Random** (p. 2232).

6.458.3.4 `virtual void decaf::security::SecureRandom::setSeed (unsigned long long seed) [virtual]`

Modifies the seed using linear congruential formula presented in *The Art of Computer Programming, Volume 2*, Section 3.2.1.

Parameters

<i>seed</i>	the seed that alters the state of the random number generator
-------------	---

See also

`next` (p. 2231)
`Random()` (p. 2230)
`#Random(long)`

Reimplemented from `decaf::util::Random` (p. 2235).

6.458.3.5 `virtual void decaf::security::SecureRandom::setSeed (const std::vector< unsigned char > & seed) [virtual]`

Supplements or sets the seed of this secure random number generator, calls to this method never reduces randomness.

Parameters

<i>seed</i>	A vector of bytes that is used update the seed of the RNG.
-------------	--

6.458.3.6 `virtual void decaf::security::SecureRandom::setSeed (const unsigned char * seed, int size) [virtual]`

Supplements or sets the seed of this secure random number generator, calls to this method never reduces randomness.

Parameters

<i>seed</i>	The seed bytes to use to seed this secure random number generator.
<i>size</i>	The number of bytes in the seed buffer.

Exceptions

<i>NullPointerException</i>	if the seed buffer is NULL.
<i>IllegalArgumentException</i>	if the size value is negative.

The documentation for this class was generated from the following file:

- `src/main/decaf/security/SecureRandom.h`

6.459 `decaf::internal::security::SecureRandomImpl` Class Reference

Secure Random Number Generator for Unix based platforms that attempts to obtain secure bytes with high entropy from known sources.

```
#include <src/main/decaf/internal/security/unix/Secure-
RandomImpl.h>
```

Inheritance diagram for decaf::internal::security::SecureRandomImpl:

Public Member Functions

- **SecureRandomImpl** ()
- virtual **~SecureRandomImpl** ()
- virtual void **providerSetSeed** (const unsigned char *seed, int size)
Reseed the Random Number Generator, the value given supplements the previous seed value if any instead of replacing it.
- virtual void **providerNextBytes** (unsigned char *bytes, int numBytes)
Generates the number of random bytes specified by the size parameter and write them to the passed bytes array.
- virtual unsigned char * **providerGenerateSeed** (int numBytes)
Generates a new set of seed bytes, the returned value may be used to seed another Random Number Generator.
- **SecureRandomImpl** ()
- virtual **~SecureRandomImpl** ()
- virtual void **providerSetSeed** (const unsigned char *seed, int size)
Reseed the Random Number Generator, the value given supplements the previous seed value if any instead of replacing it.
- virtual void **providerNextBytes** (unsigned char *bytes, int numBytes)
Generates the number of random bytes specified by the size parameter and write them to the passed bytes array.
- virtual unsigned char * **providerGenerateSeed** (int numBytes)
Generates a new set of seed bytes, the returned value may be used to seed another Random Number Generator.

6.459.1 Detailed Description

Secure Random Number Generator for Unix based platforms that attempts to obtain secure bytes with high entropy from known sources.

Secure Random Number Generator for Windows based platforms that attempts to obtain secure bytes with high entropy from known sources.

If the platform does not have a source of secure bytes then the platform random number generator is used if one exists otherwise the Decaf RNG is used as a last resort.

Since

1.0

6.459.2 Constructor & Destructor Documentation

6.459.2.1 `decaf::internal::security::SecureRandomImpl::SecureRandomImpl ()`

6.459.2.2 `virtual decaf::internal::security::SecureRandomImpl::~~SecureRandomImpl ()` `[virtual]`

6.459.2.3 `decaf::internal::security::SecureRandomImpl::SecureRandomImpl ()`

6.459.2.4 `virtual decaf::internal::security::SecureRandomImpl::~~SecureRandomImpl ()` `[virtual]`

6.459.3 Member Function Documentation

6.459.3.1 `virtual unsigned char* decaf::internal::security::SecureRandomImpl::providerGenerateSeed (int numBytes)` `[virtual]`

Generates a new set of seed bytes, the returned value may be used to seed another Random Number Generator.

The caller owns the returned array and must delete it.

Parameters

<i>numBytes</i>	The number of bytes that should be generated for the new seed array.
-----------------	--

Implements `decaf::security::SecureRandomSpi` (p. 2330).

6.459.3.2 `virtual unsigned char* decaf::internal::security::SecureRandomImpl::providerGenerateSeed (int numBytes)` `[virtual]`

Generates a new set of seed bytes, the returned value may be used to seed another Random Number Generator.

The caller owns the returned array and must delete it.

Parameters

<i>numBytes</i>	The number of bytes that should be generated for the new seed array.
-----------------	--

Implements `decaf::security::SecureRandomSpi` (p. 2330).

6.459.3.3 **virtual void decaf::internal::security::SecureRandomImpl-
::providerNextBytes (unsigned char * *bytes*, int *numBytes*)**
[virtual]

Generates the number of random bytes specified by the size parameter and write them to the passed bytes array.

The array must have already been allocated and be of the correct size to prevent segmentation faults.

Parameters

<i>bytes</i>	The array that will be filled with random bytes equal to size.
<i>numBytes</i>	The number of bytes to generate and write into the bytes array.

Implements **decaf::security::SecureRandomSpi** (p. 2330).

6.459.3.4 **virtual void decaf::internal::security::SecureRandomImpl-
::providerNextBytes (unsigned char * *bytes*, int *numBytes*)**
[virtual]

Generates the number of random bytes specified by the size parameter and write them to the passed bytes array.

The array must have already been allocated and be of the correct size to prevent segmentation faults.

Parameters

<i>bytes</i>	The array that will be filled with random bytes equal to size.
<i>numBytes</i>	The number of bytes to generate and write into the bytes array.

Implements **decaf::security::SecureRandomSpi** (p. 2330).

6.459.3.5 **virtual void decaf::internal::security::SecureRandomImpl-
::providerSetSeed (const unsigned char * *seed*, int *size*)**
[virtual]

Reseed the Random Number Generator, the value given supplements the previous seed value if any instead of replacing it.

Parameters

<i>seed</i>	The array of bytes used to update the generators seed.
<i>size</i>	The size of the passed byte array.

Implements **decaf::security::SecureRandomSpi** (p. 2331).

6.459.3.6 virtual void **decaf::internal::security::SecureRandomImpl-
::providerSetSeed** (const unsigned char * *seed*, int *size*)
[virtual]

Reseed the Random Number Generator, the value given supplements the previous seed value if any instead of replacing it.

Parameters

<i>seed</i>	The array of bytes used to update the generators seed.
<i>size</i>	The size of the passed byte array.

Implements **decaf::security::SecureRandomSpi** (p. 2331).

The documentation for this class was generated from the following files:

- src/main/decaf/internal/security/unix/**SecureRandomImpl.h**
- src/main/decaf/internal/security/windows/**SecureRandomImpl.h**

6.460 decaf::security::SecureRandomSpi Class Reference

Interface class used by Security Service Providers to implement a source of secure random bytes.

```
#include <src/main/decaf/security/SecureRandomSpi.h>
```

Inheritance diagram for decaf::security::SecureRandomSpi:

Public Member Functions

- **SecureRandomSpi** ()
- virtual **~SecureRandomSpi** ()
- virtual void **providerSetSeed** (const unsigned char **seed*, int *size*)=0
Reseed the Random Number Generator, the value given supplements the previous seed value if any instead of replacing it.
- virtual void **providerNextBytes** (unsigned char **bytes*, int *numBytes*)=0
Generates the number of random bytes specified by the size parameter and write them to the passed bytes array.
- virtual unsigned char * **providerGenerateSeed** (int *numBytes*)=0
Generates a new set of seed bytes, the returned value may be used to seed another Random Number Generator.

6.460.1 Detailed Description

Interface class used by Security Service Providers to implement a source of secure random bytes.

Since

1.0

6.460.2 Constructor & Destructor Documentation

6.460.2.1 **decaf::security::SecureRandomSpi::SecureRandomSpi ()**

6.460.2.2 **virtual decaf::security::SecureRandomSpi::~~SecureRandomSpi ()**
[virtual]

6.460.3 Member Function Documentation

6.460.3.1 **virtual unsigned char* decaf::security::SecureRandomSpi::providerGenerateSeed (int *numBytes*)** [pure virtual]

Generates a new set of seed bytes, the returned value may be used to seed another Random Number Generator.

The caller owns the returned array and must delete it.

Parameters

<i>numBytes</i>	The number of bytes that should be generated for the new seed array.
-----------------	--

Implemented in **decaf::internal::security::SecureRandomImpl** (p. 2327), and **decaf::internal::security::SecureRandomImpl** (p. 2327).

6.460.3.2 **virtual void decaf::security::SecureRandomSpi::providerNextBytes (unsigned char * *bytes*, int *numBytes*)** [pure virtual]

Generates the number of random bytes specified by the size parameter and write them to the passed bytes array.

The array must have already been allocated and be of the correct size to prevent segmentation faults.

Parameters

<i>bytes</i>	The array that will be filled with random bytes equal to size.
<i>numBytes</i>	The number of bytes to generate and write into the bytes array.

Implemented in **decaf::internal::security::SecureRandomImpl** (p. 2328), and **decaf::internal::security::SecureRandomImpl** (p. 2328).

6.460.3.3 virtual void **decaf::security::SecureRandomSpi::providerSetSeed** (const unsigned char * *seed*, int *size*) [pure virtual]

Reseed the Random Number Generator, the value given supplements the previous seed value if any instead of replacing it.

Parameters

<i>seed</i>	The array of bytes used to update the generators seed.
<i>size</i>	The size of the passed byte array.

Implemented in **decaf::internal::security::SecureRandomImpl** (p. 2329), and **decaf::internal::security::SecureRandomImpl** (p. 2329).

The documentation for this class was generated from the following file:

- src/main/decaf/security/**SecureRandomSpi.h**

6.461 decaf::util::concurrent::Semaphore Class Reference

A counting semaphore.

```
#include <src/main/decaf/util/concurrent/Semaphore.h>
```

Public Member Functions

- **Semaphore** (int permits)
*Creates a **Semaphore** (p. 2331) with the given number of permits and nonfair fairness setting.*
- **Semaphore** (int permits, bool fair)
*Creates a **Semaphore** (p. 2331) with the given number of permits and the given fairness setting.*
- virtual **~Semaphore** ()
- void **acquire** ()
Acquires a permit from this semaphore, blocking until one is available, or the thread is interrupted.
- void **acquireUninterruptibly** ()
Acquires a permit from this semaphore, blocking until one is available.
- bool **tryAcquire** ()
Acquires a permit from this semaphore, only if one is available at the time of invocation.
- bool **tryAcquire** (long long timeout, const **TimeUnit** &unit)
Acquires a permit from this semaphore, if one becomes available within the given waiting time and the current thread has not been interrupted.
- void **release** ()
Releases a permit, returning it to the semaphore.
- void **acquire** (int permits)

Acquires the given number of permits from this semaphore, blocking until all are available, or the thread is interrupted.

- void **acquireUninterruptibly** (int permits)

Acquires the given number of permits from this semaphore, blocking until all are available.

- bool **tryAcquire** (int permits)

Acquires the given number of permits from this semaphore, only if all are available at the time of invocation.

- bool **tryAcquire** (int permits, long long timeout, const **TimeUnit** &unit)

Acquires the given number of permits from this semaphore, if all become available within the given waiting time and the current thread has not been interrupted.

- void **release** (int permits)

Releases the given number of permits, returning them to the semaphore.

- int **availablePermits** () const

Returns the current number of permits available in this semaphore.

- int **drainPermits** ()

Acquires and returns all permits that are immediately available.

- bool **isFair** () const

- std::string **toString** () const

Returns a string identifying this semaphore, as well as its state.

6.461.1 Detailed Description

A counting semaphore.

Conceptually, a semaphore maintains a set of permits. Each **acquire()** (p. 2334) blocks if necessary until a permit is available, and then takes it. Each **release()** (p. 2337) adds a permit, potentially releasing a blocking acquirer. However, no actual permit objects are used; the **Semaphore** (p. 2331) just keeps a count of the number available and acts accordingly.

Semaphores are often used to restrict the number of threads than can access some (physical or logical) resource.

```
class Pool { private:
```

```
static const int MAX_AVAILABLE = 100; Semaphore (p. 2331) available;
```

```
std::vector<std::string> items; std::vector<bool> used;
```

```
Mutex (p. 1960) lock;
```

```
public:
```

```
Pool() : available( MAX_AVAILABLE, true ) { used.resize( MAX_AVAILABLE ); items.-  
resize( MAX_AVAILABLE ); }
```

```
std::string getItem() throws InterruptedException { available.acquire(); return getNext-  
AvailableItem(); }
```

```
void putItem( std::string x ) { if( markAsUnused(x) ) { available.release(); } }
```

```
std::string getNextAvailableItem() {
```



```
synchronized( &lock ) (p. 3231) { for( int i = 0; i < MAX_AVAILABLE; ++i ) { if( !used[i] ) { used[i] = true; return items[i]; } }
```

```
return std::string(); // not reached }
```

```
bool markAsUnused( const std::string& item ) { synchronized( &lock ) (p. 3231) { for( int i = 0; i < MAX_AVAILABLE; ++i ) { if( item == items[i] ) { if( used[i] ) { used[i] = false; return true; } else return false; } } } return false; } };
```

Before obtaining an item each thread must acquire a permit from the semaphore, guaranteeing that an item is available for use. When the thread has finished with the item it is returned back to the pool and a permit is returned to the semaphore, allowing another thread to acquire that item. Note that no synchronization lock is held when **acquire()** (p. 2334) is called as that would prevent an item from being returned to the pool. The semaphore encapsulates the synchronization needed to restrict access to the pool, separately from any synchronization needed to maintain the consistency of the pool itself.

A semaphore initialized to one, and which is used such that it only has at most one permit available, can serve as a mutual exclusion lock. This is more commonly known as a binary semaphore, because it only has two states: one permit available, or zero permits available. When used in this way, the binary semaphore has the property (unlike many **Lock** (p. 1682) implementations), that the "lock" can be released by a thread other than the owner (as semaphores have no notion of ownership). This can be useful in some specialized contexts, such as deadlock recovery.

The constructor for this class optionally accepts a fairness parameter. When set false, this class makes no guarantees about the order in which threads acquire permits. In particular, barging is permitted, that is, a thread invoking **acquire()** (p. 2334) can be allocated a permit ahead of a thread that has been waiting - logically the new thread places itself at the head of the queue of waiting threads. When fairness is set true, the semaphore guarantees that threads invoking any of the acquire methods are selected to obtain permits in the order in which their invocation of those methods was processed (first-in-first-out; FIFO). Note that FIFO ordering necessarily applies to specific internal points of execution within these methods. So, it is possible for one thread to invoke acquire before another, but reach the ordering point after the other, and similarly upon return from the method. Also note that the untimed tryAcquire methods do not honor the fairness setting, but will take any permits that are available.

Generally, semaphores used to control resource access should be initialized as fair, to ensure that no thread is starved out from accessing a resource. When using semaphores for other kinds of synchronization control, the throughput advantages of non-fair ordering often outweigh fairness considerations.

This class also provides convenience methods to acquire and release multiple permits at a time. Beware of the increased risk of indefinite postponement when these methods are used without fairness set true.

Since

1.0

6.461.2 Constructor & Destructor Documentation

6.461.2.1 `decaf::util::concurrent::Semaphore::Semaphore (int permits)`

Creates a **Semaphore** (p. 2331) with the given number of permits and nonfair fairness setting.

Parameters

<i>permits</i>	the initial number of permits available. This value may be negative, in which case releases must occur before any acquires will be granted.
----------------	---

6.461.2.2 `decaf::util::concurrent::Semaphore::Semaphore (int permits, bool fair)`

Creates a **Semaphore** (p. 2331) with the given number of permits and the given fairness setting.

Parameters

<i>permits</i>	the initial number of permits available. This value may be negative, in which case releases must occur before any acquires will be granted.
<i>fair</i>	true if this semaphore will guarantee first-in first-out granting of permits under contention, else false

6.461.2.3 `virtual decaf::util::concurrent::Semaphore::~~Semaphore ()` [virtual]

6.461.3 Member Function Documentation

6.461.3.1 `void decaf::util::concurrent::Semaphore::acquire ()`

Acquires a permit from this semaphore, blocking until one is available, or the thread is interrupted.

Acquires a permit, if one is available and returns immediately, reducing the number of available permits by one.

If no permit is available then the current thread becomes disabled for thread scheduling purposes and lies dormant until one of two things happens:

* Some other thread invokes the **release()** (p. 2337) method for this semaphore and the current thread is next to be assigned a permit; or * Some other thread interrupts the current thread.

If the current thread:

* has its interrupted status set on entry to this method; or * is interrupted while waiting for a permit,

then InterruptedException is thrown and the current thread's interrupted status is cleared.

Exceptions

<i>InterruptedException</i>	- if the current thread is interrupted.
<i>RuntimeException</i>	if an unexpected error occurs while acquiring the Semaphore (p. 2331).

6.461.3.2 void decaf::util::concurrent::Semaphore::acquire (int *permits*)

Acquires the given number of permits from this semaphore, blocking until all are available, or the thread is interrupted.

Acquires the given number of permits, if they are available, and returns immediately, reducing the number of available permits by the given amount.

If insufficient permits are available then the current thread becomes disabled for thread scheduling purposes and lies dormant until one of two things happens:

* Some other thread invokes one of the release methods for this semaphore, the current thread is next to be assigned permits and the number of available permits satisfies this request; or * Some other thread interrupts the current thread.

If the current thread:

* has its interrupted status set on entry to this method; or * is interrupted while waiting for a permit,

then *InterruptedException* is thrown and the current thread's interrupted status is cleared. Any permits that were to be assigned to this thread are instead assigned to other threads trying to acquire permits, as if permits had been made available by a call to **release()** (p. 2337).

Parameters

<i>permits</i>	the number of permits to acquire.
----------------	-----------------------------------

Exceptions

<i>InterruptedException</i>	if the current thread is interrupted.
<i>IllegalArgumentException</i>	if the permits argument is negative.
<i>RuntimeException</i>	if an unexpected error occurs while acquiring the Semaphore (p. 2331).

6.461.3.3 void decaf::util::concurrent::Semaphore::acquireUninterruptibly ()

Acquires a permit from this semaphore, blocking until one is available.

Acquires a permit, if one is available and returns immediately, reducing the number of available permits by one.

If no permit is available then the current thread becomes disabled for thread schedul-

ing purposes and lies dormant until some other thread invokes the **release()** (p. 2337) method for this semaphore and the current thread is next to be assigned a permit.

If the current thread is interrupted while waiting for a permit then it will continue to wait, but the time at which the thread is assigned a permit may change compared to the time it would have received the permit had no interruption occurred. When the thread does return from this method its interrupt status will be set.

Exceptions

<i>RuntimeException</i>	if an unexpected error occurs while acquiring the Semaphore (p. 2331).
-------------------------	---

6.461.3.4 void decaf::util::concurrent::Semaphore::acquireUninterruptibly (int permits)

Acquires the given number of permits from this semaphore, blocking until all are available.

Acquires the given number of permits, if they are available, and returns immediately, reducing the number of available permits by the given amount.

If insufficient permits are available then the current thread becomes disabled for thread scheduling purposes and lies dormant until some other thread invokes one of the release methods for this semaphore, the current thread is next to be assigned permits and the number of available permits satisfies this request.

If the current thread is interrupted while waiting for permits then it will continue to wait and its position in the queue is not affected. When the thread does return from this method its interrupt status will be set.

Parameters

<i>permits</i>	the number of permits to acquire.
----------------	-----------------------------------

Exceptions

<i>IllegalArgumentException</i>	if the permits argument is negative.
<i>RuntimeException</i>	if an unexpected error occurs while acquiring the Semaphore (p. 2331).

6.461.3.5 int decaf::util::concurrent::Semaphore::availablePermits () const

Returns the current number of permits available in this semaphore.

This method is typically used for debugging and testing purposes.

Returns

the number of permits available in this semaphore

6.461.3.6 int decaf::util::concurrent::Semaphore::drainPermits ()

Acquires and returns all permits that are immediately available.

Returns

the number of permits acquired

Exceptions

<i>RuntimeException</i>	if an unexpected error occurs while draining the Semaphore (p. 2331).
-------------------------	--

6.461.3.7 bool decaf::util::concurrent::Semaphore::isFair () const**Returns**

true if this semaphore has fairness set true

6.461.3.8 void decaf::util::concurrent::Semaphore::release ()

Releases a permit, returning it to the semaphore.

Releases a permit, increasing the number of available permits by one. If any threads are trying to acquire a permit, then one is selected and given the permit that was just released. That thread is (re)enabled for thread scheduling purposes.

There is no requirement that a thread that releases a permit must have acquired that permit by calling **acquire()** (p. 2334). Correct usage of a semaphore is established by programming convention in the application.

Exceptions

<i>RuntimeException</i>	if an unexpected error occurs while releasing the Semaphore (p. 2331).
-------------------------	---

6.461.3.9 void decaf::util::concurrent::Semaphore::release (int permits)

Releases the given number of permits, returning them to the semaphore.

Releases the given number of permits, increasing the number of available permits by that amount. If any threads are trying to acquire permits, then one is selected and given

the permits that were just released. If the number of available permits satisfies that thread's request then that thread is (re)enabled for thread scheduling purposes; otherwise the thread will wait until sufficient permits are available. If there are still permits available after this thread's request has been satisfied, then those permits are assigned in turn to other threads trying to acquire permits.

Parameters

<i>permits</i>	the number of permits to release
----------------	----------------------------------

Exceptions

<i>IllegalArgument-Exception</i>	if the permits argument is negative.
<i>RuntimeException</i>	if an unexpected error occurs while releasing the Semaphore (p. 2331).

6.461.3.10 `std::string decaf::util::concurrent::Semaphore::toString () const`

Returns a string identifying this semaphore, as well as its state.

The state, in brackets, includes the String "Permits =" followed by the number of permits.

Returns

a string identifying this semaphore, as well as its state

6.461.3.11 `bool decaf::util::concurrent::Semaphore::tryAcquire ()`

Acquires a permit from this semaphore, only if one is available at the time of invocation.

Acquires a permit, if one is available and returns immediately, with the value true, reducing the number of available permits by one.

If no permit is available then this method will return immediately with the value false.

Even when this semaphore has been set to use a fair ordering policy, a call to **tryAcquire()** (p. 2338) will immediately acquire a permit if one is available, whether or not other threads are currently waiting. This "barging" behavior can be useful in certain circumstances, even though it breaks fairness. If you want to honor the fairness setting, then use `tryAcquire(0, TimeUnit.SECONDS)` (p. 2764) which is almost equivalent (it also detects interruption).

Returns

true if a permit was acquired and false otherwise

Exceptions

<i>RuntimeException</i>	if an unexpected error occurs while acquiring the Semaphore (p. 2331).
-------------------------	---

6.461.3.12 `bool decaf::util::concurrent::Semaphore::tryAcquire (long long timeout,
const TimeUnit & unit)`

Acquires a permit from this semaphore, if one becomes available within the given waiting time and the current thread has not been interrupted.

Acquires a permit, if one is available and returns immediately, with the value true, reducing the number of available permits by one.

If no permit is available then the current thread becomes disabled for thread scheduling purposes and lies dormant until one of three things happens:

* Some other thread invokes the **release()** (p. 2337) method for this semaphore and the current thread is next to be assigned a permit; or * Some other thread interrupts the current thread; or * The specified waiting time elapses.

If a permit is acquired then the value true is returned.

If the current thread:

* has its interrupted status set on entry to this method; or * is interrupted while waiting to acquire a permit,

then InterruptedException is thrown and the current thread's interrupted status is cleared.

If the specified waiting time elapses then the value false is returned. If the time is less than or equal to zero, the method will not wait at all.

Parameters

<i>timeout</i>	the maximum time to wait for a permit
<i>unit</i>	the time unit of the timeout argument

Returns

true if a permit was acquired and false if the waiting time elapsed before a permit was acquired

Exceptions

<i>InterruptedException</i>	if the current thread is interrupted.
<i>RuntimeException</i>	if an unexpected error occurs while acquiring the Semaphore (p. 2331).

6.461.3.13 `bool decaf::util::concurrent::Semaphore::tryAcquire (int permits)`

Acquires the given number of permits from this semaphore, only if all are available at the time of invocation.

Acquires the given number of permits, if they are available, and returns immediately, with the value true, reducing the number of available permits by the given amount.

If insufficient permits are available then this method will return immediately with the value false and the number of available permits is unchanged.

Even when this semaphore has been set to use a fair ordering policy, a call to try-Acquire will immediately acquire a permit if one is available, whether or not other threads are currently waiting. This "barging" behavior can be useful in certain circumstances, even though it breaks fairness. If you want to honor the fairness setting, then use try-Acquire(permits, 0, **TimeUnit.SECONDS** (p. 2764)) which is almost equivalent (it also detects interruption).

Parameters

<i>permits</i>	the number of permits to acquire
----------------	----------------------------------

Returns

true if the permits were acquired and false otherwise.

Exceptions

<i>IllegalArgument-Exception</i>	if the permits argument is negative.
<i>RuntimeException</i>	if an unexpected error occurs while acquiring the Semaphore (p. 2331).

6.461.3.14 `bool decaf::util::concurrent::Semaphore::tryAcquire (int permits, long long timeout, const TimeUnit & unit)`

Acquires the given number of permits from this semaphore, if all become available within the given waiting time and the current thread has not been interrupted.

Acquires the given number of permits, if they are available and returns immediately, with the value true, reducing the number of available permits by the given amount.

If insufficient permits are available then the current thread becomes disabled for thread scheduling purposes and lies dormant until one of three things happens:

* Some other thread invokes one of the release methods for this semaphore, the current thread is next to be assigned permits and the number of available permits satisfies this request; or * Some other thread interrupts the current thread; or * The specified waiting time elapses.

If the permits are acquired then the value true is returned.

If the current thread:

* has its interrupted status set on entry to this method; or * is interrupted while waiting to acquire the permits,

then InterruptedException is thrown and the current thread's interrupted status is cleared. Any permits that were to be assigned to this thread, are instead assigned to other threads trying to acquire permits, as if the permits had been made available by

a call to **release()** (p. 2337).

If the specified waiting time elapses then the value false is returned. If the time is less than or equal to zero, the method will not wait at all. Any permits that were to be assigned to this thread, are instead assigned to other threads trying to acquire permits, as if the permits had been made available by a call to **release()** (p. 2337).

Parameters

<i>permits</i>	the number of permits to acquire
<i>timeout</i>	the maximum amount of time to wait to acquire the permits.
<i>unit</i>	the units that the timeout param represents.

Returns

true if all permits were acquired and false if the waiting time elapsed before all permits were acquired

Exceptions

<i>IllegalArgumentException</i>	if the permits argument is negative.
<i>RuntimeException</i>	if an unexpected error occurs while acquiring the Semaphore (p. 2331).

The documentation for this class was generated from the following file:

- src/main/decaf/util/concurrent/**Semaphore.h**

6.462 activemq::cmsutil::CmsTemplate::SendExecutor Class - Reference

```
#include <src/main/activemq/cmsutil/CmsTemplate.h>
```

Inheritance diagram for activemq::cmsutil::CmsTemplate::SendExecutor:

Public Member Functions

- **SendExecutor** (**MessageCreator** *messageCreator, **CmsTemplate** *parent)
- virtual ~**SendExecutor** () throw ()
- virtual void **doInCms** (**cms::Session** *session, **cms::MessageProducer** *producer)

Execute an action given a session and producer.

6.462.1 Constructor & Destructor Documentation

6.462.1.1 `activemq::cmsutil::CmsTemplate::SendExecutor::SendExecutor (MessageCreator * messageCreator, CmsTemplate * parent) [inline]`

6.462.1.2 `virtual activemq::cmsutil::CmsTemplate::SendExecutor::~~SendExecutor () throw () [inline, virtual]`

6.462.2 Member Function Documentation

6.462.2.1 `virtual void activemq::cmsutil::CmsTemplate::SendExecutor::doInCms (cms::Session * session, cms::MessageProducer * producer) [inline, virtual]`

Execute an action given a session and producer.

Parameters

<i>session</i>	the CMS Session
<i>producer</i>	the CMS Producer

Exceptions

<i>cms::CMS-Exception</i> (p. 826)	if thrown by CMS API methods
--	------------------------------

Implements `activemq::cmsutil::ProducerCallback` (p. 2179).

The documentation for this class was generated from the following file:

- `src/main/activemq/cmsutil/CmsTemplate.h`

6.463 decaf::net::ServerSocket Class Reference

This class implements server sockets.

```
#include <src/main/decaf/net/ServerSocket.h>
```

Inheritance diagram for `decaf::net::ServerSocket`:

Public Member Functions

- **`ServerSocket ()`**
Creates a non-bound server socket.

- **ServerSocket** (int port)
Creates a new **ServerSocket** (p. 2342) bound to the specified port, if the value of port is 0, then any free port is chosen.
- **ServerSocket** (int port, int backlog)
Creates a new **ServerSocket** (p. 2342) bound to the specified port, if the value of port is 0, then any free port is chosen.
- **ServerSocket** (int port, int backlog, const **InetAddress** *address)
Creates a new **ServerSocket** (p. 2342) bound to the specified port, if the value of port is 0, then any free port is chosen.
- virtual **~ServerSocket** ()
Releases socket handle if **close()** (p. 2348) hasn't been called.
- virtual void **bind** (const std::string &host, int port)
Bind and listen to given local IPAddress and port, if the address is empty than a valid local address will be chosen, and if the port of 0 than an available open port will be chosen.
- virtual void **bind** (const std::string &host, int port, int backlog)
Bind and listen to given local IPAddress and port, if the address is empty than a valid local address will be chosen, and if the port of 0 than an available open port will be chosen.
- virtual **Socket** * **accept** ()
Listens for a connection request on the bound IPAddress and Port for this **ServerSocket** (p. 2342), the caller blocks until a connection is made.
- virtual void **close** ()
Closes the server socket, causing any Threads blocked on an accept call to throw an Exception.
- virtual bool **isClosed** () const
- virtual bool **isBound** () const
- virtual int **getReceiveBufferSize** () const
Gets the receive buffer size for this socket, **SO_RCVBUF**.
- virtual void **setReceiveBufferSize** (int size)
Sets the receive buffer size for this socket, **SO_RCVBUF**.
- virtual bool **getReuseAddress** () const
Gets the reuse address flag, **SO_REUSEADDR**.
- virtual void **setReuseAddress** (bool reuse)
Sets the reuse address flag, **SO_REUSEADDR**.
- virtual int **getSoTimeout** () const
Gets the timeout for socket operations, **SO_TIMEOUT**.
- virtual void **setSoTimeout** (int timeout)
Sets the timeout for socket operations, **SO_TIMEOUT**.
- virtual int **getLocalPort** () const
Gets the port number on the Local machine that this **ServerSocket** (p. 2342) is bound to.
- virtual std::string **toString** () const

Static Public Member Functions

- static void **setSocketImplFactory** (**SocketImplFactory** *factory)
*Sets the instance of a **SocketImplFactory** (p. 2487) that the **ServerSocket** (p. 2342) class should use when new instances of this class are created.*

Protected Member Functions

- **ServerSocket** (**SocketImpl** *impl)
*Creates a **ServerSocket** (p. 2342) wrapping the provided **SocketImpl** (p. 2479) instance, this **Socket** (p. 2452) is considered unconnected.*
- virtual void **implAccept** (**Socket** *socket)
*Virtual method that allows a **ServerSocket** (p. 2342) subclass to override the accept call and provide its own **SocketImpl** (p. 2479) for the socket.*
- virtual int **getDefaultBacklog** ()
Allows a subclass to override what is considered the default backlog.
- void **checkClosed** () const
- void **ensureCreated** () const
- void **setupSocketImpl** (int port, int backlog, const **InetAddress** *ifAddress)

6.463.1 Detailed Description

This class implements server sockets.

A server socket waits for requests to come in over the network.

The actual work of the server socket is performed by an instance of the **SocketImpl** (p. 2479) class. An application can change the socket factory that creates the socket implementation to configure itself to create sockets of a particular type.

Since

1.0

6.463.2 Constructor & Destructor Documentation

6.463.2.1 `decaf::net::ServerSocket::ServerSocket ()`

Creates a non-bound server socket.

6.463.2.2 `decaf::net::ServerSocket::ServerSocket (int port)`

Creates a new **ServerSocket** (p. 2342) bound to the specified port, if the value of port is 0, then any free port is chosen.

When this constructor is called the size of the backlog queue is set at 50, connections that arrive after the backlog has been reached are refused.

If a **SocketImplFactory** (p. 2487) is registered then the createSocketImpl method on the factory will be called otherwise a default **SocketImpl** (p. 2479) is created.

Parameters

<i>port</i>	The port to bind the ServerSocket (p. 2342) to.
-------------	--

Exceptions

<i>IOException</i>	if there is an I/O error while performing this operation.
<i>IllegalArgument-Exception</i>	if the port value is negative or greater than 65535.

6.463.2.3 decaf::net::ServerSocket::ServerSocket (int *port*, int *backlog*)

Creates a new **ServerSocket** (p. 2342) bound to the specified port, if the value of port is 0, then any free port is chosen.

When this constructor is called the size of the backlog queue is set at backlog, connections that arrive after the backlog has been reached are refused. If backlog is zero or negative then the default backlog value of 50 is used.

If a **SocketImplFactory** (p. 2487) is registered then the createSocketImpl method on the factory will be called otherwise a default **SocketImpl** (p. 2479) is created.

Parameters

<i>port</i>	The port to bind the ServerSocket (p. 2342) to.
<i>backlog</i>	The the number of incoming connection attempts to queue before connections are refused.

Exceptions

<i>IOException</i>	if there is an I/O error while performing this operation.
<i>IllegalArgument-Exception</i>	if the port value is negative or greater than 65535.

6.463.2.4 decaf::net::ServerSocket::ServerSocket (int *port*, int *backlog*, const InetAddress * *address*)

Creates a new **ServerSocket** (p. 2342) bound to the specified port, if the value of port is 0, then any free port is chosen.

If the value of the ifAddress is empty or NULL then the ANY address is used.

When this constructor is called the size of the backlog queue is set at backlog, connections that arrive after the backlog has been reached are refused. If backlog is zero or negative then the default backlog value of 50 is used.

If a **SocketImplFactory** (p. 2487) is registered then the `createSocketImpl` method on the factory will be called otherwise a default **SocketImpl** (p. 2479) is created.

Parameters

<i>port</i>	The port to bind the ServerSocket (p. 2342) to.
<i>backlog</i>	The the number of incoming connection attempts to queue before connections are refused.
<i>ifAddress</i>	The IP Address to bind to on the local machine.

Exceptions

<i>IOException</i>	if there is an I/O error while performing this operation.
<i>IllegalArgumentException</i>	if the port value is negative or greater than 65535.

6.463.2.5 `virtual decaf::net::ServerSocket::~ServerSocket ()` [virtual]

Releases socket handle if `close()` (p. 2348) hasn't been called.

6.463.2.6 `decaf::net::ServerSocket::ServerSocket (SocketImpl * impl)` [protected]

Creates a **ServerSocket** (p. 2342) wrapping the provided **SocketImpl** (p. 2479) instance, this **Socket** (p. 2452) is considered unconnected.

The **ServerSocket** (p. 2342) class takes ownership of this **SocketImpl** (p. 2479) pointer and will delete it when the **Socket** (p. 2452) class is destroyed.

Parameters

<i>impl</i>	The SocketImpl (p. 2479) instance to wrap.
-------------	---

Exceptions

<i>NullPointerException</i>	if the passed SocketImpl (p. 2479) is Null.
-----------------------------	--

6.463.3 Member Function Documentation

6.463.3.1 `virtual Socket* decaf::net::ServerSocket::accept ()` [virtual]

Listens for a connection request on the bound IPAddress and Port for this **ServerSocket** (p. 2342), the caller blocks until a connection is made.

If the `SO_TIMEOUT` option is set this method could throw a **SocketTimeoutException** (p. 2493) if the operation times out.

Returns

a new **Socket** (p. 2452) object pointer. Never returns NULL, the returned pointer is owned by the caller and must be explicitly freed by them.

Exceptions

<i>IOException</i>	if an I/O error occurs while binding the socket.
SocketException (p. 2471)	if an error occurs while blocking on the accept call.
SocketTimeoutException (p. 2493)	if the SO_TIMEOUT option was used and the accept timed out.

Reimplemented in **decaf::internal::net::ssl::openssl::OpenSSLServerSocket** (p. 2006).

6.463.3.2 `virtual void decaf::net::ServerSocket::bind (const std::string & host, int port)`
[virtual]

Bind and listen to given local IPAddress and port, if the address is empty than a valid local address will be chosen, and if the port of 0 than an available open port will be chosen.

Parameters

<i>host</i>	The IP address or host name.
<i>port</i>	The TCP port between 1..65535.

Exceptions

<i>IOException</i>	if an I/O error occurs while binding the socket.
<i>IllegalArgumentException</i>	if the parameters are not valid.

6.463.3.3 `virtual void decaf::net::ServerSocket::bind (const std::string & host, int port, int backlog)` [virtual]

Bind and listen to given local IPAddress and port, if the address is empty than a valid local address will be chosen, and if the port of 0 than an available open port will be chosen.

If the backlog is greater than zero it will be used instead of the default value, otherwise the default value is used and no error is generated.

Parameters

<i>host</i>	The IP address or host name.
-------------	------------------------------

<i>port</i>	The TCP port between 1..65535.
<i>backlog</i>	The size of listen backlog.

Exceptions

<i>IOException</i>	if an I/O error occurs while binding the socket.
<i>IllegalArgument-Exception</i>	if the parameters are not valid.

6.463.3.4 **void decaf::net::ServerSocket::checkClosed () const** [protected]

6.463.3.5 **virtual void decaf::net::ServerSocket::close ()** [virtual]

Closes the server socket, causing any Threads blocked on an accept call to throw an Exception.

Exceptions

<i>IOException</i>	if an I/O error occurs while performing this operation.
--------------------	---

6.463.3.6 **void decaf::net::ServerSocket::ensureCreated () const** [protected]

6.463.3.7 **virtual int decaf::net::ServerSocket::getDefaultBacklog ()**
[protected, virtual]

Allows a subclass to override what is considered the default backlog.

Returns

the default backlog for connections.

6.463.3.8 **virtual int decaf::net::ServerSocket::getLocalPort () const** [virtual]

Gets the port number on the Local machine that this **ServerSocket** (p. 2342) is bound to.

Returns

the port number of this machine that is bound, if not bound returns -1.

6.463.3.9 **virtual int decaf::net::ServerSocket::getReceiveBufferSize () const**
[virtual]

Gets the receive buffer size for this socket, SO_RCVBUF.

This is the buffer used by the underlying platform socket to buffer received data.

Returns

the receive buffer size in bytes.

Exceptions

<i>SocketException</i> (p. 2471)	if the operation fails.
--	-------------------------

6.463.3.10 virtual bool decaf::net::ServerSocket::getReuseAddress () const
[virtual]

Gets the reuse address flag, SO_REUSEADDR.

Returns

True if the address can be reused.

Exceptions

<i>SocketException</i> (p. 2471)	if the operation fails.
--	-------------------------

6.463.3.11 virtual int decaf::net::ServerSocket::getSoTimeout () const
[virtual]

Gets the timeout for socket operations, SO_TIMEOUT.

Returns

The timeout in milliseconds for socket operations.

Exceptions

<i>SocketException</i> (p. 2471)	Thrown if unable to retrieve the information.
--	---

6.463.3.12 virtual void decaf::net::ServerSocket::implAccept (Socket * socket)
[protected, virtual]

Virtual method that allows a **ServerSocket** (p.2342) subclass to override the accept call and provide its own **SocketImpl** (p. 2479) for the socket.

Parameters

<i>socket</i>	The socket object whose SocketImpl (p. 2479) should be used for the accept call.
---------------	---

Exceptions

<i>IOException</i>	if an I/O error occurs while performing this operation.
--------------------	---

6.463.3.13 `virtual bool decaf::net::ServerSocket::isBound () const` [virtual]

Returns

true if the server socket is bound.

6.463.3.14 `virtual bool decaf::net::ServerSocket::isClosed () const` [virtual]

Returns

true if the close method has been called on the **ServerSocket** (p. 2342).

6.463.3.15 `virtual void decaf::net::ServerSocket::setReceiveBufferSize (int size)`
[virtual]

Sets the receive buffer size for this socket, SO_RCVBUF.

Parameters

<i>size</i>	Number of bytes to set the receive buffer to.
-------------	---

Exceptions

SocketException (p. 2471)	if the operation fails.
<i>IllegalArgumentException</i>	if the value is zero or negative.

6.463.3.16 `virtual void decaf::net::ServerSocket::setReuseAddress (bool reuse)`
[virtual]

Sets the reuse address flag, SO_REUSEADDR.

Parameters

<i>reuse</i>	If true, sets the flag.
--------------	-------------------------

Exceptions

SocketException (p. 2471)	if the operation fails.
-------------------------------------	-------------------------

6.463.3.17 static void decaf::net::ServerSocket::setSocketImplFactory (SocketImplFactory * *factory*) [static]

Sets the instance of a **SocketImplFactory** (p. 2487) that the **ServerSocket** (p. 2342) class should use when new instances of this class are created.

This method is only allowed to be used once during the lifetime of the application.

Parameters

<i>factory</i>	The instance of a SocketImplFactory (p. 2487) to use when new - SocketImpl (p. 2479) objects are created.
----------------	---

Exceptions

<i>IOException</i>	if an I/O error occurs while performing this operation.
SocketException (p. 2471)	if this method has already been called with a valid factory.

6.463.3.18 virtual void decaf::net::ServerSocket::setSoTimeout (int *timeout*) [virtual]

Sets the timeout for socket operations, SO_TIMEOUT.

A value of zero indicates that timeout is infinite for operations on this socket.

Parameters

<i>timeout</i>	The timeout in milliseconds for socket operations.
----------------	--

Exceptions

SocketException (p. 2471)	Thrown if unable to set the information.
<i>IllegalArgumentException</i>	if the timeout value is negative.

6.463.3.19 void decaf::net::ServerSocket::setupSocketImpl (int *port*, int *backlog*, const InetAddress * *ifAddress*) [protected]

6.463.3.20 `virtual std::string decaf::net::ServerSocket::toString () const`
`[virtual]`

Returns

a string representing this **ServerSocket** (p. 2342).

The documentation for this class was generated from the following file:

- `src/main/decaf/net/ServerSocket.h`

6.464 decaf::net::ServerSocketFactory Class Reference

Class used to create Server Sockets, subclasses can be created that create certain types of server sockets according to specific policies.

```
#include <src/main/decaf/net/ServerSocketFactory.h>
```

Inheritance diagram for `decaf::net::ServerSocketFactory`:

Public Member Functions

- `virtual ~ServerSocketFactory ()`
- `virtual ServerSocket * createServerSocket ()`
*Create a new **ServerSocket** (p. 2342) that is unbound.*
- `virtual ServerSocket * createServerSocket (int port)=0`
*Create a new **ServerSocket** (p. 2342) that is bound to the given port.*
- `virtual ServerSocket * createServerSocket (int port, int backlog)=0`
*Create a new **ServerSocket** (p. 2342) that is bound to the given port.*
- `virtual ServerSocket * createServerSocket (int port, int backlog, const InetAddress *address)=0`
*Create a new **ServerSocket** (p. 2342) that is bound to the given port.*

Static Public Member Functions

- `static ServerSocketFactory * getDefault ()`
*Returns the Default **ServerSocket** (p. 2342) factory, the pointer is owned by the Decaf runtime and should not be deleted by the caller.*

Protected Member Functions

- `ServerSocketFactory ()`

6.464.1 Detailed Description

Class used to create Server Sockets, subclasses can be created that create certain types of server sockets according to specific policies.

Since

1.0

6.464.2 Constructor & Destructor Documentation

6.464.2.1 **decaf::net::ServerSocketFactory::ServerSocketFactory** ()
[protected]

6.464.2.2 **virtual decaf::net::ServerSocketFactory::~~ServerSocketFactory** ()
[virtual]

6.464.3 Member Function Documentation

6.464.3.1 **virtual ServerSocket* decaf::net::ServerSocketFactory::createServerSocket** () [virtual]

Create a new **ServerSocket** (p. 2342) that is unbound.

The **ServerSocket** (p. 2342) will have been configured with the defaults from the factory.

Returns

new **ServerSocket** (p. 2342) pointer that is owned by the caller.

Exceptions

<i>IOException</i>	if the ServerSocket (p. 2342) cannot be created for some reason.
--------------------	---

Reimplemented in **decaf::internal::net::ssl::openssl::OpenSSLServerSocketFactory** (p. 2012), **decaf::internal::net::ssl::DefaultSSLServerSocketFactory** (p. 1167), and **decaf::internal::net::DefaultServerSocketFactory** (p. 1157).

6.464.3.2 **virtual ServerSocket* decaf::net::ServerSocketFactory::createServerSocket** (int *port*) [pure virtual]

Create a new **ServerSocket** (p. 2342) that is bound to the given port.

The **ServerSocket** (p. 2342) will have been configured with the defaults from the factory.

Parameters

<i>port</i>	The port to bind the ServerSocket (p. 2342) to.
-------------	--

Returns

new **ServerSocket** (p. 2342) pointer that is owned by the caller.

Exceptions

<i>IOException</i>	if the ServerSocket (p. 2342) cannot be created for some reason.
--------------------	---

Implemented in **decaf::internal::net::ssl::openssl::OpenSSLServerSocketFactory** (p. 2012), **decaf::internal::net::ssl::DefaultSSLServerSocketFactory** (p. 1168), and **decaf::internal::net::DefaultServerSocketFactory** (p. 1157).

```
6.464.3.3  virtual ServerSocket* decaf::net::ServerSocketFactory-
           ::createServerSocket ( int port, int backlog ) [pure
           virtual]
```

Create a new **ServerSocket** (p. 2342) that is bound to the given port.

The **ServerSocket** (p. 2342) will have been configured with the defaults from the factory.

The **ServerSocket** (p. 2342) will use the specified connection backlog setting.

Parameters

<i>port</i>	The port to bind the ServerSocket (p. 2342) to.
<i>backlog</i>	The number of pending connect request the ServerSocket (p. 2342) can queue.

Returns

new **ServerSocket** (p. 2342) pointer that is owned by the caller.

Exceptions

<i>IOException</i>	if the ServerSocket (p. 2342) cannot be created for some reason.
--------------------	---

Implemented in **decaf::internal::net::ssl::openssl::OpenSSLServerSocketFactory** (p. 2012), **decaf::internal::net::ssl::DefaultSSLServerSocketFactory** (p. 1168), and **decaf::internal::net::DefaultServerSocketFactory** (p. 1158).

```
6.464.3.4  virtual ServerSocket* decaf::net::ServerSocketFactory::createServer-
           Socket ( int port, int backlog, const InetAddress * address ) [pure
           virtual]
```

Create a new **ServerSocket** (p. 2342) that is bound to the given port.

The **ServerSocket** (p. 2342) will have been configured with the defaults from the factory. The **ServerSocket** (p. 2342) will bind to the specified interface on the local host, and accept connections only on that interface. If the address parameter is NULL than the **ServerSocket** (p. 2342) will listen on all interfaces.

Parameters

<i>port</i>	The port to bind the ServerSocket (p. 2342) to.
<i>backlog</i>	The number of pending connect request the ServerSocket (p. 2342) can queue.
<i>address</i>	The address of the interface on the local machine to bind to.

Returns

new **ServerSocket** (p. 2342) pointer that is owned by the caller.

Exceptions

<i>IOException</i>	if the ServerSocket (p. 2342) cannot be created for some reason.
--------------------	---

Implemented in **decaf::internal::net::ssl::openssl::OpenSSLServerSocketFactory** (p. 2013), **decaf::internal::net::ssl::DefaultSSLServerSocketFactory** (p. 1169), and **decaf::internal::net::DefaultServerSocketFactory** (p. 1158).

6.464.3.5 static ServerSocketFactory* decaf::net::ServerSocketFactory::get-Default () [static]

Returns the Default **ServerSocket** (p. 2342) factory, the pointer is owned by the Decaf runtime and should not be deleted by the caller.

Only one default **ServerSocketFactory** (p. 2352) exists for the lifetime of the - Application.

Returns

the default **ServerSocketFactory** (p. 2352) for this application.

Reimplemented in **decaf::net::ssl::SSLServerSocketFactory** (p. 2512).

The documentation for this class was generated from the following file:

- src/main/decaf/net/**ServerSocketFactory.h**

6.465 activemq::util::Service Class Reference

Base interface for all classes that run as a **Service** (p. 2355) inside the application.

```
#include <src/main/activemq/util/Service.h>
```

Inheritance diagram for `activemq::util::Service`:

Public Member Functions

- virtual `~Service()`
- virtual void `start()`=0
- virtual void `stop()`=0

6.465.1 Detailed Description

Base interface for all classes that run as a **Service** (p. 2355) inside the application.

Since

3.3.0

6.465.2 Constructor & Destructor Documentation

6.465.2.1 virtual `activemq::util::Service::~Service()` [virtual]

6.465.3 Member Function Documentation

6.465.3.1 virtual void `activemq::util::Service::start()` [pure virtual]

Implemented in `activemq::util::ServiceSupport` (p. 2361).

6.465.3.2 virtual void `activemq::util::Service::stop()` [pure virtual]

Implemented in `activemq::util::ServiceSupport` (p. 2361).

The documentation for this class was generated from the following file:

- `src/main/activemq/util/Service.h`

6.466 `activemq::util::ServiceListener` Class Reference

Listener interface for observers of **Service** (p. 2355) related events.

```
#include <src/main/activemq/util/ServiceListener.h>
```


Public Member Functions

- virtual `~ServiceListener()`
- virtual void **started** (const **Service** *target)=0
indicates that the target service has completed its start operation.
- virtual void **stopped** (const **Service** *target)=0
indicates that the target service has completed its stop operation.

6.466.1 Detailed Description

Listener interface for observers of **Service** (p. 2355) related events.

Since

3.3.0

6.466.2 Constructor & Destructor Documentation

6.466.2.1 virtual `activemq::util::ServiceListener::~ServiceListener()`
[virtual]

6.466.3 Member Function Documentation

6.466.3.1 virtual void `activemq::util::ServiceListener::started (const Service * target)` [pure virtual]

indicates that the target service has completed its start operation.

Parameters

<i>target</i>	The service that triggered this notification.
---------------	---

6.466.3.2 virtual void `activemq::util::ServiceListener::stopped (const Service * target)` [pure virtual]

indicates that the target service has completed its stop operation.

Parameters

<i>target</i>	The service that triggered this notification.
---------------	---

The documentation for this class was generated from the following file:

- `src/main/activemq/util/ServiceListener.h`

6.467 activemq::util::ServiceStopper Class Reference

```
#include <src/main/activemq/util/ServiceStopper.h>
```

Public Member Functions

- **ServiceStopper** ()
- virtual **~ServiceStopper** ()
- void **stop** (**Service** *service)
- void **throwFirstException** ()
- virtual void **onException** (**Service** *service, **decaf::lang::Exception** &ex)

6.467.1 Constructor & Destructor Documentation

6.467.1.1 **activemq::util::ServiceStopper::ServiceStopper** ()

6.467.1.2 **virtual activemq::util::ServiceStopper::~~ServiceStopper** ()
[virtual]

6.467.2 Member Function Documentation

6.467.2.1 **virtual void activemq::util::ServiceStopper::onException** (**Service** *
service, **decaf::lang::Exception** & *ex*) [virtual]

6.467.2.2 **void activemq::util::ServiceStopper::stop** (**Service** * *service*)

6.467.2.3 **void activemq::util::ServiceStopper::throwFirstException** ()

The documentation for this class was generated from the following file:

- src/main/activemq/util/**ServiceStopper.h**

6.468 activemq::util::ServiceSupport Class Reference

Provides a base class for **Service** (p. 2355) implementations.

```
#include <src/main/activemq/util/ServiceSupport.h>
```

Inheritance diagram for **activemq::util::ServiceSupport**:

Public Member Functions

- **ServiceSupport** (const **ServiceSupport** &)

- **ServiceSupport & operator=** (const **ServiceSupport** &)
- **ServiceSupport** ()
- virtual **~ServiceSupport** ()
- void **start** ()
*Starts the **Service** (p. 2355), notifying any registered listeners of the start if it is successful.*
- void **stop** ()
*Stops the **Service** (p. 2355).*
- bool **isStarted** () const
- bool **isStopping** () const
- bool **isStopped** () const
- void **addServiceListener** (**ServiceListener** *listener)
*Adds the given listener to this **Service** (p. 2355)'s list of listeners, call retains ownership of the pointer.*
- void **removeServiceListener** (**ServiceListener** *listener)
*Removes the given listener to this **Service** (p. 2355)'s list of listeners, call retains ownership of the pointer.*

Static Public Member Functions

- static void **dispose** (**Service** *service)
Safely shuts down a service.

Protected Member Functions

- virtual void **doStop** (**ServiceStopper** *stopper)=0
Performs the actual stop operation on the service, ensuring that all resources held are released, must be implemented in derived class.
- virtual void **doStart** ()=0
Performs the actual start operation on the service, acquiring all the resources needed to run the service.

6.468.1 Detailed Description

Provides a base class for **Service** (p. 2355) implementations.

Since

3.3.0

6.468.2 Constructor & Destructor Documentation

6.468.2.1 `activemq::util::ServiceSupport::ServiceSupport (const ServiceSupport &)`

6.468.2.2 `activemq::util::ServiceSupport::ServiceSupport ()`

6.468.2.3 `virtual activemq::util::ServiceSupport::~~ServiceSupport ()`
[virtual]

6.468.3 Member Function Documentation

6.468.3.1 `void activemq::util::ServiceSupport::addServiceListener (ServiceListener * listener)`

Adds the given listener to this **Service** (p. 2355)'s list of listeners, call retains ownership of the pointer.

6.468.3.2 `static void activemq::util::ServiceSupport::dispose (Service * service)`
[static]

Safely shuts down a service.

Parameters

<i>service</i>	The service to stop.
----------------	----------------------

6.468.3.3 `virtual void activemq::util::ServiceSupport::doStart ()` [protected, pure virtual]

Performs the actual start operation on the service, acquiring all the resources needed to run the service.

Must be implemented in derived class.

Implemented in **activemq::threads::Scheduler** (p. 2318).

6.468.3.4 `virtual void activemq::util::ServiceSupport::doStop (ServiceStopper * stopper)` [protected, pure virtual]

Performs the actual stop operation on the service, ensuring that all resources held are released, must be implemented in derived class.

Implemented in **activemq::threads::Scheduler** (p. 2319).

6.468.3.5 `bool activemq::util::ServiceSupport::isStarted () const`

Returns

true if this service has been started

6.468.3.6 **bool** `activemq::util::ServiceSupport::isStopped ()` **const**

Returns

true if this service is closed

6.468.3.7 **bool** `activemq::util::ServiceSupport::isStopping ()` **const**

Returns

true if this service is in the process of closing

6.468.3.8 **ServiceSupport&** `activemq::util::ServiceSupport::operator= (const ServiceSupport &)`

6.468.3.9 **void** `activemq::util::ServiceSupport::removeServiceListener (ServiceListener * listener)`

Removes the given listener to this **Service** (p. 2355)'s list of listeners, call retains ownership of the pointer.

6.468.3.10 **void** `activemq::util::ServiceSupport::start ()` `[virtual]`

Starts the **Service** (p. 2355), notifying any registered listeners of the start if it is successful.

Implements **activemq::util::Service** (p. 2356).

6.468.3.11 **void** `activemq::util::ServiceSupport::stop ()` `[virtual]`

Stops the **Service** (p. 2355).

Implements **activemq::util::Service** (p. 2356).

The documentation for this class was generated from the following file:

- `src/main/activemq/util/ServiceSupport.h`

6.469 cms::Session Class Reference

A **Session** (p. 2361) object is a single-threaded context for producing and consuming messages.

```
#include <src/main/cms/Session.h>
```

Inheritance diagram for cms::Session:

Public Types

- enum **AcknowledgeMode** { **AUTO_ACKNOWLEDGE**, **DUPS_OK_ACKNOWLEDGE**, **CLIENT_ACKNOWLEDGE**, **SESSION_TRANSACTED**, **INDIVIDUAL_ACKNOWLEDGE** }

Public Member Functions

- virtual **~Session** () throw ()
- virtual void **close** ()=0
Closes this session as well as any active child consumers or producers.
- virtual void **commit** ()=0
Commits all messages done in this transaction and releases any locks currently held.
- virtual void **rollback** ()=0
Rolls back all messages done in this transaction and releases any locks currently held.
- virtual void **recover** ()=0
Stops message delivery in this session, and restarts message delivery with the oldest unacknowledged message.
- virtual **MessageConsumer** * **createConsumer** (const **Destination** *destination)=0

*Creates a **MessageConsumer** (p. 1877) for the specified destination.*
- virtual **MessageConsumer** * **createConsumer** (const **Destination** *destination, const std::string &selector)=0
*Creates a **MessageConsumer** (p. 1877) for the specified destination, using a message selector.*
- virtual **MessageConsumer** * **createConsumer** (const **Destination** *destination, const std::string &selector, bool noLocal)=0
*Creates a **MessageConsumer** (p. 1877) for the specified destination, using a message selector.*
- virtual **MessageConsumer** * **createDurableConsumer** (const **Topic** *destination, const std::string &name, const std::string &selector, bool noLocal=false)=0
*Creates a durable subscriber to the specified topic, using a **Message** (p. 1839) selector.*
- virtual **MessageProducer** * **createProducer** (const **Destination** *destination=NULL)=0
*Creates a **MessageProducer** (p. 1924) to send messages to the specified destination.*
- virtual **QueueBrowser** * **createBrowser** (const cms::Queue *queue)=0
*Creates a new **QueueBrowser** (p. 2227) to peek at Messages on the given **Queue** (p. 2221).*

- virtual **QueueBrowser** * **createBrowser** (const cms::Queue *queue, const std::string &selector)=0
*Creates a new **QueueBrowser** (p. 2227) to peek at Messages on the given **Queue** (p. 2221).*
- virtual **Queue** * **createQueue** (const std::string &queueName)=0
*Creates a queue identity given a **Queue** (p. 2221) name.*
- virtual **Topic** * **createTopic** (const std::string &topicName)=0
*Creates a topic identity given a **Queue** (p. 2221) name.*
- virtual **TemporaryQueue** * **createTemporaryQueue** ()=0
*Creates a **TemporaryQueue** (p. 2698) object.*
- virtual **TemporaryTopic** * **createTemporaryTopic** ()=0
*Creates a **TemporaryTopic** (p. 2700) object.*
- virtual **Message** * **createMessage** ()=0
*Creates a new **Message** (p. 1839).*
- virtual **BytesMessage** * **createBytesMessage** ()=0
*Creates a **BytesMessage** (p. 718).*
- virtual **BytesMessage** * **createBytesMessage** (const unsigned char *bytes, int bytesSize)=0
*Creates a **BytesMessage** (p. 718) and sets the payload to the passed value.*
- virtual **StreamMessage** * **createStreamMessage** ()=0
*Creates a new **StreamMessage** (p. 2606).*
- virtual **TextMessage** * **createTextMessage** ()=0
*Creates a new **TextMessage** (p. 2701).*
- virtual **TextMessage** * **createTextMessage** (const std::string &text)=0
*Creates a new **TextMessage** (p. 2701) and set the text to the value given.*
- virtual **MapMessage** * **createMapMessage** ()=0
*Creates a new **MapMessage** (p. 1779).*
- virtual **AcknowledgeMode** **getAcknowledgeMode** () const =0
Returns the acknowledgment mode of the session.
- virtual bool **isTransacted** () const =0
*Gets if the Sessions is a Transacted **Session** (p. 2361).*
- virtual void **unsubscribe** (const std::string &name)=0
Unsubscribes a durable subscription that has been created by a client.

6.469.1 Detailed Description

A **Session** (p. 2361) object is a single-threaded context for producing and consuming messages.

A session serves several purposes:

- It is a factory for its message producers and consumers.
- It supplies provider-optimized message factories.

- It is a factory for `TemporaryTopics` and `TemporaryQueues`.
- It provides a way to create **Queue** (p. 2221) or **Topic** (p. 2764) objects for those clients that need to dynamically manipulate provider-specific destination names.
- It supports a single series of transactions that combine work spanning its producers and consumers into atomic units.
- It defines a serial order for the messages it consumes and the messages it produces.
- It retains messages it consumes until they have been acknowledged.
- It serializes execution of message listeners registered with its message consumers.

A session can create and service multiple message producers and consumers.

One typical use is to have a thread block on a synchronous **MessageConsumer** (p. 1877) until a message arrives. The thread may then use one or more of the **Session** (p. 2361)'s `MessageProducers`.

If a client desires to have one thread produce messages while others consume them, the client should use a separate session for its producing thread.

Certain rules apply to a session's `close` method and are detailed below.

- There is no need to close the producers and consumers of a closed session.
- The close call will block until a receive call or message listener in progress has completed. A blocked message consumer receive call returns null when this session is closed.
- Closing a transacted session must roll back the transaction in progress.
- The close method is the only **Session** (p. 2361) method that can be called concurrently.
- Invoking any other **Session** (p. 2361) method on a closed session must throw an **IllegalStateException** (p. 1419). Closing a closed session must not throw any exceptions.

Transacted Sessions

When a **Session** (p. 2361) is created it can be set to operate in a Transaction based mode. Each **Session** (p. 2361) then operates in a single transaction for all Producers and Consumers of that **Session** (p. 2361). Messages sent and received within a Transaction are grouped into an atomic unit that is committed or rolled back together.

For a **MessageProducer** (p. 1924) this implies that all messages sent by the producer are not sent to the Provider until the commit call is made. Rolling back the Transaction results in all produced Messages being dropped.

For a **MessageConsumer** (p. 1877) this implies that all received messages are not Acknowledged until the Commit call is made. Rolling back the Transaction results in all

Consumed **Message** (p. 1839) being redelivered to the client, the Provider may allow configuration that limits the Maximum number of redeliveries for a **Message** (p. 1839).

While the **Session** (p.2361) interface implements the **Startable** (p.2534) and **Stoppable** (p.2602) interfaces it is not required to implement these methods and can throw an `UnsupportedOperationException` exception if they are not available for the given CMS provider.

Since

1.0

6.469.2 Member Enumeration Documentation

6.469.2.1 enum cms::Session::AcknowledgeMode

Enumerator:

AUTO_ACKNOWLEDGE With this acknowledgment mode, the session automatically acknowledges a client's receipt of a message either when the session has successfully returned from a call to receive or when the message listener the session has called to process the message successfully returns.

DUPS_OK_ACKNOWLEDGE With this acknowledgment mode, the session automatically acknowledges a client's receipt of a message either when the session has successfully returned from a call to receive or when the message listener the session has called to process the message successfully returns. Acknowledgments may be delayed in this mode to increase performance at the cost of the message being redelivered this client fails.

CLIENT_ACKNOWLEDGE With this acknowledgment mode, the client acknowledges a consumed message by calling the message's `acknowledge` method.

SESSION_TRANSACTED Messages will be consumed when the transaction commits.

INDIVIDUAL_ACKNOWLEDGE **Message** (p. 1839) will be acknowledged individually. Normally the acks sent acknowledge the given message and all messages received before it, this mode only acknowledges one message.

6.469.3 Constructor & Destructor Documentation

6.469.3.1 virtual cms::Session::~~Session () throw () [virtual]

6.469.4 Member Function Documentation

6.469.4.1 virtual void cms::Session::close () [pure virtual]

Closes this session as well as any active child consumers or producers.

Exceptions

<i>CMSException</i> (p. 826)	- If an internal error occurs.
--	--------------------------------

Implements **cms::Closeable** (p. 816).

Implemented in **activemq::core::ActiveMQSession** (p. 338), and **activemq::cmsutil::PooledSession** (p. 2095).

6.469.4.2 `virtual void cms::Session::commit()` [pure virtual]

Commits all messages done in this transaction and releases any locks currently held.

Exceptions

<i>CMSException</i> (p. 826)	- If an internal error occurs.
<i>IllegalStateException</i> (p. 1419)	- if the method is not called by a transacted session.

Implemented in **activemq::core::ActiveMQSession** (p. 338), **activemq::cmsutil::PooledSession** (p. 2095), and **activemq::core::ActiveMQXASession** (p. 437).

Referenced by **activemq::cmsutil::PooledSession::commit()**.

6.469.4.3 `virtual QueueBrowser* cms::Session::createBrowser(const cms::Queue * queue)` [pure virtual]

Creates a new **QueueBrowser** (p. 2227) to peek at Messages on the given **Queue** (p. 2221).

Parameters

<i>queue</i>	the Queue (p. 2221) to browse
--------------	--------------------------------------

Returns

New **QueueBrowser** (p. 2227) that is owned by the caller.

Exceptions

<i>CMSException</i> (p. 826)	- If an internal error occurs.
<i>InvalidDestinationException</i> (p. 1535)	- if the destination given is invalid.

Implemented in **activemq::core::ActiveMQSession** (p. 339), and **activemq::cmsutil::PooledSession** (p. 2096).

6.469.4.4 virtual **QueueBrowser*** cms::Session::createBrowser (const cms::Queue * queue, const std::string & selector) [pure virtual]

Creates a new **QueueBrowser** (p. 2227) to peek at Messages on the given **Queue** (p. 2221).

Parameters

<i>queue</i>	the Queue (p. 2221) to browse
<i>selector</i>	the Message (p. 1839) selector to filter which messages are browsed.

Returns

New **QueueBrowser** (p. 2227) that is owned by the caller.

Exceptions

CMSException (p. 826)	- If an internal error occurs.
InvalidDestination-Exception (p. 1535)	- if the destination given is invalid.

Implemented in **activemq::core::ActiveMQSession** (p. 339), and **activemq::cmsutil::PooledSession** (p. 2096).

6.469.4.5 virtual **BytesMessage*** cms::Session::createBytesMessage () [pure virtual]

Creates a **BytesMessage** (p. 718).

Exceptions

CMSException (p. 826)	- If an internal error occurs.
---------------------------------	--------------------------------

Implemented in **activemq::core::ActiveMQSession** (p. 340), and **activemq::cmsutil::PooledSession** (p. 2097).

Referenced by **activemq::cmsutil::PooledSession::createBytesMessage()**.

6.469.4.6 `virtual BytesMessage* cms::Session::createBytesMessage (const unsigned char * bytes, int bytesSize) [pure virtual]`

Creates a **BytesMessage** (p. 718) and sets the payload to the passed value.

Parameters

<i>bytes</i>	an array of bytes to set in the message
<i>bytesSize</i>	the size of the bytes array, or number of bytes to use

Exceptions

<i>CMSException</i> (p. 826)	- If an internal error occurs.
--	--------------------------------

Implemented in **activemq::core::ActiveMQSession** (p. 340), and **activemq::cmsutil::PooledSession** (p. 2097).

6.469.4.7 `virtual MessageConsumer* cms::Session::createConsumer (const Destination * destination) [pure virtual]`

Creates a **MessageConsumer** (p. 1877) for the specified destination.

Parameters

<i>destination</i>	the Destination (p. 1210) that this consumer receiving messages for.
--------------------	---

Returns

pointer to a new **MessageConsumer** (p. 1877) that is owned by the caller (caller deletes)

Exceptions

<i>CMSException</i> (p. 826)	- If an internal error occurs.
<i>InvalidDestinationException</i> (p. 1535)	- if an invalid destination is specified.

Implemented in **activemq::core::ActiveMQSession** (p. 340), and **activemq::cmsutil::PooledSession** (p. 2098).

Referenced by `activemq::cmsutil::PooledSession::createConsumer()`.

6.469.4.8 virtual **MessageConsumer*** cms::Session::createConsumer (const **Destination** * *destination*, const std::string & *selector*) [pure virtual]

Creates a **MessageConsumer** (p. 1877) for the specified destination, using a message selector.

Parameters

<i>destination</i>	the Destination (p. 1210) that this consumer receiving messages for.
<i>selector</i>	the Message (p. 1839) Selector to use

Returns

pointer to a new **MessageConsumer** (p. 1877) that is owned by the caller (caller deletes)

Exceptions

CMSException (p. 826)	- If an internal error occurs.
InvalidDestination-Exception (p. 1535)	- if an invalid destination is specified.
InvalidSelector-Exception (p. 1541)	- if the message selector is invalid.

Implemented in **activemq::core::ActiveMQSession** (p. 341), and **activemq::cmsutil::PooledSession** (p. 2099).

6.469.4.9 virtual **MessageConsumer*** cms::Session::createConsumer (const **Destination** * *destination*, const std::string & *selector*, bool *noLocal*) [pure virtual]

Creates a **MessageConsumer** (p. 1877) for the specified destination, using a message selector.

Parameters

<i>destination</i>	the Destination (p. 1210) that this consumer receiving messages for.
<i>selector</i>	the Message (p. 1839) Selector to use
<i>noLocal</i>	if true, and the destination is a topic, inhibits the delivery of messages published by its own connection. The behavior for NoLocal is not specified if the destination is a queue.

Returns

pointer to a new **MessageConsumer** (p. 1877) that is owned by the caller (caller deletes)

Exceptions

<i>CMSException</i> (p. 826)	- If an internal error occurs.
<i>InvalidDestinationException</i> (p. 1535)	- if an invalid destination is specified.
<i>InvalidSelectorException</i> (p. 1541)	- if the message selector is invalid.

Implemented in **activemq::core::ActiveMQSession** (p. 341), and **activemq::cmsutil::PooledSession** (p. 2099).

6.469.4.10 **virtual MessageConsumer* cms::Session::createDurableConsumer (**
const Topic * destination, const std::string & name, const std::string & selector,
bool noLocal = false) [pure virtual]

Creates a durable subscriber to the specified topic, using a **Message** (p. 1839) selector.

Sessions that create durable consumers must use the same client Id as was used the last time the subscription was created in order to receive all messages that were delivered while the client was offline.

Parameters

<i>destination</i>	the topic to subscribe to
<i>name</i>	The name used to identify the subscription
<i>selector</i>	the Message (p. 1839) Selector to use
<i>noLocal</i>	if true, and the destination is a topic, inhibits the delivery of messages published by its own connection. The behavior for NoLocal is not specified if the destination is a queue.

Returns

pointer to a new durable **MessageConsumer** (p. 1877) that is owned by the caller
 (caller deletes)

Exceptions

<i>CMSException</i> (p. 826)	- If an internal error occurs.
<i>InvalidDestinationException</i> (p. 1535)	- if an invalid destination is specified.
<i>InvalidSelectorException</i> (p. 1541)	- if the message selector is invalid.

Implemented in **activemq::core::ActiveMQSession** (p. 342), and **activemq::cmsutil::PooledSession** (p. 2100).

Referenced by `activemq::cmsutil::PooledSession::createDurableConsumer()`.

6.469.4.11 `virtual MapMessage* cms::Session::createMapMessage ()` [pure virtual]

Creates a new **MapMessage** (p. 1779).

Exceptions

CMSEException (p. 826)	- If an internal error occurs.
----------------------------------	--------------------------------

Implemented in **activemq::core::ActiveMQSession** (p. 342), and **activemq::cmsutil::PooledSession** (p. 2101).

Referenced by `activemq::cmsutil::PooledSession::createMapMessage()`.

6.469.4.12 `virtual Message* cms::Session::createMessage ()` [pure virtual]

Creates a new **Message** (p. 1839).

Exceptions

CMSEException (p. 826)	- If an internal error occurs.
----------------------------------	--------------------------------

Implemented in **activemq::core::ActiveMQSession** (p. 343), and **activemq::cmsutil::PooledSession** (p. 2101).

Referenced by `activemq::cmsutil::PooledSession::createMessage()`.

6.469.4.13 `virtual MessageProducer* cms::Session::createProducer (const Destination * destination = NULL)` [pure virtual]

Creates a **MessageProducer** (p. 1924) to send messages to the specified destination.

Parameters

<i>destination</i>	the Destination (p. 1210) to send on
--------------------	---

Returns

New **MessageProducer** (p. 1924) that is owned by the caller.

Exceptions

<i>CMSException</i> (p. 826)	- If an internal error occurs.
<i>InvalidDestinationException</i> (p. 1535)	- if an invalid destination is specified.

Implemented in **activemq::core::ActiveMQSession** (p. 343), and **activemq::cmsutil::PooledSession** (p. 2101).

Referenced by `activemq::cmsutil::PooledSession::createProducer()`.

6.469.4.14 `virtual Queue* cms::Session::createQueue (const std::string & queueName) [pure virtual]`

Creates a queue identity given a **Queue** (p. 2221) name.

Parameters

<i>queueName</i>	the name of the new Queue (p. 2221)
------------------	--

Returns

new **Queue** (p. 2221) pointer that is owned by the caller.

Exceptions

<i>CMSException</i> (p. 826)	- If an internal error occurs.
--	--------------------------------

Implemented in **activemq::core::ActiveMQSession** (p. 343), and **activemq::cmsutil::PooledSession** (p. 2102).

Referenced by `activemq::cmsutil::PooledSession::createQueue()`.

6.469.4.15 `virtual StreamMessage* cms::Session::createStreamMessage () [pure virtual]`

Creates a new **StreamMessage** (p. 2606).

Exceptions

<i>CMSException</i> (p. 826)	- If an internal error occurs.
--	--------------------------------

Implemented in **activemq::core::ActiveMQSession** (p. 344), and **activemq::cmsutil::PooledSession** (p. 2102).

Referenced by `activemq::cmsutil::PooledSession::createStreamMessage()`.

6.469.4.16 `virtual TemporaryQueue* cms::Session::createTemporaryQueue ()`
`[pure virtual]`

Creates a **TemporaryQueue** (p. 2698) object.

Returns

new **TemporaryQueue** (p. 2698) pointer that is owned by the caller.

Exceptions

<i>CMSEException</i> (p. 826)	- If an internal error occurs.
---	--------------------------------

Implemented in **activemq::core::ActiveMQSession** (p. 344), and **activemq::cmsutil::PooledSession** (p. 2102).

Referenced by `activemq::cmsutil::PooledSession::createTemporaryQueue()`.

6.469.4.17 `virtual TemporaryTopic* cms::Session::createTemporaryTopic ()`
`[pure virtual]`

Creates a **TemporaryTopic** (p. 2700) object.

Exceptions

<i>CMSEException</i> (p. 826)	- If an internal error occurs.
---	--------------------------------

Implemented in **activemq::core::ActiveMQSession** (p. 344), and **activemq::cmsutil::PooledSession** (p. 2103).

Referenced by `activemq::cmsutil::PooledSession::createTemporaryTopic()`.

6.469.4.18 `virtual TextMessage* cms::Session::createTextMessage ()` `[pure virtual]`

Creates a new **TextMessage** (p. 2701).

Exceptions

<i>CMSEException</i> (p. 826)	- If an internal error occurs.
---	--------------------------------

Implemented in **activemq::core::ActiveMQSession** (p. 345), and **activemq::cmsutil::PooledSession** (p. 2103).

Referenced by `activemq::cmsutil::PooledSession::createTextMessage()`.

6.469.4.19 `virtual TextMessage* cms::Session::createTextMessage (const std::string & text) [pure virtual]`

Creates a new **TextMessage** (p. 2701) and set the text to the value given.

Parameters

<i>text</i>	the initial text for the message
-------------	----------------------------------

Exceptions

<i>CMSException</i> (p. 826)	- If an internal error occurs.
--	--------------------------------

Implemented in **`activemq::core::ActiveMQSession`** (p. 345), and **`activemq::cmsutil::PooledSession`** (p. 2103).

6.469.4.20 `virtual Topic* cms::Session::createTopic (const std::string & topicName) [pure virtual]`

Creates a topic identity given a **Queue** (p. 2221) name.

Parameters

<i>topicName</i>	the name of the new Topic (p. 2764)
------------------	--

Returns

new **Topic** (p. 2764) pointer that is owned by the caller.

Exceptions

<i>CMSException</i> (p. 826)	- If an internal error occurs.
--	--------------------------------

Implemented in **`activemq::core::ActiveMQSession`** (p. 345), and **`activemq::cmsutil::PooledSession`** (p. 2104).

Referenced by `activemq::cmsutil::PooledSession::createTopic()`.

6.469.4.21 `virtual AcknowledgeMode cms::Session::getAcknowledgeMode () const [pure virtual]`

Returns the acknowledgment mode of the session.

Returns

the Sessions Acknowledge Mode

Exceptions

<i>CMSException</i> (p. 826)	- If an internal error occurs.
--	--------------------------------

Implemented in **activemq::core::ActiveMQSession** (p. 347), and **activemq::cmsutil::PooledSession** (p. 2104).

Referenced by `activemq::cmsutil::PooledSession::getAcknowledgeMode()`.

6.469.4.22 `virtual bool cms::Session::isTransacted () const [pure virtual]`

Gets if the Sessions is a Transacted **Session** (p. 2361).

Returns

transacted true - false.

Exceptions

<i>CMSException</i> (p. 826)	- If an internal error occurs.
--	--------------------------------

Implemented in **activemq::core::ActiveMQSession** (p. 349), **activemq::cmsutil::PooledSession** (p. 2105), and **activemq::core::ActiveMQXASession** (p. 438).

Referenced by `activemq::cmsutil::PooledSession::isTransacted()`.

6.469.4.23 `virtual void cms::Session::recover () [pure virtual]`

Stops message delivery in this session, and restarts message delivery with the oldest unacknowledged message.

All consumers deliver messages in a serial order. Acknowledging a received message automatically acknowledges all messages that have been delivered to the client.

Restarting a session causes it to take the following actions:

- Stop message delivery
- Mark all messages that might have been delivered but not acknowledged as "re-delivered"
- Restart the delivery sequence including all unacknowledged messages that had been previously delivered. Redelivered messages do not have to be delivered in exactly their original delivery order.

Exceptions

<i>CMSException</i> (p. 826)	- if the CMS provider fails to stop and restart message delivery due to some internal error.
<i>IllegalStateException</i> (p. 1419)	- if the method is called by a transacted session.

Implemented in **activemq::core::ActiveMQSession** (p. 350), and **activemq::cmsutil::PooledSession** (p. 2105).

Referenced by `activemq::cmsutil::PooledSession::recover()`.

6.469.4.24 `virtual void cms::Session::rollback () [pure virtual]`

Rolls back all messages done in this transaction and releases any locks currently held.

Exceptions

<i>CMSException</i> (p. 826)	- If an internal error occurs.
<i>IllegalStateException</i> (p. 1419)	- if the method is not called by a transacted session.

Implemented in **activemq::core::ActiveMQSession** (p. 351), **activemq::cmsutil::PooledSession** (p. 2106), and **activemq::core::ActiveMQXASession** (p. 439).

Referenced by `activemq::cmsutil::PooledSession::rollback()`.

6.469.4.25 `virtual void cms::Session::unsubscribe (const std::string & name) [pure virtual]`

Unsubscribes a durable subscription that has been created by a client.

This method deletes the state being maintained on behalf of the subscriber by its provider. It is erroneous for a client to delete a durable subscription while there is an active **MessageConsumer** (p. 1877) or Subscriber for the subscription, or while a consumed message is part of a pending transaction or has not been acknowledged in the session.

Parameters

<i>name</i>	The name used to identify this subscription
-------------	---

Exceptions

<i>CMSException</i> (p. 826)	- If an internal error occurs.
--	--------------------------------

Implemented in **activemq::core::ActiveMQSession** (p. 353), and **activemq::cmsutil::PooledSession** (p. 2107).

Referenced by `activemq::cmsutil::PooledSession::unsubscribe()`.

The documentation for this class was generated from the following file:

- `src/main/cms/Session.h`

6.470 activemq::cmsutil::SessionCallback Class Reference

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

```
#include <src/main/activemq/cmsutil/SessionCallback.h>
```

Inheritance diagram for `activemq::cmsutil::SessionCallback`:

Public Member Functions

- virtual `~SessionCallback()`
- virtual void `doInCms(cms::Session *session)=0`
Execute any number of operations against the supplied CMS session.

6.470.1 Detailed Description

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

6.470.2 Constructor & Destructor Documentation

6.470.2.1 virtual `activemq::cmsutil::SessionCallback::~SessionCallback()`
[inline, virtual]

6.470.3 Member Function Documentation

6.470.3.1 virtual void `activemq::cmsutil::SessionCallback::doInCms(cms::Session * session)` [pure virtual]

Execute any number of operations against the supplied CMS session.

Parameters

<i>session</i>	the CMS Session
----------------	-----------------

Exceptions

<i>CMSEException</i> if thrown by CMS API methods

Implemented in **activemq::cmsutil::CmsTemplate::ReceiveExecutor** (p. 2249), and **activemq::cmsutil::CmsTemplate::ProducerExecutor** (p. 2181).

The documentation for this class was generated from the following file:

- src/main/activemq/cmsutil/**SessionCallback.h**

6.471 activemq::commands::SessionId Class Reference

```
#include <src/main/activemq/commands/SessionId.h>
```

Inheritance diagram for activemq::commands::SessionId:

Public Types

- typedef **decaf::lang::PointerComparator** < **SessionId** > **COMPARATOR**

Public Member Functions

- **SessionId** ()
- **SessionId** (const **SessionId** &other)
- **SessionId** (const **ConnectionId** *connectionId, long long sessionId)
- **SessionId** (const **ProducerId** *producerId)
- **SessionId** (const **ConsumerId** *consumerId)
- virtual ~**SessionId** ()
- virtual unsigned char **getDataStructureType** () const
Get the **DataStructure** (p. 1133) Type as defined in *CommandTypes.h*.
- virtual **SessionId** * **cloneDataStructure** () const
Clone this object and return a new instance that the caller now owns, this will be an exact copy of this one.
- virtual void **copyDataStructure** (const **DataStructure** *src)
Copy the contents of the passed object into this objects members, overwriting any existing data.
- virtual std::string **toString** () const
Returns a string containing the information for this **DataStructure** (p. 1133) such as its type and value of its elements.
- virtual bool **equals** (const **DataStructure** *value) const
Compares the **DataStructure** (p. 1133) passed in to this one, and returns if they are equivalent.
- const **Pointer**< **ConnectionId** > & **getParentId** () const

- virtual const std::string & **getConnectionId** () const
- virtual std::string & **getConnectionId** ()
- virtual void **setConnectionId** (const std::string &**connectionId**)
- virtual long long **getValue** () const
- virtual void **setValue** (long long **value**)
- virtual int **compareTo** (const **SessionId** &**value**) const
- virtual bool **equals** (const **SessionId** &**value**) const
- virtual bool **operator==** (const **SessionId** &**value**) const
- virtual bool **operator<** (const **SessionId** &**value**) const
- **SessionId** & **operator=** (const **SessionId** &other)

Static Public Attributes

- static const unsigned char **ID_SESSIONID** = 121

Protected Attributes

- std::string **connectionId**
- long long **value**

6.471.1 Member Typedef Documentation

- 6.471.1.1 `typedef decaf::lang::PointerComparator<SessionId>
activemq::commands::SessionId::COMPARATOR`

6.471.2 Constructor & Destructor Documentation

- 6.471.2.1 `activemq::commands::SessionId::SessionId ()`
- 6.471.2.2 `activemq::commands::SessionId::SessionId (const SessionId & other)`
- 6.471.2.3 `activemq::commands::SessionId::SessionId (const ConnectionId *
connectionId, long long sessionId)`
- 6.471.2.4 `activemq::commands::SessionId::SessionId (const ProducerId *
producerId)`
- 6.471.2.5 `activemq::commands::SessionId::SessionId (const ConsumerId *
consumerId)`
- 6.471.2.6 `virtual activemq::commands::SessionId::~~SessionId () [virtual]`

6.471.3 Member Function Documentation

6.471.3.1 `virtual SessionId* activemq::commands::SessionId::cloneDataStructure () const [virtual]`

Clone this obbject and return a new instance that the caller now owns, this will be an exact copy of this one.

Returns

new copy of this object.

Implements **activemq::commands::DataStructure** (p. 1133).

6.471.3.2 `virtual int activemq::commands::SessionId::compareTo (const SessionId & value) const [virtual]`

6.471.3.3 `virtual void activemq::commands::SessionId::copyDataStructure (const DataStructure * src) [virtual]`

Copy the contents of the passed object into this objects members, overwriting any existing data.

Returns

src - Source Object

Implements **activemq::commands::DataStructure** (p. 1134).

6.471.3.4 `virtual bool activemq::commands::SessionId::equals (const DataStructure * value) const [virtual]`

Compares the **DataStructure** (p. 1133) passed in to this one, and returns if they are equivalent.

Equivalent here means that they are of the same type, and that each element of the objects are the same.

Returns

true if **DataStructure** (p. 1133)'s are Equal.

Implements **activemq::commands::DataStructure** (p. 1135).

6.471.3.5 `virtual bool activemq::commands::SessionId::equals (const SessionId & value) const [virtual]`

6.471.3.6 `virtual const std::string& activemq::commands::SessionId::getConnectionId () const [virtual]`

6.471.3.7 virtual std::string& activemq::commands::SessionId::getConnectionId ()
[virtual]

6.471.3.8 virtual unsigned char activemq::commands::SessionId::getDataStructure-
Type () const [virtual]

Get the **DataSet** (p. 1133) Type as defined in CommandTypes.h.

Returns

The type of the data structure

Implements **activemq::commands::DataSet** (p. 1137).

6.471.3.9 const Pointer<ConnectionId>& activemq::commands::SessionId::get-
ParentId () const

6.471.3.10 virtual long long activemq::commands::SessionId::getValue () const
[virtual]

6.471.3.11 virtual bool activemq::commands::SessionId::operator< (const SessionId & *value*
) const [virtual]

6.471.3.12 SessionId& activemq::commands::SessionId::operator= (const SessionId &
other)

6.471.3.13 virtual bool activemq::commands::SessionId::operator== (const SessionId &
value) const [virtual]

6.471.3.14 virtual void activemq::commands::SessionId::setConnectionId (const
std::string & *connectionId*) [virtual]

6.471.3.15 virtual void activemq::commands::SessionId::setValue (long long *value*)
[virtual]

6.471.3.16 virtual std::string activemq::commands::SessionId::toString () const
[virtual]

Returns a string containing the information for this **DataSet** (p. 1133) such as its
type and value of its elements.

Returns

formatted string useful for debugging.

Reimplemented from **activemq::commands::BaseDataSet** (p. 531).

6.471.4 Field Documentation

6.471.4.1 `std::string activemq::commands::SessionId::connectionId`
[protected]

6.471.4.2 `const unsigned char activemq::commands::SessionId::ID_SESSIONID = 121`
[static]

Referenced by `activemq::state::CommandVisitorAdapter::processRemoveInfo()`.

6.471.4.3 `long long activemq::commands::SessionId::value` [protected]

The documentation for this class was generated from the following file:

- `src/main/activemq/commands/SessionId.h`

6.472 `activemq::wireformat::openwire::marshal::generated::SessionIdMarshaller` Class Reference

Marshaling code for Open Wire Format for **SessionIdMarshaller** (p. 2382).

```
#include <src/main/activemq/wireformat/openwire/marshal/generated/SessionIdMarshaller.h>
```

Inheritance diagram for `activemq::wireformat::openwire::marshal::generated::SessionIdMarshaller`:

Public Member Functions

- **SessionIdMarshaller** ()
- virtual **~SessionIdMarshaller** ()
- virtual **commands::DataStructure * createObject** () const
Creates a new instance of the class that this class is a marshaling director for.
- virtual unsigned char **getDataStructureType** () const
Gets the DataStructureType that this class marshals/unmarshals.
- virtual void **tightUnmarshal** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataInputStream** *dataIn, **utils::BooleanStream** *bs)
Tight Un-marhsal to the given stream.
- virtual int **tightMarshal1** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **utils::BooleanStream** *bs)
Tight Marhsal to the given stream.

- virtual void **tightMarshal2** (**OpenWireFormat** *wireFormat, **commands::Data-Structure** *dataStructure, **decaf::io::DataOutputStream** *dataOut, **utils::BooleanStream** *bs)

Tight Marshal to the given stream.

- virtual void **looseUnmarshal** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataInputStream** *dataIn)

Loose Un-marshal to the given stream.

- virtual void **looseMarshal** (**OpenWireFormat** *wireFormat, **commands::Data-Structure** *dataStructure, **decaf::io::DataOutputStream** *dataOut)

Tight Marshal to the given stream.

6.472.1 Detailed Description

Marshaling code for Open Wire Format for **SessionIdMarshaller** (p. 2382).

NOTE!: This file is auto generated - do not modify! if you need to make a change, please see the Java Classes in the activemq-openwire-generator module

6.472.2 Constructor & Destructor Documentation

6.472.2.1 **activemq::wireformat::openwire::marshal::generated-
::SessionIdMarshaller::SessionIdMarshaller ()**
[inline]

6.472.2.2 **virtual activemq::wireformat::openwire::marshal::generated:-
SessionIdMarshaller::~~SessionIdMarshaller ()** [inline,
virtual]

6.472.3 Member Function Documentation

6.472.3.1 **virtual commands::DataStructure* activemq::wireformat::openwire-
::marshal::generated::SessionIdMarshaller::createObject ()** const
[virtual]

Creates a new instance of the class that this class is a marshaling director for.

Returns

newly allocated Command

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller**
(p. 1120).

6.472.3.2 `virtual unsigned char activemq::wireformat::openwire::marshal-
::generated::SessionIdMarshaller::getDataStructureType () const
[virtual]`

Gets the DataStructureType that this class marshals/unmarshals.

Returns

byte Id of this classes DataStructureType

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller**
(p. 1122).

6.472.3.3 `virtual void activemq::wireformat::openwire::marshal::generated-
::SessionIdMarshaller::looseMarshal (OpenWireFormat * format,
commands::DataStructure * command, decaf::io::DataOutputStream * ds
) [virtual]`

Tight Marshal to the given stream.

Parameters

<i>format</i>	- The OpenWireFormat properties
<i>command</i>	- the object to Marshal
<i>ds</i>	- DataOutputStream to marshal to

Exceptions

<i>IOException</i>	if an error occurs.
--------------------	---------------------

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller**
(p. 1123).

6.472.3.4 `virtual void activemq::wireformat::openwire::marshal::generated-
::SessionIdMarshaller::looseUnmarshal (OpenWireFormat * format,
commands::DataStructure * command, decaf::io::DataInputStream * dis)
[virtual]`

Loose Un-marshal to the given stream.

Parameters

<i>format</i>	- The OpenWireFormat properties
<i>command</i>	- the object to Un-Marshal
<i>dis</i>	- the DataInputStream to Un-Marshal from

6.472 activemq::wireformat::openwire::marshal::generated::SessionIdMarshaller

Class Reference

2391

Exceptions

<i>IOException</i>	if an error occurs.
--------------------	---------------------

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1125).

6.472.3.5 virtual int **activemq::wireformat::openwire::marshal::generated::SessionIdMarshaller::tightMarshal1** (**OpenWireFormat** * *format*, **commands::DataStructure** * *command*, **utils::BooleanStream** * *bs*)
[virtual]

Tight Marshal to the given stream.

Parameters

<i>format</i>	The OpenWireFormat properties
<i>command</i>	The object to Marshal
<i>bs</i>	The boolean stream to marshal to.

Exceptions

<i>IOException</i>	if an error occurs.
--------------------	---------------------

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1127).

6.472.3.6 virtual void **activemq::wireformat::openwire::marshal::generated::SessionIdMarshaller::tightMarshal2** (**OpenWireFormat** * *format*, **commands::DataStructure** * *command*, **decaf::io::DataOutputStream** * *ds*, **utils::BooleanStream** * *bs*) [virtual]

Tight Marshal to the given stream.

Parameters

<i>format</i>	- The OpenWireFormat properties
<i>command</i>	- the object to Marshal
<i>ds</i>	- the DataOutputStream to Marshal to
<i>bs</i>	- boolean stream to marshal to.

Exceptions

<i>IOException</i>	if an error occurs.
--------------------	---------------------

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1129).

6.472.3.7 virtual void activemq::wireformat::openwire::marshal::generated-
::SessionIdMarshaller::tightUnmarshal (OpenWireFormat * *format*,
commands::DataStructure * *command*, decaf::io::DataInputStream * *dis*,
utils::BooleanStream * *bs*) [virtual]

Tight Un-marhsal to the given stream.

Parameters

<i>format</i>	- The OpenWireFormat properties
<i>command</i>	- the object to Un-Marshal
<i>dis</i>	- the DataInputStream to Un-Marshal from
<i>bs</i>	- boolean stream to unmarshal from.

Exceptions

<i>IOException</i>	if an error occurs.
--------------------	---------------------

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller**
(p. 1131).

The documentation for this class was generated from the following file:

- src/main/activemq/wireformat/openwire/marshal/generated/**SessionIdMarshaller.h**

6.473 activemq::commands::SessionInfo Class Reference

```
#include <src/main/activemq/commands/SessionInfo.h>
```

Inheritance diagram for activemq::commands::SessionInfo:

Public Member Functions

- **SessionInfo** ()
- virtual **~SessionInfo** ()
- virtual unsigned char **getDataStructureType** () const
*Get the **DataStructure** (p. 1133) Type as defined in CommandTypes.h.*
- virtual **SessionInfo** * **cloneDataStructure** () const
Clone this obbect and return a new instance that the caller now owns, this will be an exact copy of this one.
- virtual void **copyDataStructure** (const **DataStructure** *src)
Copy the contents of the passed object into this objects members, overwriting any existing data.
- virtual std::string **toString** () const

Returns a string containing the information for this **DataSet** (p. 1133) such as its type and value of its elements.

- virtual bool **equals** (const **DataSet** *value) const
Compares the **DataSet** (p. 1133) passed in to this one, and returns if they are equivalent.
- unsigned int **getAckMode** () const
- void **setAckMode** (unsigned int mode)
- **Pointer**< **RemoveInfo** > **createRemoveCommand** () const
- virtual const **Pointer** < **SessionId** > & **getSessionId** () const
- virtual **Pointer**< **SessionId** > & **getSessionId** ()
- virtual void **setSessionId** (const **Pointer**< **SessionId** > &sessionId)
- virtual **Pointer**< **Command** > **visit** (activemq::state::CommandVisitor *visitor)
Allows a Visitor to visit this command and return a response to the command based on the command type being visited.

Static Public Attributes

- static const unsigned char **ID_SESSIONINFO** = 4

Protected Attributes

- **Pointer**< **SessionId** > **sessionId**

6.473.1 Constructor & Destructor Documentation

6.473.1.1 **activemq::commands::SessionInfo::SessionInfo** ()

6.473.1.2 virtual **activemq::commands::SessionInfo::~~SessionInfo** ()
[virtual]

6.473.2 Member Function Documentation

6.473.2.1 virtual **SessionInfo*** **activemq::commands::SessionInfo::cloneDataSet** () const [virtual]

Clone this object and return a new instance that the caller now owns, this will be an exact copy of this one.

Returns

new copy of this object.

Implements **activemq::commands::DataSet** (p. 1133).

6.473.2.2 `virtual void activemq::commands::SessionInfo::copyDataStructure (const DataStructure * src) [virtual]`

Copy the contents of the passed object into this objects members, overwriting any existing data.

Returns

src - Source Object

Reimplemented from `activemq::commands::BaseCommand` (p. 493).

6.473.2.3 `Pointer<RemoveInfo> activemq::commands::SessionInfo::createRemoveCommand () const`

6.473.2.4 `virtual bool activemq::commands::SessionInfo::equals (const DataStructure * value) const [virtual]`

Compares the **DataStructure** (p. 1133) passed in to this one, and returns if they are equivalent.

Equivalent here means that they are of the same type, and that each element of the objects are the same.

Returns

true if **DataStructure** (p. 1133)'s are Equal.

Reimplemented from `activemq::commands::BaseCommand` (p. 494).

6.473.2.5 `unsigned int activemq::commands::SessionInfo::getAckMode () const [inline]`

6.473.2.6 `virtual unsigned char activemq::commands::SessionInfo::getDataStructureType () const [virtual]`

Get the **DataStructure** (p. 1133) Type as defined in CommandTypes.h.

Returns

The type of the data structure

Implements `activemq::commands::DataStructure` (p. 1137).

6.473.2.7 `virtual const Pointer<SessionId>& activemq::commands::SessionInfo::getSessionId () const [virtual]`

Referenced by `activemq::state::ConnectionState::addSession()`.

6.473.2.8 virtual `Pointer<SessionId>& activemq::commands::SessionInfo::getSessionId ()` [virtual]

6.473.2.9 void `activemq::commands::SessionInfo::setAckMode (unsigned int mode)` [inline]

6.473.2.10 virtual void `activemq::commands::SessionInfo::setSessionId (const Pointer< SessionId > & sessionId)` [virtual]

6.473.2.11 virtual `std::string activemq::commands::SessionInfo::toString ()` const [virtual]

Returns a string containing the information for this **DataSet** (p. 1133) such as its type and value of its elements.

Returns

formatted string useful for debugging.

Reimplemented from `activemq::commands::BaseCommand` (p. 498).

6.473.2.12 virtual `Pointer<Command> activemq::commands::SessionInfo::visit (activemq::state::CommandVisitor * visitor)` [virtual]

Allows a Visitor to visit this command and return a response to the command based on the command type being visited.

The command will call the proper processXXX method in the visitor.

Returns

a **Response** (p. 2298) to the visitor being called or NULL if no response.

Implements `activemq::commands::Command` (p. 871).

6.473.3 Field Documentation

6.473.3.1 const unsigned char `activemq::commands::SessionInfo::ID_SESSIONINFO = 4` [static]

6.473.3.2 `Pointer<SessionId> activemq::commands::SessionInfo::sessionId` [protected]

The documentation for this class was generated from the following file:

- `src/main/activemq/commands/SessionInfo.h`

6.474 activemq::wireformat::openwire::marshal::generated::SessionInfoMarshaller Class Reference

Marshaling code for Open Wire Format for **SessionInfoMarshaller** (p. 2390).

```
#include <src/main/activemq/wireformat/openwire/marshal/generated/SessionInfoMarshaller.h>
```

Inheritance diagram for activemq::wireformat::openwire::marshal::generated::SessionInfoMarshaller:

Public Member Functions

- **SessionInfoMarshaller** ()
- virtual **~SessionInfoMarshaller** ()
- virtual **commands::DataStructure * createObject** () const
Creates a new instance of the class that this class is a marshaling director for.
- virtual unsigned char **getDataStructureType** () const
Gets the DataStructureType that this class marshals/unmarshals.
- virtual void **tightUnmarshal** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataInputStream** *dataIn, **utils::BooleanStream** *bs)
Tight Un-marhsal to the given stream.
- virtual int **tightMarshal1** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **utils::BooleanStream** *bs)
Tight Marshal to the given stream.
- virtual void **tightMarshal2** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataOutputStream** *dataOut, **utils::BooleanStream** *bs)
Tight Marshal to the given stream.
- virtual void **looseUnmarshal** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataInputStream** *dataIn)
Loose Un-marhsal to the given stream.
- virtual void **looseMarshal** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataOutputStream** *dataOut)
Tight Marshal to the given stream.

6.474.1 Detailed Description

Marshaling code for Open Wire Format for **SessionInfoMarshaller** (p. 2390).

NOTE!: This file is auto generated - do not modify! if you need to make a change, please see the Java Classes in the activemq-openwire-generator module

6.474

activemq::wireformat::openwire::marshal::generated::SessionInfoMarshaller

Class Reference

2397

6.474.2 Constructor & Destructor Documentation

6.474.2.1 **activemq::wireformat::openwire::marshal::generated-
::SessionInfoMarshaller::SessionInfoMarshaller ()**
[inline]

6.474.2.2 **virtual activemq::wireformat::openwire::marshal::generated:-
SessionInfoMarshaller::~~SessionInfoMarshaller ()** [inline,
virtual]

6.474.3 Member Function Documentation

6.474.3.1 **virtual commands::DataStructure* activemq::wireformat::openwire-
::marshal::generated::SessionInfoMarshaller::createObject ()** const
[virtual]

Creates a new instance of the class that this class is a marshaling director for.

Returns

newly allocated Command

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller**
(p. 1120).

6.474.3.2 **virtual unsigned char activemq::wireformat::openwire::marshal-
::generated::SessionInfoMarshaller::getDataStructureType ()** const
[virtual]

Gets the DataStructureType that this class marshals/unmarshals.

Returns

byte Id of this classes DataStructureType

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller**
(p. 1122).

6.474.3.3 **virtual void activemq::wireformat::openwire::marshal::generated-
::SessionInfoMarshaller::looseMarshal (OpenWireFormat * *format*,
commands::DataStructure * *command*, decaf::io::DataOutputStream * *ds*
)** [virtual]

Tight Marhsal to the given stream.

Parameters

<i>format</i>	- The OpenWireFormat properties
<i>command</i>	- the object to Marshal
<i>ds</i>	- DataOutputStream to marshal to

Generated on Tue Feb 28 2012 17:47:25 for activemq-cpp-3.4.1 by Doxygen

Exceptions

<i>IOException</i>	if an error occurs.
--------------------	---------------------

Reimplemented from **activemq::wireformat::openwire::marshal::generated::Base-CommandMarshaller** (p. 500).

6.474.3.4 `virtual void activemq::wireformat::openwire::marshal::generated::SessionInfoMarshaller::looseUnmarshal (OpenWireFormat * format, commands::DataStructure * command, decaf::io::DataInputStream * dis)`
[virtual]

Loose Un-marhsal to the given stream.

Parameters

<i>format</i>	- The OpenWireFormat properties
<i>command</i>	- the object to Un-Marshal
<i>dis</i>	- the DataInputStream to Un-Marshal from

Exceptions

<i>IOException</i>	if an error occurs.
--------------------	---------------------

Reimplemented from **activemq::wireformat::openwire::marshal::generated::Base-CommandMarshaller** (p. 501).

6.474.3.5 `virtual int activemq::wireformat::openwire::marshal::generated::SessionInfoMarshaller::tightMarshal1 (OpenWireFormat * format, commands::DataStructure * command, utils::BooleanStream * bs)`
[virtual]

Tight Marhsal to the given stream.

Parameters

<i>format</i>	The OpenWireFormat properties
<i>command</i>	The object to Marshal
<i>bs</i>	The boolean stream to marshal to.

Exceptions

<i>IOException</i>	if an error occurs.
--------------------	---------------------

Reimplemented from **activemq::wireformat::openwire::marshal::generated::Base-CommandMarshaller** (p. 503).

6.474

activemq::wireformat::openwire::marshal::generated::SessionInfoMarshaller

Class Reference

2399

6.474.3.6 virtual void **activemq::wireformat::openwire::marshal::generated::SessionInfoMarshaller::tightMarshal2** (**OpenWireFormat** * *format*, **commands::DataStructure** * *command*, **decaf::io::DataOutputStream** * *ds*, **utils::BooleanStream** * *bs*) [virtual]

Tight Marhsal to the given stream.

Parameters

<i>format</i>	- The OpenWireFormat properties
<i>command</i>	- the object to Marshal
<i>ds</i>	- the DataOutputStream to Marshal to
<i>bs</i>	- boolean stream to marshal to.

Exceptions

<i>IOException</i>	if an error occurs.
--------------------	---------------------

Reimplemented from **activemq::wireformat::openwire::marshal::generated::Base-CommandMarshaller** (p. 504).

6.474.3.7 virtual void **activemq::wireformat::openwire::marshal::generated::SessionInfoMarshaller::tightUnmarshal** (**OpenWireFormat** * *format*, **commands::DataStructure** * *command*, **decaf::io::DataInputStream** * *dis*, **utils::BooleanStream** * *bs*) [virtual]

Tight Un-marhsal to the given stream.

Parameters

<i>format</i>	- The OpenWireFormat properties
<i>command</i>	- the object to Un-Marshal
<i>dis</i>	- the DataInputStream to Un-Marshal from
<i>bs</i>	- boolean stream to unmarshal from.

Exceptions

<i>IOException</i>	if an error occurs.
--------------------	---------------------

Reimplemented from **activemq::wireformat::openwire::marshal::generated::Base-CommandMarshaller** (p. 505).

The documentation for this class was generated from the following file:

- src/main/activemq/wireformat/openwire/marshal/generated/**SessionInfoMarshaller.h**

6.475 activemq::cmsutil::SessionPool Class Reference

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

```
#include <src/main/activemq/cmsutil/SessionPool.h>
```

Public Member Functions

- **SessionPool** (**cms::Connection** *connection, **cms::Session::AcknowledgeMode** ackMode, **ResourceLifecycleManager** *resourceLifecycleManager)
Constructs a session pool.
- virtual **~SessionPool** ()
Destroys the pooled session objects, but not the underlying session resources.
- virtual **PooledSession** * **takeSession** ()
Takes a session from the pool, creating one if necessary.
- virtual void **returnSession** (**PooledSession** *session)
Returns a session to the pool.
- **ResourceLifecycleManager** * **getResourceLifecycleManager** ()

6.475.1 Detailed Description

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

Internal session resources are managed through a provided **ResourceLifecycleManager** (p. 2293), not by this pool. This class is thread-safe.

6.475.2 Constructor & Destructor Documentation

- 6.475.2.1 **activemq::cmsutil::SessionPool::SessionPool** (**cms::Connection** * connection, **cms::Session::AcknowledgeMode** ackMode, **ResourceLifecycleManager** * resourceLifecycleManager)

Constructs a session pool.

Parameters

<i>connection</i>	the connection to be used for creating all sessions.
<i>ackMode</i>	the acknowledge mode to be used for all sessions
<i>resource-Lifecycle-Manager</i>	the object responsible for managing the lifecycle of any allocated cms::Session (p. 2361) resources.

6.475.2.2 virtual **activemq::cmsutil::SessionPool::~~SessionPool**() [virtual]

Destroys the pooled session objects, but not the underlying session resources.

That is the job of the **ResourceLifecycleManager** (p. 2293).

6.475.3 Member Function Documentation

6.475.3.1 **ResourceLifecycleManager*** **activemq::cmsutil::SessionPool::getResourceLifecycleManager**()
[inline]

6.475.3.2 virtual void **activemq::cmsutil::SessionPool::returnSession**(
PooledSession * *session*) [virtual]

Returns a session to the pool.

Parameters

<i>session</i>	the session to be returned.
----------------	-----------------------------

6.475.3.3 virtual **PooledSession*** **activemq::cmsutil::SessionPool::takeSession**()
[virtual]

Takes a session from the pool, creating one if necessary.

Returns

the pooled session object

Exceptions

<i>CMSEException</i>	if an error occurred
----------------------	----------------------

The documentation for this class was generated from the following file:

- src/main/activemq/cmsutil/**SessionPool.h**

6.476 activemq::state::SessionState Class Reference

```
#include <src/main/activemq/state/SessionState.h>
```

Public Member Functions

- **SessionState**(const **Pointer**< **SessionInfo** > &info)

- virtual **~SessionState** ()
- std::string **toString** () const
- const **Pointer**< **SessionInfo** > **getInfo** () const
- void **addProducer** (const **Pointer**< **ProducerInfo** > &info)
- **Pointer**< **ProducerState** > **removeProducer** (const **Pointer**< **ProducerId** > &id)
- void **addConsumer** (const **Pointer**< **ConsumerInfo** > &info)
- **Pointer**< **ConsumerState** > **removeConsumer** (const **Pointer**< **ConsumerId** > &id)
- std::vector< **Pointer** < **ProducerState** > > **getProducerStates** () const
- **Pointer**< **ProducerState** > **getProducerState** (const **Pointer**< **ProducerId** > &id)
- std::vector< **Pointer** < **ConsumerState** > > **getConsumerStates** () const
- **Pointer**< **ConsumerState** > **getConsumerState** (const **Pointer**< **ConsumerId** > &id)
- void **checkShutdown** () const
- void **shutdown** ()

6.476.1 Constructor & Destructor Documentation

6.476.1.1 **activemq::state::SessionState::SessionState** (const **Pointer**< **SessionInfo** > & *info*)

6.476.1.2 virtual **activemq::state::SessionState::~~SessionState** () [virtual]

6.476.2 Member Function Documentation

6.476.2.1 void **activemq::state::SessionState::addConsumer** (const **Pointer**< **ConsumerInfo** > & *info*) [inline]

References **activemq::commands::ConsumerInfo::getConsumerId()**.

6.476.2.2 void **activemq::state::SessionState::addProducer** (const **Pointer**< **ProducerInfo** > & *info*)

6.476.2.3 void **activemq::state::SessionState::checkShutdown** () const

6.476.2.4 **Pointer**<**ConsumerState**> **activemq::state::SessionState::getConsumerState** (const **Pointer**< **ConsumerId** > & *id*) [inline]

6.476.2.5 std::vector< **Pointer**<**ConsumerState**> > **activemq::state::SessionState::getConsumerStates** () const [inline]

6.476.2.6 `const Pointer<SessionInfo> activemq::state::SessionState::getInfo ()`
`const [inline]`

6.476.2.7 `Pointer<ProducerState> activemq::state::SessionState-
 ::getProducerState (const Pointer< ProducerId > & id)`
`[inline]`

6.476.2.8 `std::vector< Pointer<ProducerState> > activemq-
 ::state::SessionState::getProducerStates () const`
`[inline]`

6.476.2.9 `Pointer<ConsumerState> activemq::state::SessionState-
 ::removeConsumer (const Pointer< ConsumerId > & id)`
`[inline]`

6.476.2.10 `Pointer<ProducerState> activemq::state::SessionState-
 ::removeProducer (const Pointer< ProducerId > & id)`
`)`

6.476.2.11 `void activemq::state::SessionState::shutdown () [inline]`

6.476.2.12 `std::string activemq::state::SessionState::toString () const`

The documentation for this class was generated from the following file:

- `src/main/activemq/state/SessionState.h`

6.477 decaf::util::Set< E > Class Template Reference

A collection that contains no duplicate elements.

```
#include <src/main/decaf/util/Set.h>
```

Inheritance diagram for `decaf::util::Set< E >`:

Public Member Functions

- `virtual ~Set ()`

6.477.1 Detailed Description

```
template<typename E> class decaf::util::Set< E >
```

A collection that contains no duplicate elements.

More formally, sets contain no pair of elements e_1 and e_2 such that $e_1 == e_2$, and at most one null element. As implied by its name, this interface models the mathematical set abstraction.

The additional stipulation on constructors is, not surprisingly, that all constructors must create a set that contains no duplicate elements (as defined above).

Note: Great care must be exercised if mutable objects are used as set elements. The behavior of a set is not specified if the value of an object is changed in a manner that affects equals comparisons while the object is an element in the set.

Since

1.0

6.477.2 Constructor & Destructor Documentation

6.477.2.1 `template<typename E> virtual decaf::util::Set< E >::~~Set () [inline, virtual]`

The documentation for this class was generated from the following file:

- `src/main/decaf/util/Set.h`

6.478 decaf::lang::Short Class Reference

```
#include <src/main/decaf/lang/Short.h>
```

Inheritance diagram for `decaf::lang::Short`:

Public Member Functions

- **Short** (short value)
- **Short** (const std::string &value)
- virtual **~Short** ()
- virtual int **compareTo** (const **Short** &s) const
*Compares this **Short** (p. 2398) instance with another.*
- bool **equals** (const **Short** &s) const
- virtual bool **operator==** (const **Short** &s) const
Compares equality between this object and the one passed.
- virtual bool **operator<** (const **Short** &s) const
Compares this object to another and returns true if this object is considered to be less than the one passed.
- virtual int **compareTo** (const short &s) const
*Compares this **Short** (p. 2398) instance with another.*

- bool **equals** (const short &s) const
- virtual bool **operator==** (const short &s) const
Compares equality between this object and the one passed.
- virtual bool **operator<** (const short &s) const
Compares this object to another and returns true if this object is considered to be less than the one passed.
- std::string **toString** () const
- virtual double **doubleValue** () const
Answers the double value which the receiver represents.
- virtual float **floatValue** () const
Answers the float value which the receiver represents.
- virtual unsigned char **byteValue** () const
Answers the byte value which the receiver represents.
- virtual short **shortValue** () const
Answers the short value which the receiver represents.
- virtual int **intValue** () const
Answers the int value which the receiver represents.
- virtual long long **longValue** () const
Answers the long value which the receiver represents.

Static Public Member Functions

- static std::string **toString** (short value)
- static **Short decode** (const std::string &value)
*Decodes a **String** (p. 2620) into a **Short** (p. 2398).*
- static short **reverseBytes** (short value)
Returns the value obtained by reversing the order of the bytes in the two's complement representation of the specified short value.
- static short **parseShort** (const std::string &s, int radix)
Parses the string argument as a signed short in the radix specified by the second argument.
- static short **parseShort** (const std::string &s)
Parses the string argument as a signed decimal short.
- static **Short valueOf** (short value)
*Returns a **Short** (p. 2398) instance representing the specified short value.*
- static **Short valueOf** (const std::string &value)
*Returns a **Short** (p. 2398) object holding the value given by the specified std::string.*
- static **Short valueOf** (const std::string &value, int radix)
*Returns a **Short** (p. 2398) object holding the value extracted from the specified std::string when parsed with the radix given by the second argument.*

Static Public Attributes

- static const int **SIZE** = 16
Size of this objects primitive type in bits.
- static const short **MAX_VALUE** = (short)0x7FFF
Max Value for this Object's primitive type.
- static const short **MIN_VALUE** = (short)0x8000
Max Value for this Object's primitive type.

6.478.1 Constructor & Destructor Documentation

6.478.1.1 decaf::lang::Short::Short (short value)

Parameters

<i>value</i>	- short to wrap
--------------	-----------------

6.478.1.2 decaf::lang::Short::Short (const std::string & value)

Parameters

<i>value</i>	The string value to convert to short and wrap.
--------------	--

Exceptions

<i>NumberFormatException</i>	if the string is not well formed number value.
------------------------------	--

6.478.1.3 virtual decaf::lang::Short::~~Short () [inline, virtual]

6.478.2 Member Function Documentation

6.478.2.1 virtual unsigned char decaf::lang::Short::byteValue () const [inline, virtual]

Answers the byte value which the receiver represents.

Returns

int the value of the receiver.

Reimplemented from **decaf::lang::Number** (p. 1993).

6.478.2.2 `virtual int decaf::lang::Short::compareTo (const Short & s) const`
[virtual]

Compares this **Short** (p. 2398) instance with another.

Parameters

<code>s</code>	- the Short (p. 2398) instance to be compared
----------------	--

Returns

zero if this object represents the same short value as the argument; a positive value if this object represents a value greater than the passed in value, and -1 if this object represents a value less than the passed in value.

Implements **decaf::lang::Comparable**< **Short** > (p. 886).

6.478.2.3 `virtual int decaf::lang::Short::compareTo (const short & s) const`
[virtual]

Compares this **Short** (p. 2398) instance with another.

Parameters

<code>s</code>	- the Short (p. 2398) instance to be compared
----------------	--

Returns

zero if this object represents the same short value as the argument; a positive value if this object represents a value greater than the passed in value, and -1 if this object represents a value less than the passed in value.

Implements **decaf::lang::Comparable**< **short** > (p. 886).

6.478.2.4 `static Short decaf::lang::Short::decode (const std::string & value)`
[static]

Decodes a **String** (p. 2620) into a **Short** (p. 2398).

Accepts decimal, hexadecimal, and octal numbers given by the following grammar:

The sequence of characters following an (optional) negative sign and/or radix specifier ("0x", "0X", "#", or leading zero) is parsed as by the **Short.parseShort** (p. 2405) method with the indicated radix (10, 16, or 8). This sequence of characters must represent a positive value or a **NumberFormatException** will be thrown. The result is negated if first character of the specified **String** (p. 2620) is the minus sign. No whitespace characters are permitted in the string.

Parameters

<i>value</i>	- The string to decode
--------------	------------------------

Returns

a **Short** (p. 2398) object containing the decoded value

Exceptions

<i>NumberFormatException</i>	if the string is not formatted correctly.
------------------------------	---

6.478.2.5 `virtual double decaf::lang::Short::doubleValue () const [inline, virtual]`

Answers the double value which the receiver represents.

Returns

double the value of the receiver.

Implements **decaf::lang::Number** (p. 1993).

6.478.2.6 `bool decaf::lang::Short::equals (const Short & s) const [inline, virtual]`

Returns

true if the two **Short** (p. 2398) Objects have the same value.

Implements **decaf::lang::Comparable< Short >** (p. 887).

6.478.2.7 `bool decaf::lang::Short::equals (const short & s) const [inline, virtual]`

Returns

true if the two **Short** (p. 2398) Objects have the same value.

Implements **decaf::lang::Comparable< short >** (p. 887).

6.478.2.8 `virtual float decaf::lang::Short::floatValue () const [inline, virtual]`

Answers the float value which the receiver represents.

Returns

float the value of the receiver.

Implements **decaf::lang::Number** (p. 1993).

6.478.2.9 `virtual int decaf::lang::Short::intValue () const [inline, virtual]`

Answers the int value which the receiver represents.

Returns

int the value of the receiver.

Implements **decaf::lang::Number** (p. 1993).

6.478.2.10 `virtual long long decaf::lang::Short::longValue () const [inline, virtual]`

Answers the long value which the receiver represents.

Returns

long the value of the receiver.

Implements **decaf::lang::Number** (p. 1994).

6.478.2.11 `virtual bool decaf::lang::Short::operator< (const Short & s) const [inline, virtual]`

Compares this object to another and returns true if this object is considered to be less than the one passed.

This

Parameters

s	- the value to be compared to this one.
----------	---

Returns

true if this object is equal to the one passed.

Implements **decaf::lang::Comparable**< **Short** > (p. 887).

6.478.2.12 `virtual bool decaf::lang::Short::operator< (const short & s) const` `[inline, virtual]`

Compares this object to another and returns true if this object is considered to be less than the one passed.

This

Parameters

<code>s</code>	- the value to be compared to this one.
----------------	---

Returns

true if this object is equal to the one passed.

Implements **decaf::lang::Comparable**< **short** > (p. 887).

6.478.2.13 `virtual bool decaf::lang::Short::operator== (const Short & s) const` `[inline, virtual]`

Compares equality between this object and the one passed.

Parameters

<code>s</code>	- the value to be compared to this one.
----------------	---

Returns

true if this object is equal to the one passed.

Implements **decaf::lang::Comparable**< **Short** > (p. 888).

6.478.2.14 `virtual bool decaf::lang::Short::operator== (const short & s) const` `[inline, virtual]`

Compares equality between this object and the one passed.

Parameters

<code>s</code>	- the value to be compared to this one.
----------------	---

Returns

true if this object is equal to the one passed.

Implements **decaf::lang::Comparable**< **short** > (p. 888).

6.478.2.15 static short decaf::lang::Short::parseShort (const std::string & s, int radix)
[static]

Parses the string argument as a signed short in the radix specified by the second argument.

The characters in the string must all be digits, of the specified radix (as determined by whether **Character.digit(char, int)** (p. 768) returns a nonnegative value) except that the first character may be an ASCII minus sign '-' to indicate a negative value. The resulting byte value is returned.

An exception of type **NumberFormatException** is thrown if any of the following situations occurs: * The first argument is null or is a string of length zero. * The radix is either smaller than **Character.MIN_RADIX** (p. 773) or larger than **Character.MAX_RADIX** (p. 772). * Any character of the string is not a digit of the specified radix, except that the first character may be a minus sign '-' provided that the string is longer than length 1. * The value represented by the string is not a value of type short.

Parameters

s	- the String (p. 2620) containing the short representation to be parsed
radix	- the radix to be used while parsing s

Returns

the short represented by the string argument in the specified radix.

Exceptions

<i>NumberFormatException</i>	- If String (p. 2620) does not contain a parsable short.
------------------------------	---

6.478.2.16 static short decaf::lang::Short::parseShort (const std::string & s)
[static]

Parses the string argument as a signed decimal short.

The characters in the string must all be decimal digits, except that the first character may be an ASCII minus sign '-' to indicate a negative value. The resulting short value is returned, exactly as if the argument and the radix 10 were given as arguments to the **parseShort(const std::string, int)** method.

Parameters

<i>s</i>	- String (p. 2620) to convert to a short
----------	---

Returns

the converted short value

Exceptions

<i>NumberFormatException</i>	if the string is not a short.
------------------------------	-------------------------------

6.478.2.17 `static short decaf::lang::Short::reverseBytes (short value)` `[static]`

Returns the value obtained by reversing the order of the bytes in the two's complement representation of the specified short value.

Parameters

<i>value</i>	- the short whose bytes we are to reverse
--------------	---

Returns

the reversed short.

6.478.2.18 `virtual short decaf::lang::Short::shortValue () const` `[inline, virtual]`

Answers the short value which the receiver represents.

Returns

int the value of the receiver.

Reimplemented from **decaf::lang::Number** (p. 1994).

6.478.2.19 `std::string decaf::lang::Short::toString () const`

Returns

this **Short** (p. 2398) Object as a **String** (p. 2620) Representation

6.478.2.20 `static std::string decaf::lang::Short::toString (short value)` `[static]`

Returns

a string representing the primitive value as Base 10

6.478.2.21 static **Short** decaf::lang::Short::valueOf (short *value*) [static]

Returns a **Short** (p. 2398) instance representing the specified short value.

Parameters

<i>value</i>	- the short to wrap
--------------	---------------------

Returns

the new **Short** (p. 2398) object wrapping value.

6.478.2.22 static **Short** decaf::lang::Short::valueOf (const std::string & *value*)
[static]

Returns a **Short** (p. 2398) object holding the value given by the specified std::string.

The argument is interpreted as representing a signed decimal short, exactly as if the argument were given to the parseShort(std::string) method. The result is a **Short** (p. 2398) object that represents the short value specified by the string.

Parameters

<i>value</i>	- std::string to parse as base 10
--------------	-----------------------------------

Returns

new **Short** (p. 2398) Object wrapping the primitive

Exceptions

<i>NumberFormatException</i>	if the string is not a decimal short.
------------------------------	---------------------------------------

6.478.2.23 static **Short** decaf::lang::Short::valueOf (const std::string & *value*, int *radix*)
[static]

Returns a **Short** (p. 2398) object holding the value extracted from the specified std::string when parsed with the radix given by the second argument.

The first argument is interpreted as representing a signed short in the radix specified by the second argument, exactly as if the argument were given to the parseShort(std-

::string, int) method. The result is a **Short** (p. 2398) object that represents the short value specified by the string.

Parameters

<i>value</i>	- std::string to parse as base (radix)
<i>radix</i>	- base of the string to parse.

Returns

new **Short** (p. 2398) Object wrapping the primitive

Exceptions

<i>NumberFormatException</i>	if the string is not a valid short.
------------------------------	-------------------------------------

6.478.3 Field Documentation

6.478.3.1 `const short decaf::lang::Short::MAX_VALUE = (short)0x7FFF` [static]

Max Value for this Object's primitive type.

6.478.3.2 `const short decaf::lang::Short::MIN_VALUE = (short)0x8000` [static]

Max Value for this Object's primitive type.

6.478.3.3 `const int decaf::lang::Short::SIZE = 16` [static]

Size of this objects primitive type in bits.

The documentation for this class was generated from the following file:

- src/main/decaf/lang/**Short.h**

6.479 decaf::internal::nio::ShortArrayBuffer Class Reference

```
#include <src/main/decaf/internal/nio/ShortArrayBuffer.h>
```

Inheritance diagram for decaf::internal::nio::ShortArrayBuffer:

Public Member Functions

- **ShortArrayBuffer** (int size, bool readOnly=false)
Creates a **ShortArrayBuffer** (p. 2408) object that has its backing array allocated internally and is then owned and deleted when this object is deleted.
- **ShortArrayBuffer** (short *array, int size, int offset, int length, bool readOnly=false)
Creates a **ShortArrayBuffer** (p. 2408) object that wraps the given array.
- **ShortArrayBuffer** (const decaf::lang::Pointer< **ByteArrayAdapter** > &array, int offset, int length, bool readOnly=false)
Creates a byte buffer that wraps the passed **ByteArrayAdapter** and start at the given offset.
- **ShortArrayBuffer** (const **ShortArrayBuffer** &other)
Create a **ShortArrayBuffer** (p. 2408) that mirrors this one, meaning it shares a reference to this buffers **ByteArrayAdapter** and when changes are made to that data it is reflected in both.
- virtual ~**ShortArrayBuffer** ()
- virtual short * array ()
Returns the short array that backs this buffer (optional operation).
Modifications to this buffer's content will cause the returned array's content to be modified, and vice versa.
Invoke the `hasArray` method before invoking this method in order to ensure that this buffer has an accessible backing array.
Returns
the array that backs this **Buffer** (p. 582)

Exceptions

ReadOnlyBuffer-Exception (p. 2244)	if this Buffer (p. 582) is read only.
UnsupportedOperation-Exception	if the underlying store has no array.

- virtual int arrayOffset ()
Returns the offset within this buffer's backing array of the first element of the buffer (optional operation).
Invoke the `hasArray` method before invoking this method in order to ensure that this buffer has an accessible backing array.
Returns
The offset into the backing array where index zero starts.

Exceptions

ReadOnlyBuffer-Exception (p. 2244)	if this Buffer (p. 582) is read only.
UnsupportedOperation-Exception	if the underlying store has no array.

- virtual **ShortBuffer** * asReadOnlyBuffer () const
Creates a new, read-only short buffer that shares this buffer's content.

The content of the new buffer will be that of this buffer. Changes to this buffer's content will be visible in the new buffer; the new buffer itself, however, will be read-only and will not allow the shared content to be modified. The two buffers' position, limit, and mark values will be independent.

If this buffer is itself read-only then this method behaves in exactly the same way as the duplicate method.

The new buffer's capacity, limit, position, and mark values will be identical to those of this buffer.

Returns

The new, read-only short buffer which the caller then owns.

- virtual **ShortBuffer & compact ()**

Compacts this buffer.

The bytes between the buffer's current position and its limit, if any, are copied to the beginning of the buffer. That is, the byte at index $p = \text{position}()$ (p. 587) is copied to index zero, the byte at index $p + 1$ is copied to index one, and so forth until the byte at index $\text{limit}()$ (p. 586) - 1 is copied to index $n = \text{limit}()$ (p. 586) - 1 - p . The buffer's position is then set to $n+1$ and its limit is set to its capacity. The mark, if defined, is discarded.

The buffer's position is set to the number of bytes copied, rather than to zero, so that an invocation of this method can be followed immediately by an invocation of another relative put method.

Returns

*a reference to this **ShortBuffer** (p. 2419).*

Exceptions

ReadOnlyBuffer-Exception (p. 2244)	<i>if this buffer is read-only.</i>
--	-------------------------------------

- virtual **ShortBuffer * duplicate ()**

Creates a new short buffer that shares this buffer's content.

The content of the new buffer will be that of this buffer. Changes to this buffer's content will be visible in the new buffer, and vice versa; the two buffers' position, limit, and mark values will be independent.

The new buffer's capacity, limit, position, and mark values will be identical to those of this buffer. The new buffer will be read-only if, and only if, this buffer is read-only.

Returns

*a new short **Buffer** (p. 582) which the caller owns.*

- virtual short **get ()**

Relative get method.

Reads the value at this buffer's current position, and then increments the position.

Returns

the short at the current position.

Exceptions

BufferUnderflow-Exception (p. 611)	<i>if there no more data to return.</i>
--	---

- virtual short **get (int index) const**

Absolute get method.

Reads the value at the given index.

Parameters

index	<i>The index in the Buffer (p. 582) where the short is to be read.</i>
-------	---

Returns

the short that is located at the given index.

Exceptions

IndexOutOfBoundsException	<i>if index is not smaller than the buffer's limit, or the index is negative.</i>
---------------------------	---

- virtual bool **hasArray** () const

Tells whether or not this buffer is backed by an accessible short array.

If this method returns true then the array and arrayOffset methods may safely be invoked. Subclasses should override this method if they do not have a backing array as this class always returns true.

Returns

true if, and only if, this buffer is backed by an array and is not read-only.

- virtual bool **isReadOnly** () const

Tells whether or not this buffer is read-only.

Returns

true if, and only if, this buffer is read-only.

- virtual **ShortBuffer** & **put** (short value)

Writes the given shorts into this buffer at the current position, and then increments the position.

Parameters

value	<i>The shorts value to be written.</i>
-------	--

Returns

a reference to this buffer.

Exceptions

BufferOverflow-Exception (p. 609)	<i>if this buffer's current position is not smaller than its limit.</i>
ReadOnlyBuffer-Exception (p. 2244)	<i>if this buffer is read-only.</i>

- virtual **ShortBuffer** & **put** (int index, short value)

Writes the given shorts into this buffer at the given index.

Parameters

index	<i>The position in the Buffer (p. 582) to write the data.</i>
value	<i>The shorts to write.</i>

Returns

a reference to this buffer.

Exceptions

IndexOutOfBounds-Exception	if index greater than the buffer's limit minus the size of the type being written.
ReadOnlyBuffer-Exception (p. 2244)	if this buffer is read-only.

- virtual **ShortBuffer * slice ()** const

Creates a new **ShortBuffer** (p. 2419) whose content is a shared subsequence of this buffer's content.

The content of the new buffer will start at this buffer's current position. Changes to this buffer's content will be visible in the new buffer, and vice versa; the two buffers' position, limit, and mark values will be independent.

The new buffer's position will be zero, its capacity and its limit will be the number of bytes remaining in this buffer, and its mark will be undefined. The new buffer will be read-only if, and only if, this buffer is read-only.

Returns

the newly create **ShortBuffer** (p. 2419) which the caller owns.

Protected Member Functions

- virtual void **setReadOnly** (bool value)

Sets this **ShortArrayBuffer** (p. 2408) as Read-Only.

6.479.1 Constructor & Destructor Documentation

6.479.1.1 **decaf::internal::nio::ShortArrayBuffer::ShortArrayBuffer** (int size, bool readOnly = false)

Creates a **ShortArrayBuffer** (p. 2408) object that has its backing array allocated internally and is then owned and deleted when this object is deleted.

The array is initially created with all elements initialized to zero.

Parameters

size	The size of the array, this is the limit we read and write to.
readOnly	Boolean indicating if this buffer should be read-only, default as false.

Exceptions

<i>IllegalArgument-Exception</i>	if the capacity value is negative.
----------------------------------	------------------------------------

6.479.1.2 decaf::internal::nio::ShortArrayBuffer::ShortArrayBuffer (short * array, int size, int offset, int length, bool readOnly = false)

Creates a **ShortArrayBuffer** (p. 2408) object that wraps the given array.

If the own flag is set then it will delete this array when this object is deleted.

Parameters

<i>array</i>	The actual array to wrap.
<i>size</i>	The size of the given array.
<i>offset</i>	The position that is this buffers start position.
<i>length</i>	The limit of how many bytes into the array this Buffer can write.
<i>readOnly</i>	Boolean indicating if this buffer should be read-only, default as false.

Exceptions

<i>NullPointerException</i>	if buffer is NULL
<i>IndexOutOfBoundsException</i>	if offset is greater than array capacity.

6.479.1.3 decaf::internal::nio::ShortArrayBuffer::ShortArrayBuffer (const decaf::lang::Pointer< ByteArrayAdapter > & array, int offset, int length, bool readOnly = false)

Creates a byte buffer that wraps the passed ByteArrayAdapter and start at the given offset.

The capacity and limit of the new **ShortArrayBuffer** (p. 2408) will be that of the remaining capacity of the passed buffer.

Parameters

<i>array</i>	The ByteArrayAdapter to wrap.
<i>offset</i>	The position that is this buffers start position.
<i>length</i>	The limit of how many bytes into the array this Buffer can write.
<i>readOnly</i>	Boolean indicating if this buffer should be read-only, default as false.

Exceptions

<i>NullPointerException</i>	if array is NULL
<i>IndexOutOfBoundsException</i>	if offset + length is greater than array size.

6.479.1.4 `decaf::internal::nio::ShortArrayBuffer::ShortArrayBuffer (const ShortArrayBuffer & other)`

Create a **ShortArrayBuffer** (p. 2408) that mirrors this one, meaning it shares a reference to this buffers `ByteArrayAdapter` and when changes are made to that data it is reflected in both.

Parameters

<i>other</i>	The ShortArrayBuffer (p. 2408) this one is to mirror.
--------------	--

6.479.1.5 `virtual decaf::internal::nio::ShortArrayBuffer::~~ShortArrayBuffer ()` [virtual]

6.479.2 Member Function Documentation

6.479.2.1 `virtual short* decaf::internal::nio::ShortArrayBuffer::array ()` [virtual]

Returns the short array that backs this buffer (optional operation).

Modifications to this buffer's content will cause the returned array's content to be modified, and vice versa.

Invoke the `hasArray` method before invoking this method in order to ensure that this buffer has an accessible backing array.

Returns

the array that backs this **Buffer** (p. 582)

Exceptions

<i>ReadOnlyBuffer-Exception</i> (p. 2244)	if this Buffer (p. 582) is read only.
<i>Unsupported-OperationException</i>	if the underlying store has no array.

Implements **decaf::nio::ShortBuffer** (p. 2422).

6.479.2.2 `virtual int decaf::internal::nio::ShortArrayBuffer::arrayOffset ()` [virtual]

Returns the offset within this buffer's backing array of the first element of the buffer (optional operation).

Invoke the `hasArray` method before invoking this method in order to ensure that this buffer has an accessible backing array.

Returns

The offset into the backing array where index zero starts.

Exceptions

ReadOnlyBufferException (p. 2244)	if this Buffer (p. 582) is read only.
<i>UnsupportedOperationException</i>	if the underlying store has no array.

Implements **decaf::nio::ShortBuffer** (p. 2422).

6.479.2.3 virtual **ShortBuffer*** **decaf::internal::nio::ShortArrayBuffer::asReadOnlyBuffer ()** const [virtual]

Creates a new, read-only short buffer that shares this buffer's content.

The content of the new buffer will be that of this buffer. Changes to this buffer's content will be visible in the new buffer; the new buffer itself, however, will be read-only and will not allow the shared content to be modified. The two buffers' position, limit, and mark values will be independent.

If this buffer is itself read-only then this method behaves in exactly the same way as the duplicate method.

The new buffer's capacity, limit, position, and mark values will be identical to those of this buffer.

Returns

The new, read-only short buffer which the caller then owns.

Implements **decaf::nio::ShortBuffer** (p. 2423).

6.479.2.4 virtual **ShortBuffer&** **decaf::internal::nio::ShortArrayBuffer::compact ()** [virtual]

Compacts this buffer.

The bytes between the buffer's current position and its limit, if any, are copied to the beginning of the buffer. That is, the byte at index $p = \text{position}()$ (p. 587) is copied to index zero, the byte at index $p + 1$ is copied to index one, and so forth until the byte at index $\text{limit}()$ (p. 586) - 1 is copied to index $n = \text{limit}()$ (p. 586) - 1 - p . The buffer's position is then set to $n+1$ and its limit is set to its capacity. The mark, if defined, is discarded.

The buffer's position is set to the number of bytes copied, rather than to zero, so that an invocation of this method can be followed immediately by an invocation of another relative put method.

Returns

a reference to this **ShortBuffer** (p. 2419).

Exceptions

<i>ReadOnlyBuffer-Exception</i> (p. 2244)	if this buffer is read-only.
---	------------------------------

Implements **decaf::nio::ShortBuffer** (p. 2423).

6.479.2.5 **virtual ShortBuffer* decaf::internal::nio::ShortArrayBuffer::duplicate ()**
[virtual]

Creates a new short buffer that shares this buffer's content.

The content of the new buffer will be that of this buffer. Changes to this buffer's content will be visible in the new buffer, and vice versa; the two buffers' position, limit, and mark values will be independent.

The new buffer's capacity, limit, position, and mark values will be identical to those of this buffer. The new buffer will be read-only if, and only if, this buffer is read-only.

Returns

a new short **Buffer** (p. 582) which the caller owns.

Implements **decaf::nio::ShortBuffer** (p. 2424).

6.479.2.6 **virtual short decaf::internal::nio::ShortArrayBuffer::get ()** [virtual]

Relative get method.

Reads the value at this buffer's current position, and then increments the position.

Returns

the short at the current position.

Exceptions

<i>BufferUnderflow-Exception</i> (p. 611)	if there no more data to return.
---	----------------------------------

Implements **decaf::nio::ShortBuffer** (p. 2424).

6.479.2.7 `virtual short decaf::internal::nio::ShortArrayBuffer::get (int index) const`
[virtual]

Absolute get method.

Reads the value at the given index.

Parameters

<i>index</i>	The index in the Buffer (p. 582) where the short is to be read.
--------------	--

Returns

the short that is located at the given index.

Exceptions

<i>IndexOutOfBoundsException</i>	if index is not smaller than the buffer's limit, or the index is negative.
----------------------------------	--

Implements **decaf::nio::ShortBuffer** (p. 2425).

6.479.2.8 `virtual bool decaf::internal::nio::ShortArrayBuffer::hasArray () const`
[inline, virtual]

Tells whether or not this buffer is backed by an accessible short array.

If this method returns true then the array and arrayOffset methods may safely be invoked. Subclasses should override this method if they do not have a backing array as this class always returns true.

Returns

true if, and only if, this buffer is backed by an array and is not read-only.

Implements **decaf::nio::ShortBuffer** (p. 2426).

6.479.2.9 `virtual bool decaf::internal::nio::ShortArrayBuffer::isReadOnly () const`
[inline, virtual]

Tells whether or not this buffer is read-only.

Returns

true if, and only if, this buffer is read-only.

Implements **decaf::nio::Buffer** (p. 586).

6.479.2.10 `virtual ShortBuffer& decaf::internal::nio::ShortArrayBuffer::put (short value) [virtual]`

Writes the given shorts into this buffer at the current position, and then increments the position.

Parameters

<i>value</i>	The shorts value to be written.
--------------	---------------------------------

Returns

a reference to this buffer.

Exceptions

<i>BufferOverflow-Exception</i> (p. 609)	if this buffer's current position is not smaller than its limit.
<i>ReadOnlyBuffer-Exception</i> (p. 2244)	if this buffer is read-only.

Implements **decaf::nio::ShortBuffer** (p. 2429).

6.479.2.11 `virtual ShortBuffer& decaf::internal::nio::ShortArrayBuffer::put (int index, short value) [virtual]`

Writes the given shorts into this buffer at the given index.

Parameters

<i>index</i>	The position in the Buffer (p. 582) to write the data.
<i>value</i>	The shorts to write.

Returns

a reference to this buffer.

Exceptions

<i>IndexOutOfBoundsException</i>	if index greater than the buffer's limit minus the size of the type being written.
<i>ReadOnlyBuffer-Exception</i> (p. 2244)	if this buffer is read-only.

Implements **decaf::nio::ShortBuffer** (p. 2429).

6.479.2.12 virtual void **decaf::internal::nio::ShortArrayBuffer::setReadOnly** (bool *value*) [inline, protected, virtual]

Sets this **ShortArrayBuffer** (p. 2408) as Read-Only.

Parameters

<i>value</i>	Boolean value, true if this buffer is to be read-only, false otherwise.
--------------	---

6.479.2.13 virtual **ShortBuffer*** **decaf::internal::nio::ShortArrayBuffer::slice** ()
const [virtual]

Creates a new **ShortBuffer** (p. 2419) whose content is a shared subsequence of this buffer's content.

The content of the new buffer will start at this buffer's current position. Changes to this buffer's content will be visible in the new buffer, and vice versa; the two buffers' position, limit, and mark values will be independent.

The new buffer's position will be zero, its capacity and its limit will be the number of bytes remaining in this buffer, and its mark will be undefined. The new buffer will be read-only if, and only if, this buffer is read-only.

Returns

the newly create **ShortBuffer** (p. 2419) which the caller owns.

Implements **decaf::nio::ShortBuffer** (p. 2430).

The documentation for this class was generated from the following file:

- src/main/decaf/internal/nio/**ShortArrayBuffer.h**

6.480 decaf::nio::ShortBuffer Class Reference

This class defines four categories of operations upon short buffers:

```
#include <src/main/decaf/nio/ShortBuffer.h>
```

Inheritance diagram for decaf::nio::ShortBuffer:

Public Member Functions

- virtual **~ShortBuffer** ()
- virtual std::string **toString** () const
- virtual short * **array** ()=0

Returns the short array that backs this buffer (optional operation).

- virtual int **arrayOffset** ()=0

Returns the offset within this buffer's backing array of the first element of the buffer (optional operation).

- virtual **ShortBuffer** * **asReadOnlyBuffer** () const =0

Creates a new, read-only short buffer that shares this buffer's content.

- virtual **ShortBuffer** & **compact** ()=0

Compacts this buffer.

- virtual **ShortBuffer** * **duplicate** ()=0

Creates a new short buffer that shares this buffer's content.

- virtual short **get** ()=0

Relative get method.

- virtual short **get** (int index) const =0

Absolute get method.

- **ShortBuffer** & **get** (std::vector< short > buffer)

Relative bulk get method.

- **ShortBuffer** & **get** (short *buffer, int size, int offset, int length)

Relative bulk get method.

- virtual bool **hasArray** () const =0

Tells whether or not this buffer is backed by an accessible short array.

- **ShortBuffer** & **put** (**ShortBuffer** &src)

This method transfers the shorts remaining in the given source buffer into this buffer.

- **ShortBuffer** & **put** (const short *buffer, int size, int offset, int length)

This method transfers shorts into this buffer from the given source array.

- **ShortBuffer** & **put** (std::vector< short > &buffer)

This method transfers the entire content of the given source shorts array into this buffer.

- virtual **ShortBuffer** & **put** (short value)=0

Writes the given shorts into this buffer at the current position, and then increments the position.

- virtual **ShortBuffer** & **put** (int index, short value)=0

Writes the given shorts into this buffer at the given index.

- virtual **ShortBuffer** * **slice** () const =0

*Creates a new **ShortBuffer** (p. 2419) whose content is a shared subsequence of this buffer's content.*

- virtual int **compareTo** (const **ShortBuffer** &value) const

- virtual bool **equals** (const **ShortBuffer** &value) const

- virtual bool **operator==** (const **ShortBuffer** &value) const

- virtual bool **operator<** (const **ShortBuffer** &value) const

Static Public Member Functions

- static **ShortBuffer** * **allocate** (int **capacity**)
Allocates a new Double buffer.
- static **ShortBuffer** * **wrap** (short ***array**, int **size**, int **offset**, int **length**)
*Wraps the passed buffer with a new **ShortBuffer** (p. 2419).*
- static **ShortBuffer** * **wrap** (std::vector< short > &**buffer**)
*Wraps the passed STL short Vector in a **ShortBuffer** (p. 2419).*

Protected Member Functions

- **ShortBuffer** (int **capacity**)
*Creates a **ShortBuffer** (p. 2419) object that has its backing array allocated internally and is then owned and deleted when this object is deleted.*

6.480.1 Detailed Description

This class defines four categories of operations upon short buffers:

o Absolute and relative get and put methods that read and write single shorts; o Relative bulk get methods that transfer contiguous sequences of shorts from this buffer into an array; and o Relative bulk put methods that transfer contiguous sequences of shorts from a short array or some other short buffer into this buffer o Methods for compacting, duplicating, and slicing a short buffer.

Double buffers can be created either by allocation, which allocates space for the buffer's content, by wrapping an existing short array into a buffer, or by creating a view of an existing byte buffer

Methods in this class that do not otherwise have a value to return are specified to return the buffer upon which they are invoked. This allows method invocations to be chained.

6.480.2 Constructor & Destructor Documentation

6.480.2.1 decaf::nio::ShortBuffer::ShortBuffer (int *capacity*) [protected]

Creates a **ShortBuffer** (p. 2419) object that has its backing array allocated internally and is then owned and deleted when this object is deleted.

The array is initially created with all elements initialized to zero.

Parameters

<i>capacity</i>	The size and limit of the Buffer (p. 582) in doubles
-----------------	---

Exceptions

<i>IllegalArgument-Exception</i>	if capacity is negative.
----------------------------------	--------------------------

6.480.2.2 `virtual decaf::nio::ShortBuffer::~~ShortBuffer () [inline, virtual]`

6.480.3 Member Function Documentation

6.480.3.1 `static ShortBuffer* decaf::nio::ShortBuffer::allocate (int capacity) [static]`

Allocates a new Double buffer.

The new buffer's position will be zero, its limit will be its capacity, and its mark will be undefined. It will have a backing array, and its array offset will be zero.

Parameters

<i>capacity</i>	The size of the Double buffer in shorts.
-----------------	--

Returns

the **ShortBuffer** (p. 2419) that was allocated, caller owns.

6.480.3.2 `virtual short* decaf::nio::ShortBuffer::array () [pure virtual]`

Returns the short array that backs this buffer (optional operation).

Modifications to this buffer's content will cause the returned array's content to be modified, and vice versa.

Invoke the `hasArray` method before invoking this method in order to ensure that this buffer has an accessible backing array.

Returns

the array that backs this **Buffer** (p. 582)

Exceptions

<i>ReadOnlyBuffer-Exception</i> (p. 2244)	if this Buffer (p. 582) is read only.
<i>Unsupported-OperationException</i>	if the underlying store has no array.

Implemented in **decaf::internal::nio::ShortArrayBuffer** (p. 2414).

6.480.3.3 virtual int decaf::nio::ShortBuffer::arrayOffset () [pure virtual]

Returns the offset within this buffer's backing array of the first element of the buffer (optional operation).

Invoke the `hasArray` method before invoking this method in order to ensure that this buffer has an accessible backing array.

Returns

The offset into the backing array where index zero starts.

Exceptions

<i>ReadOnlyBufferException</i> (p. 2244)	if this Buffer (p. 582) is read only.
<i>UnsupportedOperationException</i>	if the underlying store has no array.

Implemented in **decaf::internal::nio::ShortArrayBuffer** (p. 2414).

6.480.3.4 virtual ShortBuffer* decaf::nio::ShortBuffer::asReadOnlyBuffer () const [pure virtual]

Creates a new, read-only short buffer that shares this buffer's content.

The content of the new buffer will be that of this buffer. Changes to this buffer's content will be visible in the new buffer; the new buffer itself, however, will be read-only and will not allow the shared content to be modified. The two buffers' position, limit, and mark values will be independent.

If this buffer is itself read-only then this method behaves in exactly the same way as the `duplicate` method.

The new buffer's capacity, limit, position, and mark values will be identical to those of this buffer.

Returns

The new, read-only short buffer which the caller then owns.

Implemented in **decaf::internal::nio::ShortArrayBuffer** (p. 2415).

6.480.3.5 virtual ShortBuffer& decaf::nio::ShortBuffer::compact () [pure virtual]

Compacts this buffer.

The bytes between the buffer's current position and its limit, if any, are copied to the beginning of the buffer. That is, the byte at index `p = position()` (p. 587) is copied to

index zero, the byte at index $p + 1$ is copied to index one, and so forth until the byte at index `limit()` (p. 586) - 1 is copied to index $n = \text{limit}() - 1 - p$. The buffer's position is then set to $n+1$ and its limit is set to its capacity. The mark, if defined, is discarded.

The buffer's position is set to the number of bytes copied, rather than to zero, so that an invocation of this method can be followed immediately by an invocation of another relative put method.

Returns

a reference to this **ShortBuffer** (p. 2419).

Exceptions

ReadOnlyBuffer-Exception (p. 2244)	if this buffer is read-only.
--	------------------------------

Implemented in **decaf::internal::nio::ShortArrayBuffer** (p. 2415).

6.480.3.6 `virtual int decaf::nio::ShortBuffer::compareTo (const ShortBuffer & value)`
`const [virtual]`

6.480.3.7 `virtual ShortBuffer* decaf::nio::ShortBuffer::duplicate ()` [pure
`virtual]`

Creates a new short buffer that shares this buffer's content.

The content of the new buffer will be that of this buffer. Changes to this buffer's content will be visible in the new buffer, and vice versa; the two buffers' position, limit, and mark values will be independent.

The new buffer's capacity, limit, position, and mark values will be identical to those of this buffer. The new buffer will be read-only if, and only if, this buffer is read-only.

Returns

a new short **Buffer** (p. 582) which the caller owns.

Implemented in **decaf::internal::nio::ShortArrayBuffer** (p. 2416).

6.480.3.8 `virtual bool decaf::nio::ShortBuffer::equals (const ShortBuffer & value)`
`const [virtual]`

6.480.3.9 `virtual short decaf::nio::ShortBuffer::get ()` [pure virtual]

Relative get method.

Reads the value at this buffer's current position, and then increments the position.

Returns

the short at the current position.

Exceptions

<i>BufferUnderflow-Exception</i> (p. 611)	if there no more data to return.
---	----------------------------------

Implemented in **decaf::internal::nio::ShortArrayBuffer** (p. 2416).

6.480.3.10 virtual short decaf::nio::ShortBuffer::get (int *index*) const [pure virtual]

Absolute get method.

Reads the value at the given index.

Parameters

<i>index</i>	The index in the Buffer (p. 582) where the short is to be read.
--------------	--

Returns

the short that is located at the given index.

Exceptions

<i>IndexOutOfBoundsException</i>	if index is not smaller than the buffer's limit, or the index is negative.
---	--

Implemented in **decaf::internal::nio::ShortArrayBuffer** (p. 2416).

6.480.3.11 ShortBuffer& decaf::nio::ShortBuffer::get (std::vector< short > *buffer*)

Relative bulk get method.

This method transfers values from this buffer into the given destination vector. An invocation of this method of the form src.get(a) behaves in exactly the same way as the invocation. The vector must be sized to the amount of data that is to be read, that is to say, the caller should call buffer.resize(N) before calling this get method.

Returns

a reference to this **Buffer** (p. 582).

Exceptions

<i>BufferUnderflow-Exception</i> (p. 611)	if there are fewer than length shorts remaining in this buffer.
---	---

6.480.3.12 ShortBuffer& decaf::nio::ShortBuffer::get (short * *buffer*, int *size*, int *offset*, int *length*)

Relative bulk get method.

This method transfers shorts from this buffer into the given destination array. If there are fewer shorts remaining in the buffer than are required to satisfy the request, that is, if `length > remaining()` (p. 588), then no bytes are transferred and a **BufferUnderflow-Exception** (p. 611) is thrown.

Otherwise, this method copies length shorts from this buffer into the given array, starting at the current position of this buffer and at the given offset in the array. The position of this buffer is then incremented by length.

Parameters

<i>buffer</i>	The pointer to an allocated buffer to fill.
<i>size</i>	The size of the buffer provided.
<i>offset</i>	The position in the buffer to start filling.
<i>length</i>	The amount of data to put in the passed buffer.

Returns

a reference to this **Buffer** (p. 582).

Exceptions

<i>BufferUnderflow-Exception</i> (p. 611)	if there are fewer than length shorts remaining in this buffer
<i>NullPointerException</i>	if the passed buffer is null.
<i>IndexOutOfBoundsException</i>	if the preconditions of size, offset, or length are not met.

6.480.3.13 virtual bool decaf::nio::ShortBuffer::hasArray () const [pure virtual]

Tells whether or not this buffer is backed by an accessible short array.

If this method returns true then the `array` and `arrayOffset` methods may safely be invoked. Subclasses should override this method if they do not have a backing array as this class always returns true.

Returns

true if, and only if, this buffer is backed by an array and is not read-only.

Implemented in **decaf::internal::nio::ShortArrayBuffer** (p. 2417).

6.480.3.14 virtual bool decaf::nio::ShortBuffer::operator< (const ShortBuffer & value)
const [virtual]

6.480.3.15 virtual bool decaf::nio::ShortBuffer::operator== (const ShortBuffer & value)
const [virtual]

6.480.3.16 ShortBuffer& decaf::nio::ShortBuffer::put (ShortBuffer & src)

This method transfers the shorts remaining in the given source buffer into this buffer.

If there are more shorts remaining in the source buffer than in this buffer, that is, if `src.remaining() > remaining()` (p. 588), then no shorts are transferred and a **BufferOverflowException** (p. 609) is thrown.

Otherwise, this method copies `n = src.remaining()` shorts from the given buffer into this buffer, starting at each buffer's current position. The positions of both buffers are then incremented by `n`.

Parameters

src	The buffer to take shorts from an place in this one.
-----	--

Returns

a reference to this buffer.

Exceptions

BufferOverflowException (p. 609)	if there is insufficient space in this buffer for the remaining shorts in the source buffer.
<i>IllegalArgumentException</i>	if the source buffer is this buffer.
ReadOnlyBufferException (p. 2244)	if this buffer is read-only.

6.480.3.17 ShortBuffer& decaf::nio::ShortBuffer::put (const short * buffer, int size, int offset, int length)

This method transfers shorts into this buffer from the given source array.

If there are more shorts to be copied from the array than remain in this buffer, that is,

if `length > remaining()` (p. 588), then no shorts are transferred and a **BufferOverflow-Exception** (p. 609) is thrown.

Otherwise, this method copies `length` bytes from the given array into this buffer, starting at the given offset in the array and at the current position of this buffer. The position of this buffer is then incremented by `length`.

Parameters

<i>buffer</i>	The array from which shorts are to be read.
<i>size</i>	The size of the buffer passed.
<i>offset</i>	The offset within the array of the first char to be read.
<i>length</i>	The number of shorts to be read from the given array.

Returns

a reference to this buffer.

Exceptions

BufferOverflow-Exception (p. 609)	if there is insufficient space in this buffer
ReadOnlyBuffer-Exception (p. 2244)	if this buffer is read-only
<i>NullPointerException</i>	if the passed buffer is null.
<i>IndexOutOfBoundsException</i>	if the preconditions of size, offset, or length are not met.

6.480.3.18 ShortBuffer& decaf::nio::ShortBuffer::put (std::vector< short > & buffer)

This method transfers the entire content of the given source shorts array into this buffer.

This is the same as calling `put(&buffer[0], 0, buffer.size()`.

Parameters

<i>buffer</i>	The buffer whose contents are copied to this ShortBuffer (p. 2419).
---------------	--

Returns

a reference to this buffer.

Exceptions

BufferOverflow-Exception (p. 609)	if there is insufficient space in this buffer.
---	--

<i>ReadOnlyBuffer-Exception</i> (p. 2244)	if this buffer is read-only.
---	------------------------------

6.480.3.19 virtual **ShortBuffer&** decaf::nio::ShortBuffer::put (short *value*) [pure virtual]

Writes the given shorts into this buffer at the current position, and then increments the position.

Parameters

<i>value</i>	The shorts value to be written.
--------------	---------------------------------

Returns

a reference to this buffer.

Exceptions

<i>BufferOverflow-Exception</i> (p. 609)	if this buffer's current position is not smaller than its limit.
<i>ReadOnlyBuffer-Exception</i> (p. 2244)	if this buffer is read-only.

Implemented in **decaf::internal::nio::ShortArrayBuffer** (p. 2417).

6.480.3.20 virtual **ShortBuffer&** decaf::nio::ShortBuffer::put (int *index*, short *value*) [pure virtual]

Writes the given shorts into this buffer at the given index.

Parameters

<i>index</i>	The position in the Buffer (p. 582) to write the data.
<i>value</i>	The shorts to write.

Returns

a reference to this buffer.

Exceptions

<i>IndexOutOfBoundsException</i>	if index greater than the buffer's limit minus the size of the type being written.
ReadOnlyBufferException (p. 2244)	if this buffer is read-only.

Implemented in **decaf::internal::nio::ShortArrayBuffer** (p. 2418).

6.480.3.21 `virtual ShortBuffer* decaf::nio::ShortBuffer::slice () const` [pure virtual]

Creates a new **ShortBuffer** (p. 2419) whose content is a shared subsequence of this buffer's content.

The content of the new buffer will start at this buffer's current position. Changes to this buffer's content will be visible in the new buffer, and vice versa; the two buffers' position, limit, and mark values will be independent.

The new buffer's position will be zero, its capacity and its limit will be the number of bytes remaining in this buffer, and its mark will be undefined. The new buffer will be read-only if, and only if, this buffer is read-only.

Returns

the newly create **ShortBuffer** (p. 2419) which the caller owns.

Implemented in **decaf::internal::nio::ShortArrayBuffer** (p. 2419).

6.480.3.22 `virtual std::string decaf::nio::ShortBuffer::toString () const` [virtual]

Returns

a std::string describing this object

6.480.3.23 `static ShortBuffer* decaf::nio::ShortBuffer::wrap (short * array, int size, int offset, int length)` [static]

Wraps the passed buffer with a new **ShortBuffer** (p. 2419).

The new buffer will be backed by the given short array; that is, modifications to the buffer will cause the array to be modified and vice versa. The new buffer's capacity will be array.length, its position will be offset, its limit will be offset + length, and its mark will be undefined. Its backing array will be the given array, and its array offset will be zero.

Parameters

<i>array</i>	The array that will back the new buffer.
<i>size</i>	The size of the passed in array.
<i>offset</i>	The offset of the subarray to be used.
<i>length</i>	The length of the subarray to be used.

Returns

a new **ShortBuffer** (p. 2419) that is backed by buffer, caller owns.

Exceptions

<i>NullPointerException</i>	if the array pointer is NULL.
<i>IndexOutOfBoundsException</i>	if the preconditions of size, offset, or length are not met.

6.480.3.24 `static ShortBuffer* decaf::nio::ShortBuffer::wrap (std::vector< short > & buffer) [static]`

Wraps the passed STL short Vector in a **ShortBuffer** (p. 2419).

The new buffer will be backed by the given short array; modifications to the buffer will cause the array to be modified and vice versa. The new buffer's capacity and limit will be buffer.size(), its position will be zero, and its mark will be undefined. Its backing array will be the given array, and its array offset will be zero.

Parameters

<i>buffer</i>	The vector that will back the new buffer, the vector must have been sized to the desired size already by calling vector.resize(N).
---------------	--

Returns

a new **ShortBuffer** (p. 2419) that is backed by buffer, caller owns.

The documentation for this class was generated from the following file:

- src/main/decaf/nio/**ShortBuffer.h**

6.481 activemq::commands::ShutdownInfo Class Reference

```
#include <src/main/activemq/commands/ShutdownInfo.h>
```

Inheritance diagram for activemq::commands::ShutdownInfo:

Public Member Functions

- **ShutdownInfo** ()
- virtual **~ShutdownInfo** ()
- virtual unsigned char **getDataStructureType** () const

Get the **DataSet** (p. 1133) Type as defined in *CommandTypes.h*.

- virtual **ShutdownInfo** * **cloneDataSet** () const

Clone this object and return a new instance that the caller now owns, this will be an exact copy of this one.

- virtual void **copyDataSet** (const **DataSet** *src)

Copy the contents of the passed object into this objects members, overwriting any existing data.

- virtual std::string **toString** () const

Returns a string containing the information for this **DataSet** (p. 1133) such as its type and value of its elements.

- virtual bool **equals** (const **DataSet** *value) const

Compares the **DataSet** (p. 1133) passed in to this one, and returns if they are equivalent.

- virtual bool **isShutdownInfo** () const

- virtual **Pointer**< **Command** > **visit** (**activemq::state::CommandVisitor** *visitor)

Allows a Visitor to visit this command and return a response to the command based on the command type being visited.

Static Public Attributes

- static const unsigned char **ID_SHUTDOWNINFO** = 11

6.481.1 Constructor & Destructor Documentation

- 6.481.1.1 **activemq::commands::ShutdownInfo::ShutdownInfo** ()

- 6.481.1.2 virtual **activemq::commands::ShutdownInfo::~~ShutdownInfo** ()
[virtual]

6.481.2 Member Function Documentation

- 6.481.2.1 virtual **ShutdownInfo*** **activemq::commands::ShutdownInfo::cloneDataSet** () const [virtual]

Clone this object and return a new instance that the caller now owns, this will be an exact copy of this one.

Returns

new copy of this object.

Implements **activemq::commands::DataSet** (p. 1133).

6.481.2.2 virtual void **activemq::commands::ShutdownInfo::copyDataStructure** (
const **DataStructure** * *src*) [virtual]

Copy the contents of the passed object into this objects members, overwriting any existing data.

Returns

src - Source Object

Reimplemented from **activemq::commands::BaseCommand** (p. 493).

6.481.2.3 virtual bool **activemq::commands::ShutdownInfo::equals** (const
DataStructure * *value*) const [virtual]

Compares the **DataStructure** (p. 1133) passed in to this one, and returns if they are equivalent.

Equivalent here means that they are of the same type, and that each element of the objects are the same.

Returns

true if **DataStructure** (p. 1133)'s are Equal.

Reimplemented from **activemq::commands::BaseCommand** (p. 494).

6.481.2.4 virtual unsigned char **activemq::commands::ShutdownInfo::getData-**
StructureType () const [virtual]

Get the **DataStructure** (p. 1133) Type as defined in CommandTypes.h.

Returns

The type of the data structure

Implements **activemq::commands::DataStructure** (p. 1137).

6.481.2.5 virtual bool **activemq::commands::ShutdownInfo::isShutdownInfo** ()
const [inline, virtual]

Returns

an answer of true to the **isShutdownInfo()** (p. 2433) query.

Reimplemented from **activemq::commands::BaseCommand** (p. 497).

6.481.2.6 `virtual std::string activemq::commands::ShutdownInfo::toString () const`
`[virtual]`

Returns a string containing the information for this **DataStructure** (p. 1133) such as its type and value of its elements.

Returns

formatted string useful for debugging.

Reimplemented from **activemq::commands::BaseCommand** (p. 498).

6.481.2.7 `virtual Pointer<Command> activemq::commands::ShutdownInfo::visit (`
`activemq::state::CommandVisitor * visitor) [virtual]`

Allows a Visitor to visit this command and return a response to the command based on the command type being visited.

The command will call the proper processXXX method in the visitor.

Returns

a **Response** (p. 2298) to the visitor being called or NULL if no response.

Implements **activemq::commands::Command** (p. 871).

6.481.3 Field Documentation

6.481.3.1 `const unsigned char activemq::commands::ShutdownInfo::ID_SHUTDOWN-`
`INFO = 11 [static]`

The documentation for this class was generated from the following file:

- `src/main/activemq/commands/ShutdownInfo.h`

6.482 **activemq::wireformat::openwire::marshal::generated::-** **ShutdownInfoMarshaller Class Reference**

Marshaling code for Open Wire Format for **ShutdownInfoMarshaller** (p. 2434).

```
#include <src/main/activemq/wireformat/openwire/marshal/generated/-
ShutdownInfoMarshaller.h>
```

Inheritance diagram for **activemq::wireformat::openwire::marshal::generated::-**
ShutdownInfoMarshaller:

6.482

activemq::wireformat::openwire::marshal::generated::ShutdownInfoMarshaller

Class Reference

2441

Public Member Functions

- **ShutdownInfoMarshaller** ()
- virtual **~ShutdownInfoMarshaller** ()
- virtual **commands::DataStructure * createObject** () const
Creates a new instance of the class that this class is a marshaling director for.
- virtual unsigned char **getDataStructureType** () const
Gets the DataStructureType that this class marshals/unmarshals.
- virtual void **tightUnmarshal** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataInputStream** *dataIn, **utils::BooleanStream** *bs)
Tight Un-marhsal to the given stream.
- virtual int **tightMarshal1** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **utils::BooleanStream** *bs)
Tight Marshal to the given stream.
- virtual void **tightMarshal2** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataOutputStream** *dataOut, **utils::BooleanStream** *bs)
Tight Marshal to the given stream.
- virtual void **looseUnmarshal** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataInputStream** *dataIn)
Loose Un-marhsal to the given stream.
- virtual void **looseMarshal** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataOutputStream** *dataOut)
Tight Marshal to the given stream.

6.482.1 Detailed Description

Marshaling code for Open Wire Format for **ShutdownInfoMarshaller** (p. 2434).

NOTE!: This file is auto generated - do not modify! if you need to make a change, please see the Java Classes in the activemq-openwire-generator module

6.482.2 Constructor & Destructor Documentation

6.482.2.1 **activemq::wireformat::openwire::marshal::generated::ShutdownInfoMarshaller::ShutdownInfoMarshaller** ()
[inline]

6.482.2.2 **virtual activemq::wireformat::openwire::marshal::generated::ShutdownInfoMarshaller::~~ShutdownInfoMarshaller** () [inline, virtual]

6.482.3 Member Function Documentation

6.482.3.1 `virtual commands::DataStructure* activemq::wireformat::openwire-
::marshal::generated::ShutdownInfoMarshaller::createObject () const`
[virtual]

Creates a new instance of the class that this class is a marshaling director for.

Returns

newly allocated Command

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller**
(p. 1120).

6.482.3.2 `virtual unsigned char activemq::wireformat::openwire::marshal-
::generated::ShutdownInfoMarshaller::getDataStructureType () const`
[virtual]

Gets the DataStructureType that this class marshals/unmarshals.

Returns

byte Id of this classes DataStructureType

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller**
(p. 1122).

6.482.3.3 `virtual void activemq::wireformat::openwire::marshal::generated:-
ShutdownInfoMarshaller::looseMarshal (OpenWireFormat * format,
commands::DataStructure * command, decaf::io::DataOutputStream * ds
)` [virtual]

Tight Marshal to the given stream.

Parameters

<i>format</i>	- The OpenWireFormat properties
<i>command</i>	- the object to Marshal
<i>ds</i>	- DataOutputStream to marshal to

Exceptions

<i>IOException</i>	if an error occurs.
--------------------	---------------------

Reimplemented from **activemq::wireformat::openwire::marshal::generated::Base-
CommandMarshaller** (p. 500).

6.482

activemq::wireformat::openwire::marshal::generated::ShutdownInfoMarshaller

Class Reference

2443

6.482.3.4 virtual void **activemq::wireformat::openwire::marshal::generated::ShutdownInfoMarshaller::looseUnmarshal** (**OpenWireFormat** * *format*, **commands::DataStructure** * *command*, **decaf::io::DataInputStream** * *dis*)
[virtual]

Loose Un-marhsal to the given stream.

Parameters

<i>format</i>	- The OpenWireFormat properties
<i>command</i>	- the object to Un-Marshal
<i>dis</i>	- the DataInputStream to Un-Marshal from

Exceptions

<i>IOException</i>	if an error occurs.
--------------------	---------------------

Reimplemented from **activemq::wireformat::openwire::marshal::generated::Base-CommandMarshaller** (p. 501).

6.482.3.5 virtual int **activemq::wireformat::openwire::marshal::generated::ShutdownInfoMarshaller::tightMarshal1** (**OpenWireFormat** * *format*, **commands::DataStructure** * *command*, **utils::BooleanStream** * *bs*)
[virtual]

Tight Marhsal to the given stream.

Parameters

<i>format</i>	The OpenWireFormat properties
<i>command</i>	The object to Marshal
<i>bs</i>	The boolean stream to marshal to.

Exceptions

<i>IOException</i>	if an error occurs.
--------------------	---------------------

Reimplemented from **activemq::wireformat::openwire::marshal::generated::Base-CommandMarshaller** (p. 503).

6.482.3.6 virtual void **activemq::wireformat::openwire::marshal::generated::ShutdownInfoMarshaller::tightMarshal2** (**OpenWireFormat** * *format*, **commands::DataStructure** * *command*, **decaf::io::DataOutputStream** * *ds*, **utils::BooleanStream** * *bs*) [virtual]

Tight Marhsal to the given stream.

Parameters

<i>format</i>	- The OpenwireFormat properties
<i>command</i>	- the object to Marshal
<i>ds</i>	- the DataOutputStream to Marshal to
<i>bs</i>	- boolean stream to marshal to.

Exceptions

<i>IOException</i>	if an error occurs.
--------------------	---------------------

Reimplemented from **activemq::wireformat::openwire::marshal::generated::Base-CommandMarshaller** (p. 504).

6.482.3.7 virtual void **activemq::wireformat::openwire::marshal::generated::ShutdownInfoMarshaller::tightUnmarshal** (**OpenWireFormat** * *format*, **commands::DataStructure** * *command*, **decaf::io::DataInputStream** * *dis*, **utils::BooleanStream** * *bs*) [virtual]

Tight Un-marhsal to the given stream.

Parameters

<i>format</i>	- The OpenwireFormat properties
<i>command</i>	- the object to Un-Marshal
<i>dis</i>	- the DataInputStream to Un-Marshal from
<i>bs</i>	- boolean stream to unmarshal from.

Exceptions

<i>IOException</i>	if an error occurs.
--------------------	---------------------

Reimplemented from **activemq::wireformat::openwire::marshal::generated::Base-CommandMarshaller** (p. 505).

The documentation for this class was generated from the following file:

- `src/main/activemq/wireformat/openwire/marshal/generated/ShutdownInfo-Marshaller.h`

6.483 decaf::security::SignatureException Class Reference

```
#include <src/main/decaf/security/SignatureException.h>
```

Inheritance diagram for `decaf::security::SignatureException`:

Public Member Functions

- **SignatureException** () throw ()
Default Constructor.
- **SignatureException** (const **Exception** &ex) throw ()
Conversion Constructor from some other Exception.
- **SignatureException** (const **SignatureException** &ex) throw ()
Copy Constructor.
- **SignatureException** (const char *file, const int lineNumber, const std::exception *cause, const char *msg,...) throw ()
Constructor - Initializes the file name and line number where this message occurred.
- **SignatureException** (const std::exception *cause) throw ()
Constructor.
- **SignatureException** (const char *file, const int lineNumber, const char *msg,...) throw ()
Constructor - Initializes the file name and line number where this message occurred.
- virtual **SignatureException** * clone () const
Clones this exception.
- virtual ~**SignatureException** () throw ()

6.483.1 Constructor & Destructor Documentation

6.483.1.1 **decaf::security::SignatureException::SignatureException** () throw ()
[inline]

Default Constructor.

6.483.1.2 **decaf::security::SignatureException::SignatureException** (const **Exception** & ex) throw () [inline]

Conversion Constructor from some other Exception.

Parameters

ex	An exception that should become this type of Exception
----	--

6.483.1.3 **decaf::security::SignatureException::SignatureException** (const **SignatureException** & ex) throw () [inline]

Copy Constructor.

Parameters

ex	An exception that should become this type of Exception
----	--

6.483.1.4 **decaf::security::SignatureException::SignatureException** (*const char * file*, *const int lineNumber*, *const std::exception * cause*, *const char * msg*, ...) *throw ()* [inline]

Constructor - Initializes the file name and line number where this message occurred.

Sets the message to report, using an optional list of arguments to parse into the message

Parameters

<i>file</i>	The file name where exception occurs
<i>lineNumber</i>	The line number where the exception occurred.
<i>cause</i>	The exception that was the cause for this one to be thrown.
<i>msg</i>	The message to report
...	list of primitives that are formatted into the message

6.483.1.5 **decaf::security::SignatureException::SignatureException** (*const std::exception * cause*) *throw ()* [inline]

Constructor.

Parameters

<i>cause</i>	Pointer to the exception that caused this one to be thrown, the object is cloned caller retains ownership.
--------------	--

6.483.1.6 **decaf::security::SignatureException::SignatureException** (*const char * file*, *const int lineNumber*, *const char * msg*, ...) *throw ()* [inline]

Constructor - Initializes the file name and line number where this message occurred.

Sets the message to report, using an optional list of arguments to parse into the message

Parameters

<i>file</i>	name where exception occurs
<i>lineNumber</i>	line number where the exception occurred.
<i>msg</i>	message to report
...	list of primitives that are formatted into the message

6.483.1.7 **virtual decaf::security::SignatureException::~SignatureException** () *throw ()* [inline, virtual]

6.483.2 Member Function Documentation

6.483.2.1 virtual **SignatureException*** **decaf::security::SignatureException::clone** (
) const [inline, virtual]

Clones this exception.

This is useful for cases where you need to preserve the type of the original exception as well as the message. All subclasses should override.

Returns

A deep copy of this exception.

Reimplemented from **decaf::security::GeneralSecurityException** (p. 1397).

The documentation for this class was generated from the following file:

- src/main/decaf/security/**SignatureException.h**

6.484 decaf::util::logging::SimpleFormatter Class Reference

Print a brief summary of the **LogRecord** (p. 1719) in a human readable format.

```
#include <src/main/decaf/util/logging/SimpleFormatter.h>
```

Inheritance diagram for decaf::util::logging::SimpleFormatter:

Public Member Functions

- **SimpleFormatter** ()
- virtual **~SimpleFormatter** ()
- virtual std::string **format** (const **LogRecord** &record) const
Format the given log record and return the formatted string.

6.484.1 Detailed Description

Print a brief summary of the **LogRecord** (p. 1719) in a human readable format.

The summary will typically be 1 or 2 lines.

Since

1.0

6.484.2 Constructor & Destructor Documentation

6.484.2.1 **decaf::util::logging::SimpleFormatter::SimpleFormatter** ()

6.484.2.2 virtual **decaf::util::logging::SimpleFormatter::~SimpleFormatter** ()
[virtual]

6.484.3 Member Function Documentation

6.484.3.1 virtual std::string **decaf::util::logging::SimpleFormatter::format** (const
LogRecord & record) const [virtual]

Format the given log record and return the formatted string.

Parameters

<i>record</i>	The Log Record to Format.
---------------	---------------------------

Implements **decaf::util::logging::Formatter** (p. 1388).

The documentation for this class was generated from the following file:

- src/main/decaf/util/logging/**SimpleFormatter.h**

6.485 decaf::util::logging::SimpleLogger Class Reference

```
#include <src/main/decaf/util/logging/SimpleLogger.h>
```

Public Member Functions

- **SimpleLogger** (const std::string &name)
Constructor.
- virtual **~SimpleLogger** ()
Destructor.
- virtual void **mark** (const std::string &message)
*Log a Mark Block **Level** (p. 1616) Log.*
- virtual void **debug** (const std::string &file, const int line, const std::string &message)
*Log a Debug **Level** (p. 1616) Log.*
- virtual void **info** (const std::string &file, const int line, const std::string &message)
*Log a Informational **Level** (p. 1616) Log.*
- virtual void **warn** (const std::string &file, const int line, const std::string &message)
*Log a Warning **Level** (p. 1616) Log.*
- virtual void **error** (const std::string &file, const int line, const std::string &message)
*Log a Error **Level** (p. 1616) Log.*

- virtual void **fatal** (const std::string &file, const int line, const std::string &message)
*Log a Fatal **Level** (p. 1616) Log.*
- virtual void **log** (const std::string &message)
No-frills log.

6.485.1 Constructor & Destructor Documentation

6.485.1.1 **decaf::util::logging::SimpleLogger::SimpleLogger** (const std::string &name)

Constructor.

6.485.1.2 **virtual decaf::util::logging::SimpleLogger::~SimpleLogger** ()
[virtual]

Destructor.

6.485.2 Member Function Documentation

6.485.2.1 **virtual void decaf::util::logging::SimpleLogger::debug** (const std::string &file, const int line, const std::string &message) [virtual]

Log a Debug **Level** (p. 1616) Log.

6.485.2.2 **virtual void decaf::util::logging::SimpleLogger::error** (const std::string &file, const int line, const std::string &message) [virtual]

Log a Error **Level** (p. 1616) Log.

6.485.2.3 **virtual void decaf::util::logging::SimpleLogger::fatal** (const std::string &file, const int line, const std::string &message) [virtual]

Log a Fatal **Level** (p. 1616) Log.

6.485.2.4 **virtual void decaf::util::logging::SimpleLogger::info** (const std::string &file, const int line, const std::string &message) [virtual]

Log a Informational **Level** (p. 1616) Log.

6.485.2.5 **virtual void decaf::util::logging::SimpleLogger::log** (const std::string &message) [virtual]

No-frills log.

6.485.2.6 `virtual void decaf::util::logging::SimpleLogger::mark (const std::string & message) [virtual]`

Log a Mark Block **Level** (p. 1616) Log.

6.485.2.7 `virtual void decaf::util::logging::SimpleLogger::warn (const std::string & file, const int line, const std::string & message) [virtual]`

Log a Warning **Level** (p. 1616) Log.

The documentation for this class was generated from the following file:

- `src/main/decaf/util/logging/SimpleLogger.h`

6.486 `activemq::core::SimplePriorityMessageDispatchChannel` - Class Reference

```
#include <src/main/activemq/core/SimplePriorityMessageDispatchChannel.h>
```

Inheritance diagram for `activemq::core::SimplePriorityMessageDispatchChannel`:

Public Member Functions

- `SimplePriorityMessageDispatchChannel ()`
- `virtual ~SimplePriorityMessageDispatchChannel ()`
- `virtual void enqueue (const Pointer< MessageDispatch > &message)`
Add a Message to the Channel behind all pending message.
- `virtual void enqueueFirst (const Pointer< MessageDispatch > &message)`
Add a message to the front of the Channel.
- `virtual bool isEmpty () const`
- `virtual bool isClosed () const`
- `virtual bool isRunning () const`
- `virtual Pointer< MessageDispatch > dequeue (long long timeout)`
Used to get an enqueued message.
- `virtual Pointer< MessageDispatch > dequeueNoWait ()`
Used to get an enqueued message if there is one queued right now.
- `virtual Pointer< MessageDispatch > peek () const`
Peek in the Queue and return the first message in the Channel without removing it from the channel.
- `virtual void start ()`
Starts dispatch of messages from the Channel.

6.486 activemq::core::SimplePriorityMessageDispatchChannel Class Reference

- virtual void **stop** ()
Stops dispatch of message from the Channel.
- virtual void **close** ()
Close this channel no messages will be dispatched after this method is called.
- virtual void **clear** ()
Clear the Channel, all pending messages are removed.
- virtual int **size** () const
- virtual std::vector< **Pointer** < **MessageDispatch** > > **removeAll** ()
Remove all messages that are currently in the Channel and return them as a list of Messages.
- virtual void **lock** () throw (decaf::lang::exceptions::RuntimeException)
Locks the object.
- virtual bool **tryLock** () throw (decaf::lang::exceptions::RuntimeException)
Attempts to Lock the object, if the lock is already held by another thread than this method returns false.
- virtual void **unlock** () throw (decaf::lang::exceptions::RuntimeException)
Unlocks the object.
- virtual void **wait** () throw (decaf::lang::exceptions::RuntimeException, decaf::lang::exceptions::IllegalMonitorStateException, decaf::lang::exceptions::InterruptedException)
Waits on a signal from this object, which is generated by a call to Notify.
- virtual void **wait** (long long millisecs) throw (decaf::lang::exceptions::RuntimeException, decaf::lang::exceptions::IllegalMonitorStateException, decaf::lang::exceptions::InterruptedException)
Waits on a signal from this object, which is generated by a call to Notify.
- virtual void **wait** (long long millisecs, int nanos) throw (decaf::lang::exceptions::RuntimeException, decaf::lang::exceptions::IllegalArgumentException, decaf::lang::exceptions::IllegalMonitorStateException, decaf::lang::exceptions::InterruptedException)
Waits on a signal from this object, which is generated by a call to Notify.
- virtual void **notify** () throw (decaf::lang::exceptions::RuntimeException, decaf::lang::exceptions::IllegalMonitorStateException)
Signals a waiter on this object that it can now wake up and continue.
- virtual void **notifyAll** () throw (decaf::lang::exceptions::RuntimeException, decaf::lang::exceptions::IllegalMonitorStateException)
Signals the waiters on this object that it can now wake up and continue.

6.486.1 Constructor & Destructor Documentation

6.486.1.1 **activemq::core::SimplePriorityMessageDispatchChannel::SimplePriorityMessageDispatchChannel** ()

6.486.1.2 **virtual activemq::core::SimplePriorityMessageDispatchChannel::~SimplePriorityMessageDispatchChannel** ()
[virtual]

6.486.2 Member Function Documentation

6.486.2.1 **virtual void activemq::core::SimplePriorityMessageDispatchChannel::clear ()** [virtual]

Clear the Channel, all pending messages are removed.

Implements **activemq::core::MessageDispatchChannel** (p. 1887).

6.486.2.2 **virtual void activemq::core::SimplePriorityMessageDispatchChannel::close ()** [virtual]

Close this channel no messages will be dispatched after this method is called.

Implements **activemq::core::MessageDispatchChannel** (p. 1887).

6.486.2.3 **virtual Pointer<MessageDispatch> activemq::core::SimplePriorityMessageDispatchChannel::dequeue (long long timeout)** [virtual]

Used to get an enqueued message.

The amount of time this method blocks is based on the timeout value. - if timeout==-1 then it blocks until a message is received. - if timeout==0 then it tries to not block at all, it returns a message if it is available - if timeout>0 then it blocks up to timeout amount of time. Expired messages will be consumed by this method.

Returns

null if we timeout or if the consumer is closed.

Exceptions

<i>ActiveMQException</i>

Implements **activemq::core::MessageDispatchChannel** (p. 1887).

6.486.2.4 **virtual Pointer<MessageDispatch> activemq::core::SimplePriorityMessageDispatchChannel::dequeueNoWait ()** [virtual]

Used to get an enqueued message if there is one queued right now.

If there is no waiting message then this method returns Null.

Returns

a message if there is one in the queue.

Implements **activemq::core::MessageDispatchChannel** (p. 1888).

6.486 `activemq::core::SimplePriorityMessageDispatchChannel` Class Reference 2453

6.486.2.5 `virtual void activemq::core::SimplePriorityMessageDispatchChannel::enqueue (const Pointer< MessageDispatch > & message)`
[virtual]

Add a Message to the Channel behind all pending message.

Parameters

<i>message</i>	- The message to add to the Channel.
----------------	--------------------------------------

Implements `activemq::core::MessageDispatchChannel` (p. 1888).

6.486.2.6 `virtual void activemq::core::SimplePriorityMessageDispatchChannel::enqueueFirst (const Pointer< MessageDispatch > & message)`
[virtual]

Add a message to the front of the Channel.

Parameters

<i>message</i>	- The Message to add to the front of the Channel.
----------------	---

Implements `activemq::core::MessageDispatchChannel` (p. 1888).

6.486.2.7 `virtual bool activemq::core::SimplePriorityMessageDispatchChannel::isClosed () const` [inline, virtual]

Returns

has the Queue been closed.

Implements `activemq::core::MessageDispatchChannel` (p. 1888).

6.486.2.8 `virtual bool activemq::core::SimplePriorityMessageDispatchChannel::isEmpty () const` [virtual]

Returns

true if there are no messages in the Channel.

Implements `activemq::core::MessageDispatchChannel` (p. 1889).

6.486.2.9 `virtual bool activemq::core::SimplePriorityMessageDispatchChannel::isRunning () const` [inline, virtual]

Returns

true if the Channel currently running and will dequeue message.

Implements **activemq::core::MessageDispatchChannel** (p. 1889).

6.486.2.10 **virtual void activemq::core::SimplePriorityMessageDispatchChannel-
::lock () throw (decaf::lang::exceptions::RuntimeException)**
[inline, virtual]

Locks the object.

Exceptions

<i>RuntimeException</i>	if an error occurs while locking the object.
-------------------------	--

Implements **decaf::util::concurrent::Synchronizable** (p. 2640).

6.486.2.11 **virtual void activemq::core::SimplePriorityMessageDispatchChannel-
::notify () throw (decaf::lang::exceptions::RuntimeException,
decaf::lang::exceptions::IllegalMonitorStateException)** [inline,
virtual]

Signals a waiter on this object that it can now wake up and continue.

Must have this object locked before calling.

Exceptions

<i>IllegalMonitorStateException</i>	- if the current thread is not the owner of the the Synchronizable Object.
<i>RuntimeException</i>	if an error occurs while notifying one of the waiting threads.

Implements **decaf::util::concurrent::Synchronizable** (p. 2641).

6.486.2.12 **virtual void activemq::core::SimplePriorityMessageDispatchChannel-
::notifyAll () throw (decaf::lang::exceptions::RuntimeException,
decaf::lang::exceptions::IllegalMonitorStateException)** [inline,
virtual]

Signals the waiters on this object that it can now wake up and continue.

Must have this object locked before calling.

Exceptions

<i>IllegalMonitorStateException</i>	- if the current thread is not the owner of the the Synchronizable Object.
<i>RuntimeException</i>	if an error occurs while notifying the waiting threads.

6.486 `activemq::core::SimplePriorityMessageDispatchChannel` Class Reference

Implements `decaf::util::concurrent::Synchronizable` (p. 2642).

6.486.2.13 `virtual Pointer<MessageDispatch> activemq::core::SimplePriorityMessageDispatchChannel::peek () const`
[virtual]

Peek in the Queue and return the first message in the Channel without removing it from the channel.

Returns

a message if there is one in the queue.

Implements `activemq::core::MessageDispatchChannel` (p. 1889).

6.486.2.14 `virtual std::vector< Pointer<MessageDispatch> > activemq::core::SimplePriorityMessageDispatchChannel::removeAll ()`
[virtual]

Remove all messages that are currently in the Channel and return them as a list of Messages.

Returns

a list of Messages that was previously in the Channel.

Implements `activemq::core::MessageDispatchChannel` (p. 1889).

6.486.2.15 `virtual int activemq::core::SimplePriorityMessageDispatchChannel::size () const` [virtual]

Returns

the number of Messages currently in the Channel.

Implements `activemq::core::MessageDispatchChannel` (p. 1890).

6.486.2.16 `virtual void activemq::core::SimplePriorityMessageDispatchChannel::start ()` [virtual]

Starts dispatch of messages from the Channel.

Implements `activemq::core::MessageDispatchChannel` (p. 1890).

6.486.2.17 `virtual void activemq::core::SimplePriorityMessageDispatchChannel::stop ()` [virtual]

Stops dispatch of message from the Channel.

Implements `activemq::core::MessageDispatchChannel` (p. 1890).

6.486.2.18 `virtual bool activemq::core::SimplePriorityMessageDispatchChannel-
::tryLock () throw (decaf::lang::exceptions::RuntimeException)
[inline, virtual]`

Attempts to Lock the object, if the lock is already held by another thread than this method returns false.

Returns

true if the lock was acquired, false if it is already held by another thread.

Exceptions

<i>RuntimeException</i>	if an error occurs while locking the object.
-------------------------	--

Implements **decaf::util::concurrent::Synchronizable** (p. 2643).

6.486.2.19 `virtual void activemq::core::SimplePriorityMessageDispatchChannel-
::unlock () throw (decaf::lang::exceptions::RuntimeException)
[inline, virtual]`

Unlocks the object.

Exceptions

<i>RuntimeException</i>	if an error occurs while unlocking the object.
-------------------------	--

Implements **decaf::util::concurrent::Synchronizable** (p. 2645).

6.486.2.20 `virtual void activemq::core::SimplePriorityMessageDispatchChannel-
::wait () throw (decaf::lang::exceptions::RuntimeException,
decaf::lang::exceptions::IllegalMonitorStateException,
decaf::lang::exceptions::InterruptedException) [inline,
virtual]`

Waits on a signal from this object, which is generated by a call to Notify.

Must have this object locked before calling.

Exceptions

<i>RuntimeException</i>	if an error occurs while waiting on the object.
<i>InterruptedException</i>	if the wait is interrupted before it completes.
<i>IllegalMonitorState- Exception</i>	- if the current thread is not the owner of the the Synchronizable Object.

Implements **decaf::util::concurrent::Synchronizable** (p. 2646).

6.486 activemq::core::SimplePriorityMessageDispatchChannel Class Reference

6.486.2.21 `virtual void activemq::core::SimplePriorityMessageDispatchChannel::wait (long long millisecs) throw (decaf::lang::exceptions::RuntimeException, decaf::lang::exceptions::IllegalMonitorStateException, decaf::lang::exceptions::InterruptedException) [inline, virtual]`

Waits on a signal from this object, which is generated by a call to Notify.

Must have this object locked before calling. This wait will timeout after the specified time interval.

Parameters

<i>millisecs</i>	the time in milliseconds to wait, or WAIT_INFINITE
------------------	--

Exceptions

<i>RuntimeException</i>	if an error occurs while waiting on the object.
<i>InterruptedException</i>	if the wait is interrupted before it completes.
<i>IllegalMonitorStateException</i>	- if the current thread is not the owner of the the Synchronizable Object.

Implements **decaf::util::concurrent::Synchronizable** (p. 2647).

6.486.2.22 `virtual void activemq::core::SimplePriorityMessageDispatchChannel::wait (long long millisecs, int nanos) throw (decaf::lang::exceptions::RuntimeException, decaf::lang::exceptions::IllegalArgumentException, decaf::lang::exceptions::IllegalMonitorStateException, decaf::lang::exceptions::InterruptedException) [inline, virtual]`

Waits on a signal from this object, which is generated by a call to Notify.

Must have this object locked before calling. This wait will timeout after the specified time interval. This method is similar to the one argument wait function except that it add a finer grained control over the amount of time that it waits by adding in the additional nanosecond argument.

NOTE: The ability to wait accurately at a nanosecond scale depends on the platform and OS that the Decaf API is running on, some systems do not provide an accurate enough clock to provide this level of granularity.

Parameters

<i>millisecs</i>	the time in milliseconds to wait, or WAIT_INFINITE
<i>nanos</i>	additional time in nanoseconds with a range of 0-999999

Exceptions

<i>IllegalArgument-Exception</i>	if an error occurs or the nanos argument is not in the range of [0-999999]
<i>RuntimeException</i>	if an error occurs while waiting on the object.
<i>InterruptedException</i>	if the wait is interrupted before it completes.
<i>IllegalMonitorState-Exception</i>	- if the current thread is not the owner of the the Synchronizable Object.

Implements **decaf::util::concurrent::Synchronizable** (p. 2648).

The documentation for this class was generated from the following file:

- src/main/activemq/core/**SimplePriorityMessageDispatchChannel.h**

6.487 decaf::net::Socket Class Reference

```
#include <src/main/decaf/net/Socket.h>
```

Inheritance diagram for decaf::net::Socket:

Public Member Functions

- **Socket ()**
*Creates an unconnected **Socket** (p. 2452) using the set **SocketImplFactory** (p. 2487) or if non is set than the default **SocketImpl** type is created.*
- **Socket (SocketImpl *impl)**
*Creates a **Socket** (p. 2452) wrapping the provided **SocketImpl** (p. 2479) instance, this **Socket** (p. 2452) is considered unconnected.*
- **Socket (const InetAddress *address, int port)**
*Creates a new **Socket** (p. 2452) instance and connects it to the given address and port.*
- **Socket (const InetAddress *address, int port, const InetAddress *localAddress, int localPort)**
*Creates a new **Socket** (p. 2452) instance and connects it to the given address and port.*
- **Socket (const std::string &host, int port)**
*Creates a new **Socket** (p. 2452) instance and connects it to the given host and port.*
- **Socket (const std::string &host, int port, const InetAddress *localAddress, int localPort)**
*Creates a new **Socket** (p. 2452) instance and connects it to the given host and port.*
- virtual **~Socket ()**
- virtual void **bind** (const std::string &ipaddress, int port)
*Binds this **Socket** (p. 2452) to the given local address and port.*

- virtual void **close** ()
*Closes the **Socket** (p. 2452).*
- virtual void **connect** (const std::string &host, int port)
Connects to the specified destination.
- virtual void **connect** (const std::string &host, int port, int timeout)
Connects to the specified destination, with a specified timeout value.
- bool **isConnected** () const
Indicates whether or not this socket is connected to an end point.
- bool **isClosed** () const
- bool **isBound** () const
- bool **isInputShutdown** () const
- bool **isOutputShutdown** () const
- virtual **decaf::io::InputStream** * **getInputStream** ()
Gets the InputStream for this socket if its connected.
- virtual **decaf::io::OutputStream** * **getOutputStream** ()
Gets the OutputStream for this socket if it is connected.
- int **getPort** () const
*Gets the on the remote host this **Socket** (p. 2452) is connected to.*
- int **getLocalPort** () const
Gets the local port the socket is bound to.
- std::string **getInetAddress** () const
Returns the address to which the socket is connected.
- std::string **getLocalAddress** () const
Gets the local address to which the socket is bound.
- virtual void **shutdownInput** ()
Shuts down the InputStream for this socket essentially marking it as EOF.
- virtual void **shutdownOutput** ()
Shuts down the OutputStream for this socket, any data already written to the socket will be sent, any further calls to OutputStream::write will throw an IOException.
- virtual int **getSoLinger** () const
Gets the linger time for the socket, SO_LINGER.
- virtual void **setSoLinger** (bool state, int timeout)
Sets the linger time (SO_LINGER) using a specified time value, this limits of this value are platform specific.
- virtual bool **getKeepAlive** () const
Gets the keep alive flag for this socket, SO_KEEPALIVE.
- virtual void **setKeepAlive** (bool keepAlive)
Enables/disables the keep alive flag for this socket, SO_KEEPALIVE.
- virtual int **getReceiveBufferSize** () const
Gets the receive buffer size for this socket, SO_RCVBUF.
- virtual void **setReceiveBufferSize** (int size)
Sets the receive buffer size for this socket, SO_RCVBUF.
- virtual bool **getReuseAddress** () const
Gets the reuse address flag, SO_REUSEADDR.

- virtual void **setReuseAddress** (bool reuse)
Sets the reuse address flag, SO_REUSEADDR.
- virtual int **getSendBufferSize** () const
Gets the send buffer size for this socket, SO_SNDBUF, this value is used by the platform socket to buffer data written to the socket.
- virtual void **setSendBufferSize** (int size)
Gets the send buffer size for this socket, SO_SNDBUF, this value is used by the platform socket to buffer data written to the socket.
- virtual int **getSoTimeout** () const
Gets the timeout for socket operations, SO_TIMEOUT.
- virtual void **setSoTimeout** (int timeout)
Sets the timeout for socket operations, SO_TIMEOUT.
- virtual bool **getTcpNoDelay** () const
Gets the Status of the TCP_NODELAY setting for this socket.
- virtual void **setTcpNoDelay** (bool value)
*Sets the Status of the TCP_NODELAY param for this socket., this setting is used to disable or enable Nagle's algorithm on the **Socket** (p. 2452).*
- virtual int **getTrafficClass** () const
*Gets the Traffic Class setting for this **Socket** (p. 2452), sometimes referred to as Type of Service setting.*
- virtual void **setTrafficClass** (int value)
*Gets the Traffic Class setting for this **Socket** (p. 2452), sometimes referred to as Type of Service setting.*
- virtual bool **getOOBInline** () const
Gets the value of the OOBINLINE for this socket.
- virtual void **setOOBInline** (bool value)
Sets the value of the OOBINLINE for this socket, by default this option is disabled.
- virtual void **sendUrgentData** (int data)
*Sends on byte of urgent data to the **Socket** (p. 2452).*
- virtual std::string **toString** () const

Static Public Member Functions

- static void **setSocketImplFactory** (SocketImplFactory *factory)
*Sets the instance of a **SocketImplFactory** (p. 2487) that the **Socket** (p. 2452) class should use when new instances of this class are created.*

Protected Member Functions

- void **accepted** ()
- void **initSocketImpl** (const std::string &address, int port, const InetAddress *localAddress, int localPort)
- void **checkClosed** () const
- void **ensureCreated** () const

Protected Attributes

- **SocketImpl** * *impl*

Friends

- class **ServerSocket**

6.487.1 Detailed Description

Since

1.0

6.487.2 Constructor & Destructor Documentation

6.487.2.1 decaf::net::Socket::Socket ()

Creates an unconnected **Socket** (p. 2452) using the set **SocketImplFactory** (p. 2487) or if non is set than the default **SocketImpl** type is created.

6.487.2.2 decaf::net::Socket::Socket (**SocketImpl** * *impl*)

Creates a **Socket** (p. 2452) wrapping the provided **SocketImpl** (p. 2479) instance, this **Socket** (p. 2452) is considered unconnected.

The **Socket** (p. 2452) class takes ownership of this **SocketImpl** (p. 2479) pointer and will delete it when the **Socket** (p. 2452) class is destroyed.

Parameters

<i>impl</i>	The SocketImpl (p. 2479) instance to wrap.
-------------	---

Exceptions

<i>NullPointerException</i>	if the passed SocketImpl (p. 2479) is Null.
-----------------------------	--

6.487.2.3 decaf::net::Socket::Socket (const **InetAddress** * *address*, int *port*)

Creates a new **Socket** (p. 2452) instance and connects it to the given address and port.

If there is a **SocketImplFactory** (p. 2487) set then the **SocketImpl** is created using the factory otherwise the default **Socket** (p. 2452) implementation is used.

If the host parameter is empty then the loop back address is used.

Parameters

<i>address</i>	The address to connect to.
<i>port</i>	The port number to connect to [0...65535]

Exceptions

UnknownHostException (p. 2816)	if the host cannot be resolved.
<i>IOException</i>	if an I/O error occurs while connecting the Socket (p. 2452).
<i>NullPointerException</i>	if the InetAddress (p. 1437) instance is NULL.
<i>IllegalArgumentException</i>	if the port is not in range [0...65535]

6.487.2.4 `decaf::net::Socket::Socket (const InetAddress * address, int port, const InetAddress * localAddress, int localPort)`

Creates a new **Socket** (p. 2452) instance and connects it to the given address and port.

If there is a **SocketImplFactory** (p. 2487) set then the **SocketImpl** is created using the factory otherwise the default **Socket** (p. 2452) implementation is used. The **Socket** (p. 2452) will also bind to the local address and port specified.

Parameters

<i>address</i>	The address to connect to.
<i>port</i>	The port number to connect to [0...65535]
<i>localAddress</i>	The IP address on the local machine to bind to.
<i>localPort</i>	The port on the local machine to bind to.

Exceptions

UnknownHostException (p. 2816)	if the host cannot be resolved.
<i>IOException</i>	if an I/O error occurs while connecting the Socket (p. 2452).
<i>NullPointerException</i>	if the InetAddress (p. 1437) instance is NULL.
<i>IllegalArgumentException</i>	if the port is not in range [0...65535]

6.487.2.5 `decaf::net::Socket::Socket (const std::string & host, int port)`

Creates a new **Socket** (p. 2452) instance and connects it to the given host and port.

If there is a **SocketImplFactory** (p. 2487) set then the **SocketImpl** is created using the factory otherwise the default **Socket** (p. 2452) implementation is used.

If the host parameter is empty then the loop back address is used.

Parameters

<i>host</i>	The host name or IP address to connect to, empty string means loop-back.
<i>port</i>	The port number to connect to [0...65535]

Exceptions

UnknownHostException (p. 2816)	if the host cannot be resolved.
<i>IOException</i>	if an I/O error occurs while connecting the Socket (p. 2452).
<i>IllegalArgumentException</i>	if the port if not in range [0...65535]

6.487.2.6 decaf::net::Socket::Socket (const std::string & *host*, int *port*, const InetAddress * *localAddress*, int *localPort*)

Creates a new **Socket** (p. 2452) instance and connects it to the given host and port.

If there is a **SocketImplFactory** (p. 2487) set then the SokcetImpl is created using the factory otherwise the default **Socket** (p. 2452) implementation is used.

If the host parameter is empty then the loop back address is used.

Parameters

<i>host</i>	The host name or IP address to connect to, empty string means loop-back.
<i>port</i>	The port number to connect to [0...65535]
<i>localAddress</i>	The IP address on the local machine to bind to.
<i>localPort</i>	The port on the local machine to bind to.

Exceptions

UnknownHostException (p. 2816)	if the host cannot be resolved.
<i>IOException</i>	if an I/O error occurs while connecting the Socket (p. 2452).
<i>IllegalArgumentException</i>	if the port if not in range [0...65535]

6.487.2.7 virtual decaf::net::Socket::~Socket () [virtual]

6.487.3 Member Function Documentation

6.487.3.1 `void decaf::net::Socket::accepted ()` [protected]

6.487.3.2 `virtual void decaf::net::Socket::bind (const std::string & ipaddress, int port)`
[virtual]

Binds this **Socket** (p. 2452) to the given local address and port.

If the **SocketAddress** (p. 2470) value is NULL then the **Socket** (p. 2452) will be bound to an available local address and port.

Parameters

<i>ipaddress</i>	The local address and port to bind the socket to.
<i>port</i>	The port on the local machine to bind to.

Exceptions

<i>IOException</i>	if an error occurs during the bind operation.
<i>IllegalArgument-Exception</i>	if the Socket (p.2452) can't process the subclass of Socket-Address (p. 2470) that has been provided.

6.487.3.3 `void decaf::net::Socket::checkClosed () const` [protected]

6.487.3.4 `virtual void decaf::net::Socket::close ()` [virtual]

Closes the **Socket** (p. 2452).

Once closed a **Socket** (p. 2452) cannot be connected or otherwise operated upon, a new **Socket** (p. 2452) instance must be created.

Exceptions

<i>IOException</i>	if an I/O error occurs while closing the Socket (p. 2452).
--------------------	---

Implements **decaf::io::Closeable** (p. 817).

Reimplemented in **decaf::internal::net::ssl::openssl::OpenSSLSocket** (p. 2020).

6.487.3.5 `virtual void decaf::net::Socket::connect (const std::string & host, int port)`
[virtual]

Connects to the specified destination.

Parameters

<i>host</i>	The host name or IP address of the remote host to connect to.
<i>port</i>	The port on the remote host to connect to.

Exceptions

<i>IOException</i>	Thrown if a failure occurred in the connect.
<i>IllegalArgument-Exception</i>	if the timeout value is negative or the endpoint is invalid.

6.487.3.6 `virtual void decaf::net::Socket::connect (const std::string & host, int port, int timeout) [virtual]`

Connects to the specified destination, with a specified timeout value.

If a connection to the remote host is not established within the specified timeout interval than an **SocketTimeoutException** (p. 2493) is thrown. A timeout value of zero is treated as an infinite timeout.

Parameters

<i>host</i>	The host name or IP address of the remote host to connect to.
<i>port</i>	The port on the remote host to connect to.
<i>timeout</i>	The number of Milliseconds to wait before treating the connection as failed.

Exceptions

<i>IOException</i>	Thrown if a failure occurred in the connect.
SocketTimeout-Exception (p. 2493)	if the timeout for connection is exceeded.
<i>IllegalArgument-Exception</i>	if the timeout value is negative or the endpoint is invalid.

Reimplemented in **decaf::internal::net::ssl::openssl::OpenSSLSocket** (p. 2020).

6.487.3.7 `void decaf::net::Socket::ensureCreated () const [protected]`

6.487.3.8 `std::string decaf::net::Socket::getInetAddress () const`

Returns the address to which the socket is connected.

Returns

the remote IP address to which this socket is connected, or null if the socket is not connected.

6.487.3.9 `virtual decaf::io::InputStream* decaf::net::Socket::getInputStream () [virtual]`

Gets the InputStream for this socket if its connected.

The pointer returned is the property of the associated **Socket** (p. 2452) and should not be deleted by the caller.

When the returned `InputStream` is performing a blocking operation and the underlying connection is closed or otherwise broken the read calls will normally throw an exception to indicate the failure.

Closing the `InputStream` will also close the underlying **Socket** (p. 2452).

Returns

The `InputStream` for this socket.

Exceptions

<i>IOException</i>	if an error occurs during creation of the <code>InputStream</code> , also if the - Socket (p. 2452) is not connected or the input has been shutdown previously.
---------------------------	--

Reimplemented in **decaf::internal::net::ssl::openssl::OpenSSLSocket** (p. 2022).

6.487.3.10 `virtual bool decaf::net::Socket::getKeepAlive () const` [virtual]

Gets the keep alive flag for this socket, `SO_KEEPALIVE`.

Returns

true if keep alive is enabled for this socket.

Exceptions

<i>SocketException</i> (p. 2471)	if the operation fails.
--	-------------------------

6.487.3.11 `std::string decaf::net::Socket::getLocalAddress () const`

Gets the local address to which the socket is bound.

Returns

the local address to which the socket is bound or `InetAddress.anyLocalAddress()` if the socket is not bound yet.

6.487.3.12 `int decaf::net::Socket::getLocalPort () const`

Gets the local port the socket is bound to.

Returns

the local port the socket was bound to, or -1 if the socket is not bound.

6.487.3.13 `virtual bool decaf::net::Socket::getOOBInline () const` [virtual]

Gets the value of the OOBINLINE for this socket.

Returns

true if OOBINLINE is enabled, false otherwise.

Exceptions

<i>SocketException</i> (p. 2471)	if an error is encountered while performing this operation.
--	---

6.487.3.14 `virtual decaf::io::OutputStream* decaf::net::Socket::getOutputStream ()` [virtual]

Gets the OutputStream for this socket if it is connected.

The pointer returned is the property of the **Socket** (p. 2452) instance and should not be deleted by the caller.

Closing the returned **Socket** (p. 2452) will also close the underlying **Socket** (p. 2452).

Returns

the OutputStream for this socket.

Exceptions

<i>IOException</i>	if an error occurs during the creation of this OutputStream, or if the Socket (p. 2452) is closed or the output has been shutdown previously.
---------------------------	--

Reimplemented in **decaf::internal::net::ssl::openssl::OpenSSLSocket** (p. 2022).

6.487.3.15 `int decaf::net::Socket::getPort () const`

Gets the on the remote host this **Socket** (p. 2452) is connected to.

Returns

the port on the remote host the socket is connected to, or 0 if not connected.

6.487.3.16 `virtual int decaf::net::Socket::getReceiveBufferSize () const` [virtual]

Gets the receive buffer size for this socket, SO_RCVBUF.

This is the buffer used by the underlying platform socket to buffer received data.

Returns

the receive buffer size in bytes.

Exceptions

SocketException (p. 2471)	if the operation fails.
-------------------------------------	-------------------------

6.487.3.17 `virtual bool decaf::net::Socket::getReuseAddress () const` [virtual]

Gets the reuse address flag, SO_REUSEADDR.

Returns

True if the address can be reused.

Exceptions

SocketException (p. 2471)	if the operation fails.
-------------------------------------	-------------------------

6.487.3.18 `virtual int decaf::net::Socket::getSendBufferSize () const` [virtual]

Gets the send buffer size for this socket, SO_SNDBUF, this value is used by the platform socket to buffer data written to the socket.

Returns

the size in bytes of the send buffer.

Exceptions

SocketException (p. 2471)	if the operation fails.
-------------------------------------	-------------------------

6.487.3.19 `virtual int decaf::net::Socket::getSoLinger () const` [virtual]

Gets the linger time for the socket, SO_LINGER.

A return value of -1 indicates that the option is disabled.

Returns

The linger time in seconds.

Exceptions

SocketException (p. 2471)	if the operation fails.
-------------------------------------	-------------------------

6.487.3.20 `virtual int decaf::net::Socket::getSoTimeout () const` [virtual]

Gets the timeout for socket operations, SO_TIMEOUT.

Returns

The timeout in milliseconds for socket operations.

Exceptions

SocketException (p. 2471)	Thrown if unable to retrieve the information.
-------------------------------------	---

6.487.3.21 `virtual bool decaf::net::Socket::getTcpNoDelay () const` [virtual]

Gets the Status of the TCP_NODELAY setting for this socket.

Returns

true if TCP_NODELAY is enabled for the socket.

Exceptions

SocketException (p. 2471)	Thrown if unable to set the information.
-------------------------------------	--

6.487.3.22 `virtual int decaf::net::Socket::getTrafficClass () const` [virtual]

Gets the Traffic Class setting for this **Socket** (p. 2452), sometimes referred to as Type of Service setting.

This setting is dependent on the underlying network implementation for the platform this **Socket** (p. 2452) runs on and is not guaranteed to have any effect.

Refer to your platforms network documentation regarding support for this setting.

Returns

the bitset result of querying the traffic class setting.

Exceptions

SocketException (p. 2471)	if an error is encountered while performing this operation.
-------------------------------------	---

6.487.3.23 `void decaf::net::Socket::initSocketImpl (const std::string & address, int port, const InetAddress * localAddress, int localPort)` [protected]

6.487.3.24 `bool decaf::net::Socket::isBound () const` [inline]

Returns

true if this **Socket** (p. 2452) has been bound to a Local address.

6.487.3.25 `bool decaf::net::Socket::isClosed () const` [inline]

Returns

true if the **Socket** (p. 2452) has been closed.

6.487.3.26 `bool decaf::net::Socket::isConnected () const` [inline]

Indicates whether or not this socket is connected to an end point.

Returns

true if connected, false otherwise.

6.487.3.27 `bool decaf::net::Socket::isInputShutdown () const` [inline]

Returns

true if input on this **Socket** (p. 2452) has been shutdown.

6.487.3.28 `bool decaf::net::Socket::isOutputShutdown () const [inline]`

Returns

true if output on this **Socket** (p. 2452) has been shutdown.

6.487.3.29 `virtual void decaf::net::Socket::sendUrgentData (int data) [virtual]`

Sends on byte of urgent data to the **Socket** (p. 2452).

Parameters

<i>data</i>	The value to write as urgent data, only the lower eight bits are sent.
-------------	--

Exceptions

<i>IOException</i>	if an I/O error occurs while performing this operation.
--------------------	---

Reimplemented in **decaf::internal::net::ssl::openssl::OpenSSLSocket** (p. 2024).

6.487.3.30 `virtual void decaf::net::Socket::setKeepAlive (bool keepAlive) [virtual]`

Enables/disables the keep alive flag for this socket, SO_KEEPALIVE.

Parameters

<i>keepAlive</i>	If true, enables the flag.
------------------	----------------------------

Exceptions

SocketException (p. 2471)	if the operation fails.
-------------------------------------	-------------------------

6.487.3.31 `virtual void decaf::net::Socket::setOOBInline (bool value) [virtual]`

Sets the value of the OOBINLINE for this socket, by default this option is disabled.

If enabled the urgent data is read inline on the **Socket** (p. 2452)'s InputStream, no notification is give.

Returns

true if OOBINLINE is enabled, false otherwise.

Exceptions

<i>SocketException</i> (p. 2471)	if an error is encountered while performing this operation.
--	---

Reimplemented in **decaf::internal::net::ssl::openssl::OpenSSLSocket** (p. 2026).

6.487.3.32 **virtual void decaf::net::Socket::setReceiveBufferSize (int *size*)**
[virtual]

Sets the receive buffer size for this socket, SO_RCVBUF.

Parameters

<i>size</i>	Number of bytes to set the receive buffer to.
-------------	---

Exceptions

<i>SocketException</i> (p. 2471)	if the operation fails.
<i>IllegalArgument-Exception</i>	if the value is zero or negative.

6.487.3.33 **virtual void decaf::net::Socket::setReuseAddress (bool *reuse*)**
[virtual]

Sets the reuse address flag, SO_REUSEADDR.

Parameters

<i>reuse</i>	If true, sets the flag.
--------------	-------------------------

Exceptions

<i>SocketException</i> (p. 2471)	if the operation fails.
--	-------------------------

6.487.3.34 **virtual void decaf::net::Socket::setSendBufferSize (int *size*)**
[virtual]

Gets the send buffer size for this socket, SO_SNDBUF, this value is used by the platform socket to buffer data written to the socket.

Parameters

<i>size</i>	The number of bytes to set the send buffer to, must be larger than zero.
-------------	--

Exceptions

SocketException (p. 2471)	if the operation fails.
<i>IllegalArgument-Exception</i>	if the value is zero or negative.

6.487.3.35 static void decaf::net::Socket::setSocketImplFactory (SocketImplFactory * factory) [static]

Sets the instance of a **SocketImplFactory** (p. 2487) that the **Socket** (p. 2452) class should use when new instances of this class are created.

This method is only allowed to be used once during the lifetime of the application.

Parameters

<i>factory</i>	The instance of a SocketImplFactory (p. 2487) to use when new - Socket (p. 2452) objects are created.
----------------	---

Exceptions

<i>IOException</i>	if an I/O error occurs while performing this operation.
SocketException (p. 2471)	if this method has already been called with a valid factory.

6.487.3.36 virtual void decaf::net::Socket::setSoLinger (bool state, int timeout) [virtual]

Sets the linger time (SO_LINGER) using a specified time value, this limits of this value are platform specific.

Parameters

<i>state</i>	The state of SO_LINGER, true is on.
<i>timeout</i>	The linger time in seconds, must be non-negative.

Exceptions

SocketException (p. 2471)	if the operation fails.
<i>IllegalArgument-Exception</i>	if state is true and timeout is negative.

6.487.3.37 virtual void **decaf::net::Socket::setSoTimeout** (int *timeout*) [virtual]

Sets the timeout for socket operations, SO_TIMEOUT.

A value of zero indicates that timeout is infinite for operations on this socket.

Parameters

<i>timeout</i>	The timeout in milliseconds for socket operations.
----------------	--

Exceptions

SocketException (p. 2471)	Thrown if unable to set the information.
IllegalArgument-Exception	if the timeout value is negative.

6.487.3.38 virtual void **decaf::net::Socket::setTcpNoDelay** (bool *value*)
[virtual]

Sets the Status of the TCP_NODELAY param for this socket., this setting is used to disable or enable Nagle's algorithm on the **Socket** (p. 2452).

Parameters

<i>value</i>	The setting for the socket's TCP_NODELAY option, true to enable.
--------------	--

Exceptions

SocketException (p. 2471)	Thrown if unable to set the information.
-------------------------------------	--

6.487.3.39 virtual void **decaf::net::Socket::setTrafficClass** (int *value*) [virtual]

Gets the Traffic Class setting for this **Socket** (p. 2452), sometimes referred to as Type of Service setting.

This setting is dependent on the underlying network implementation for the platform this **Socket** (p. 2452) runs on and is not guaranteed to have any effect.

Refer to your platforms network documentation regarding support for this setting.

Parameters

<i>value</i>	The integer value representing the traffic class setting bitset.
--------------	--

Exceptions

<i>SocketException</i> (p. 2471)	if an error is encountered while performing this operation.
<i>IllegalArgument-Exception</i>	if the value is not in the range [0..255].

6.487.3.40 virtual void **decaf::net::Socket::shutdownInput** () [virtual]

Shuts down the InputStream for this socket essentially marking it as EOF.

The stream returns EOF for any calls to read after this method has been called.

Exceptions

<i>IOException</i>	if an I/O error occurs while performing this operation.
---------------------------	---

Reimplemented in **decaf::internal::net::ssl::openssl::OpenSSLSocket** (p. 2027).

6.487.3.41 virtual void **decaf::net::Socket::shutdownOutput** () [virtual]

Shuts down the OutputStream for this socket, any data already written to the socket will be sent, any further calls to OutputStream::write will throw an IOException.

Exceptions

<i>IOException</i>	if an I/O error occurs while performing this operation.
---------------------------	---

Reimplemented in **decaf::internal::net::ssl::openssl::OpenSSLSocket** (p. 2027).

6.487.3.42 virtual std::string **decaf::net::Socket::toString** () const [virtual]

Returns

a string representing this **Socket** (p. 2452).

6.487.4 Friends And Related Function Documentation

6.487.4.1 friend class **ServerSocket** [friend]

6.487.5 Field Documentation

6.487.5.1 **SocketImpl*** **decaf::net::Socket::impl** [mutable, protected]

The documentation for this class was generated from the following file:

- src/main/decaf/net/**Socket.h**

6.488 decaf::net::SocketAddress Class Reference

Base class for protocol specific **Socket** (p. 2452) addresses.

```
#include <src/main/decaf/net/SocketAddress.h>
```

Inheritance diagram for decaf::net::SocketAddress:

Public Member Functions

- virtual **~SocketAddress** ()

6.488.1 Detailed Description

Base class for protocol specific **Socket** (p. 2452) addresses.

These classes provide an immutable address object that is used by the **Socket** (p. 2452) classes.

Since

1.0

6.488.2 Constructor & Destructor Documentation

6.488.2.1 virtual **decaf::net::SocketAddress::~SocketAddress** () [inline, virtual]

The documentation for this class was generated from the following file:

- src/main/decaf/net/**SocketAddress.h**

6.489 decaf::net::SocketError Class Reference

Static utility class to simplify handling of error codes for socket operations.

```
#include <src/main/decaf/net/SocketError.h>
```

Static Public Member Functions

- static int **getErrorCode** ()
Gets the last error appropriate for the platform.
- static std::string **getErrorString** ()
Gets the string description for the last error.

6.489.1 Detailed Description

Static utility class to simplify handling of error codes for socket operations.

6.489.2 Member Function Documentation

6.489.2.1 static int decaf::net::SocketError::getErrorCode () [static]

Gets the last error appropriate for the platform.

6.489.2.2 static std::string decaf::net::SocketError::getErrorString () [static]

Gets the string description for the last error.

The documentation for this class was generated from the following file:

- src/main/decaf/net/**SocketError.h**

6.490 decaf::net::SocketException Class Reference

Exception for errors when manipulating sockets.

```
#include <src/main/decaf/net/SocketException.h>
```

Inheritance diagram for decaf::net::SocketException:

Public Member Functions

- **SocketException** () throw ()
- **SocketException** (const lang::Exception &ex) throw ()
- **SocketException** (const SocketException &ex) throw ()
- **SocketException** (const char *file, const int lineNumber, const std::exception *cause, const char *msg,...) throw ()
Constructor - Initializes the file name and line number where this message occurred.
- **SocketException** (const std::exception *cause) throw ()
Constructor.
- **SocketException** (const char *file, const int lineNumber, const char *msg,...) throw ()
Constructor - Initializes the file name and line number where this message occurred.
- virtual **SocketException** * clone () const
Clones this exception.
- virtual ~**SocketException** () throw ()

6.490.1 Detailed Description

Exception for errors when manipulating sockets.

6.490.2 Constructor & Destructor Documentation

6.490.2.1 `decaf::net::SocketException::SocketException () throw () [inline]`

6.490.2.2 `decaf::net::SocketException::SocketException (const lang::Exception & ex) throw () [inline]`

6.490.2.3 `decaf::net::SocketException::SocketException (const SocketException & ex) throw () [inline]`

6.490.2.4 `decaf::net::SocketException::SocketException (const char * file, const int lineNumber, const std::exception * cause, const char * msg, ...) throw () [inline]`

Constructor - Initializes the file name and line number where this message occurred.

Sets the message to report, using an optional list of arguments to parse into the message

Parameters

<i>file</i>	The file name where exception occurs
<i>lineNumber</i>	The line number where the exception occurred.
<i>cause</i>	The exception that was the cause for this one to be thrown.
<i>msg</i>	The message to report
<i>...</i>	list of primitives that are formatted into the message

6.490.2.5 `decaf::net::SocketException::SocketException (const std::exception * cause) throw () [inline]`

Constructor.

Parameters

<i>cause</i>	Pointer to the exception that caused this one to be thrown, the object is cloned caller retains ownership.
--------------	--

6.490.2.6 `decaf::net::SocketException::SocketException (const char * file, const int lineNumber, const char * msg, ...) throw () [inline]`

Constructor - Initializes the file name and line number where this message occurred.

Sets the message to report, using an optional list of arguments to parse into the mes-

sage

Parameters

<i>file</i>	The file name where exception occurs
<i>lineNumber</i>	The line number where the exception occurred.
<i>msg</i>	The message to report
...	list of primitives that are formatted into the message

6.490.2.7 `virtual decaf::net::SocketException::~~SocketException () throw ()`
`[inline, virtual]`

6.490.3 Member Function Documentation

6.490.3.1 `virtual SocketException* decaf::net::SocketException::clone () const`
`[inline, virtual]`

Clones this exception.

This is useful for cases where you need to preserve the type of the original exception as well as the message. All subclasses should override.

Returns

a new Exception instance that is a copy of this Exception object.

Reimplemented from **decaf::io::IOException** (p. 1547).

Reimplemented in **decaf::internal::net::ssl::openssl::OpenSSLSocketException** (p. 2032), **decaf::net::BindException** (p. 534), **decaf::net::ConnectException** (p. 933), **decaf::net::NoRouteToHostException** (p. 1981), and **decaf::net::PortUnreachableException** (p. 2109).

The documentation for this class was generated from the following file:

- `src/main/decaf/net/SocketException.h`

6.491 decaf::net::SocketFactory Class Reference

The **SocketFactory** (p. 2473) is used to create **Socket** (p. 2452) objects and can be sub-classed to provide other types of Sockets or Sockets with varying configurations.

```
#include <src/main/decaf/net/SocketFactory.h>
```

Inheritance diagram for `decaf::net::SocketFactory`:

Public Member Functions

- virtual **~SocketFactory** ()
- virtual **Socket * createSocket** ()
*Creates an unconnected **Socket** (p. 2452) object.*
- virtual **Socket * createSocket** (const **InetAddress** *host, int port)=0
*Creates a new **Socket** (p. 2452) object and connects it to the specified remote host and port using the configuration of this **SocketFactory** (p. 2473).*
- virtual **Socket * createSocket** (const **InetAddress** *host, int port, const **InetAddress** *ifAddress, int localPort)=0
*Creates a new **Socket** (p. 2452) object and connects it to the specified remote host and port using the configuration of this **SocketFactory** (p. 2473).*
- virtual **Socket * createSocket** (const std::string &name, int port)=0
*Creates a new **Socket** (p. 2452) object and connects it to the specified remote host and port using the configuration of this **SocketFactory** (p. 2473).*
- virtual **Socket * createSocket** (const std::string &name, int port, const **InetAddress** *ifAddress, int localPort)=0
*Creates a new **Socket** (p. 2452) object and connects it to the specified remote host and port using the configuration of this **SocketFactory** (p. 2473).*

Static Public Member Functions

- static **SocketFactory * getDefault** ()
*Returns an pointer to the default **SocketFactory** (p. 2473) for this Application, there is only one default **SocketFactory** (p. 2473) per application, the pointer returned by this method is owned by the **SocketFactory** (p. 2473) class and in not to be deleted by the caller.*

Protected Member Functions

- **SocketFactory** ()

6.491.1 Detailed Description

The **SocketFactory** (p. 2473) is used to create **Socket** (p. 2452) objects and can be sub-classed to provide other types of Sockets or Sockets with varying configurations.

See also

decaf.net.Socket (p. 2452)

Since

1.0

6.491.2 Constructor & Destructor Documentation

6.491.2.1 `decaf::net::SocketFactory::SocketFactory ()` [protected]

6.491.2.2 `virtual decaf::net::SocketFactory::~~SocketFactory ()` [virtual]

6.491.3 Member Function Documentation

6.491.3.1 `virtual Socket* decaf::net::SocketFactory::createSocket ()`
[virtual]

Creates an unconnected **Socket** (p. 2452) object.

Returns

a new **Socket** (p. 2452) object, caller must free this object when done.

Exceptions

<i>IOException</i>	if the Socket (p. 2452) cannot be created.
--------------------	---

Reimplemented in **decaf::internal::net::ssl::openssl::OpenSSLSocketFactory** (p. 2036), **decaf::internal::net::ssl::DefaultSSLSocketFactory** (p. 1173), and **decaf::internal::net::DefaultSocketFactory** (p. 1161).

6.491.3.2 `virtual Socket* decaf::net::SocketFactory::createSocket (const InetAddress * host, int port)` [pure virtual]

Creates a new **Socket** (p. 2452) object and connects it to the specified remote host and port using the configuration of this **SocketFactory** (p. 2473).

Parameters

<i>host</i>	The host to connect the socket to.
<i>port</i>	The port on the remote host to connect to.

Returns

a new **Socket** (p. 2452) object, caller must free this object when done.

Exceptions

<i>IOException</i>	if an I/O error occurs while creating the Socket (p. 2452) object.
UnknownHostException (p. 2816)	if the host name is not known.

Implemented in **decaf::internal::net::ssl::openssl::OpenSSLSocketFactory** (p. 2036), **decaf::internal::net::ssl::DefaultSSLSocketFactory** (p. 1174), and **decaf::internal::net::DefaultSocketFactory** (p. 1161).

6.491.3.3 **virtual Socket* decaf::net::SocketFactory::createSocket (const InetAddress * host, int port, const InetAddress * ifAddress, int localPort)**
[pure virtual]

Creates a new **Socket** (p. 2452) object and connects it to the specified remote host and port using the configuration of this **SocketFactory** (p. 2473).

The **Socket** (p. 2452) will be bound to the specified local address and port.

Parameters

<i>host</i>	The host to connect the socket to.
<i>port</i>	The port on the remote host to connect to.
<i>ifAddress</i>	The address on the local machine to bind the Socket (p. 2452) to.
<i>localPort</i>	The local port to bind the Socket (p. 2452) to.

Returns

a new **Socket** (p. 2452) object, caller must free this object when done.

Exceptions

<i>IOException</i>	if an I/O error occurs while creating the Socket (p. 2452) object.
UnknownHostException (p. 2816)	if the host name is not known.

Implemented in **decaf::internal::net::ssl::openssl::OpenSSLSocketFactory** (p. 2037), **decaf::internal::net::ssl::DefaultSSLSocketFactory** (p. 1174), and **decaf::internal::net::DefaultSocketFactory** (p. 1162).

6.491.3.4 **virtual Socket* decaf::net::SocketFactory::createSocket (const std::string & name, int port)** [pure virtual]

Creates a new **Socket** (p. 2452) object and connects it to the specified remote host and port using the configuration of this **SocketFactory** (p. 2473).

Parameters

<i>host</i>	The host name or IP address to connect the socket to.
<i>port</i>	The port on the remote host to connect to.

Returns

a new **Socket** (p. 2452) object, caller must free this object when done.

Exceptions

<i>IOException</i>	if an I/O error occurs while creating the Socket (p. 2452) object.
UnknownHostException (p. 2816)	if the host name is not known.

Implemented in **decaf::internal::net::ssl::openssl::OpenSSLSocketFactory** (p. 2037), **decaf::internal::net::ssl::DefaultSSLSocketFactory** (p. 1175), and **decaf::internal::net::DefaultSocketFactory** (p. 1163).

6.491.3.5 virtual **Socket*** **decaf::net::SocketFactory::createSocket** (const std::string & *name*, int *port*, const InetAddress * *ifAddress*, int *localPort*) [pure virtual]

Creates a new **Socket** (p. 2452) object and connects it to the specified remote host and port using the configuration of this **SocketFactory** (p. 2473).

Parameters

<i>host</i>	The host name or IP address to connect the socket to.
<i>port</i>	The port on the remote host to connect to.
<i>ifAddress</i>	The address on the local machine to bind the Socket (p. 2452) to.
<i>localPort</i>	The local port to bind the Socket (p. 2452) to.

Returns

a new **Socket** (p. 2452) object, caller must free this object when done.

Exceptions

<i>IOException</i>	if an I/O error occurs while creating the Socket (p. 2452) object.
UnknownHostException (p. 2816)	if the host name is not known.

Implemented in **decaf::internal::net::ssl::openssl::OpenSSLSocketFactory** (p. 2038), **decaf::internal::net::ssl::DefaultSSLSocketFactory** (p. 1176), and **decaf::internal::net::DefaultSocketFactory** (p. 1163).

6.491.3.6 **static SocketFactory* decaf::net::SocketFactory::getDefault ()**
 [static]

Returns an pointer to the default **SocketFactory** (p. 2473) for this Application, there is only one default **SocketFactory** (p. 2473) per application, the pointer returned by this method is owned by the **SocketFactory** (p. 2473) class and in not to be deleted by the caller.

Returns

pointer to the applications default **SocketFactory** (p. 2473).

Exceptions

SocketException (p. 2471)	if an error occurs while getting the default instance.
-------------------------------------	--

Reimplemented in **decaf::net::ssl::SSLSocketFactory** (p. 2523).

The documentation for this class was generated from the following file:

- src/main/decaf/net/**SocketFactory.h**

6.492 decaf::internal::net::SocketFileDescriptor Class Reference

File Descriptor type used internally by Decaf Socket objects.

```
#include <src/main/decaf/internal/net/SocketFileDescriptor.h>
```

Inheritance diagram for decaf::internal::net::SocketFileDescriptor:

Public Member Functions

- **SocketFileDescriptor** (long value)
- virtual **~SocketFileDescriptor** ()
- long **getValue** () const
Gets the OS Level FileDescriptor.

6.492.1 Detailed Description

File Descriptor type used internally by Decaf Socket objects.

Since

1.0

6.492.2 Constructor & Destructor Documentation

6.492.2.1 `decaf::internal::net::SocketFileDescriptor::SocketFileDescriptor (long value)`

6.492.2.2 `virtual decaf::internal::net::SocketFileDescriptor::~~SocketFileDescriptor () [virtual]`

6.492.3 Member Function Documentation

6.492.3.1 `long decaf::internal::net::SocketFileDescriptor::getValue () const`

Gets the OS Level FileDescriptor.

Returns

a FileDescriptor value.

The documentation for this class was generated from the following file:

- `src/main/decaf/internal/net/SocketFileDescriptor.h`

6.493 decaf::net::SocketImpl Class Reference

Acts as a base class for all physical **Socket** (p. 2452) implementations.

```
#include <src/main/decaf/net/SocketImpl.h>
```

Inheritance diagram for decaf::net::SocketImpl:

Public Member Functions

- **SocketImpl** ()
- virtual **~SocketImpl** ()
- virtual void **create** ()=0
*Creates the underlying platform **Socket** (p. 2452) data structures which allows for - **Socket** (p. 2452) options to be applied.*
- virtual void **accept** (**SocketImpl** *socket)=0
*Accepts a new connection on the given **Socket** (p. 2452).*
- virtual void **connect** (const std::string &hostname, int **port**, int timeout)=0
Connects this socket to the given host and port.
- virtual void **bind** (const std::string &ipaddress, int **port**)=0
*Binds this **Socket** (p. 2452) instance to the local ip address and port number given.*
- virtual void **listen** (int backlog)=0

Sets the maximum queue length for incoming connection indications (a request to connect) to the count argument.

- virtual **decaf::io::InputStream** * **getInputStream** ()=0
*Gets the InputStream linked to this **Socket** (p. 2452).*
- virtual **decaf::io::OutputStream** * **getOutputStream** ()=0
*Gets the OutputStream linked to this **Socket** (p. 2452).*
- virtual int **available** ()=0
*Gets the number of bytes that can be read from the **Socket** (p. 2452) without blocking.*
- virtual void **close** ()=0
Closes the socket, terminating any blocked reads or writes.
- virtual void **shutdownInput** ()=0
Places the input stream for this socket at "end of stream".
- virtual void **shutdownOutput** ()=0
Disables the output stream for this socket.
- virtual int **getOption** (int option) const =0
*Gets the specified **Socket** (p. 2452) option.*
- virtual void **setOption** (int option, int value)=0
*Sets the specified option on the **Socket** (p. 2452) if supported.*
- int **getPort** () const
Gets the port that this socket has been assigned.
- int **getLocalPort** () const
*Gets the value of this **SocketImpl** (p. 2479)'s local port field.*
- std::string **getInetAddress** () const
*Gets the value of this **SocketImpl** (p. 2479)'s address field.*
- const **decaf::io::FileDescriptor** * **getFileDescriptor** () const
*Gets the FileDescriptor for this **Socket** (p. 2452), the Object is owned by this **Socket** (p. 2452) and should not be deleted by the caller.*
- virtual std::string **getLocalAddress** () const =0
*Gets the value of the local Inet address the **Socket** (p. 2452) is bound to if bound, otherwise return the **InetAddress** (p. 1437) ANY value "0.0.0.0".*
- std::string **toString** () const
*Returns a string containing the address and port of this **Socket** (p. 2452) instance.*
- virtual bool **supportsUrgentData** () const
- virtual void **sendUrgentData** (int data)
*Sends on byte of urgent data to the **Socket** (p. 2452).*

Protected Attributes

- int **port**
*The remote port that this **Socket** (p. 2452) is connected to.*
- int **localPort**
*The port on the Local Machine that this **Socket** (p. 2452) is Bound to.*
- std::string **address**
*The Remote Address that the **Socket** (p. 2452) is connected to.*
- **io::FileDescriptor** * **fd**
*The File Descriptor for this **Socket** (p. 2452).*

6.493.1 Detailed Description

Acts as a base class for all physical **Socket** (p. 2452) implementations.

Since

1.0

6.493.2 Constructor & Destructor Documentation

6.493.2.1 `decaf::net::SocketImpl::SocketImpl ()`

6.493.2.2 `virtual decaf::net::SocketImpl::~~SocketImpl ()` [virtual]

6.493.3 Member Function Documentation

6.493.3.1 `virtual void decaf::net::SocketImpl::accept (SocketImpl * socket)`
[pure virtual]

Accepts a new connection on the given **Socket** (p. 2452).

Parameters

<i>socket</i>	The accepted connection.
---------------	--------------------------

Exceptions

<i>IOException</i>	if an I/O error occurs while attempting this operation.
SocketException (p. 2471)	if an error occurs while performing an Accept on the socket.
SocketTimeout-Exception (p. 2493)	if the accept call times out due to SO_TIMEOUT being set.

6.493.3.2 `virtual int decaf::net::SocketImpl::available ()` [pure virtual]

Gets the number of bytes that can be read from the **Socket** (p. 2452) without blocking.

Returns

the number of bytes that can be read from the **Socket** (p. 2452) without blocking.

Exceptions

<i>IOException</i>	if an I/O error occurs while attempting this operation.
--------------------	---

Implemented in **decaf::internal::net::tcp::TcpSocket** (p. 2682).

6.493.3.3 virtual void **decaf::net::SocketImpl::bind** (const std::string & *ipaddress*, int *port*) [pure virtual]

Binds this **Socket** (p. 2452) instance to the local ip address and port number given.

Parameters

<i>ipaddress</i>	The address of local ip to bind to.
<i>port</i>	The port number on the host to bind to.

Exceptions

<i>IOException</i>	if an I/O error occurs while attempting this operation.
--------------------	---

Implemented in **decaf::internal::net::tcp::TcpSocket** (p. 2682).

6.493.3.4 virtual void **decaf::net::SocketImpl::close** () [pure virtual]

Closes the socket, terminating any blocked reads or writes.

Exceptions

<i>IOException</i>	if an I/O error occurs while attempting this operation.
--------------------	---

Implemented in **decaf::internal::net::tcp::TcpSocket** (p. 2683).

6.493.3.5 virtual void **decaf::net::SocketImpl::connect** (const std::string & *hostname*, int *port*, int *timeout*) [pure virtual]

Connects this socket to the given host and port.

Parameters

<i>hostname</i>	The name of the host to connect to, or IP address.
<i>port</i>	The port number on the host to connect to.
<i>timeout</i>	Time in milliseconds to wait for a connection, 0 indicates forever.

Exceptions

<i>IOException</i>	if an I/O error occurs while attempting this operation.
SocketTimeout-Exception (p. 2493)	if the connect call times out due to timeout being set.
<i>IllegalArgument-Exception</i>	if a parameter has an illegal value.

Implemented in **decaf::internal::net::tcp::TcpSocket** (p. 2683).

6.493.3.6 virtual void **decaf::net::SocketImpl::create** () [pure virtual]

Creates the underlying platform **Socket** (p. 2452) data structures which allows for - **Socket** (p. 2452) options to be applied.

The created socket is in an unconnected state.

Exceptions

<i>IOException</i>	if an I/O error occurs while attempting this operation.
--------------------	---

Implemented in **decaf::internal::net::tcp::TcpSocket** (p. 2683).

6.493.3.7 const **decaf::io::FileDescriptor*** **decaf::net::SocketImpl::getFileDescriptor** () const [inline]

Gets the FileDescriptor for this **Socket** (p. 2452), the Object is owned by this **Socket** (p. 2452) and should not be deleted by the caller.

Returns

a pointer to this **Socket** (p. 2452)'s FileDescriptor object.

6.493.3.8 std::string **decaf::net::SocketImpl::getInetAddress** () const [inline]

Gets the value of this **SocketImpl** (p. 2479)'s address field.

Returns

the value of the address field.

6.493.3.9 virtual **decaf::io::InputStream*** **decaf::net::SocketImpl::getInputStream** () [pure virtual]

Gets the InputStream linked to this **Socket** (p. 2452).

Returns

an InputStream pointer owned by the **Socket** (p. 2452) object.

Exceptions

<i>IOException</i>	if an I/O error occurs while attempting this operation.
--------------------	---

Implemented in **decaf::internal::net::tcp::TcpSocket** (p. 2684).

6.493.3.10 `virtual std::string decaf::net::SocketImpl::getLocalAddress () const`
`[pure virtual]`

Gets the value of the local Inet address the **Socket** (p. 2452) is bound to if bound, otherwise return the **InetAddress** (p. 1437) ANY value "0.0.0.0".

Returns

the local address bound to, or ANY.

Implemented in **decaf::internal::net::tcp::TcpSocket** (p. 2684).

6.493.3.11 `int decaf::net::SocketImpl::getLocalPort () const` `[inline]`

Gets the value of this **SocketImpl** (p. 2479)'s local port field.

Returns

the value of localPort.

6.493.3.12 `virtual int decaf::net::SocketImpl::getOption (int option) const` `[pure virtual]`

Gets the specified **Socket** (p. 2452) option.

Parameters

<i>option</i>	The Socket (p. 2452) options whose value is to be retrieved.
---------------	---

Returns

the value of the given socket option.

Exceptions

<i>IOException</i>	if an I/O error occurs while performing this operation.
--------------------	---

Implemented in **decaf::internal::net::tcp::TcpSocket** (p. 2684).

6.493.3.13 `virtual decaf::io::OutputStream* decaf::net::SocketImpl::getOutputStream ()` `[pure virtual]`

Gets the OutputStream linked to this **Socket** (p. 2452).

Returns

an OutputStream pointer owned by the **Socket** (p. 2452) object.

Exceptions

<i>IOException</i>	if an I/O error occurs while attempting this operation.
--------------------	---

Implemented in **decaf::internal::net::tcp::TcpSocket** (p. 2685).

6.493.3.14 `int decaf::net::SocketImpl::getPort () const` [inline]

Gets the port that this socket has been assigned.

Returns

the **Socket** (p. 2452)'s port number.

6.493.3.15 `virtual void decaf::net::SocketImpl::listen (int backlog)` [pure virtual]

Sets the maximum queue length for incoming connection indications (a request to connect) to the count argument.

If a connection indication arrives when the queue is full, the connection is refused.

Parameters

<i>backlog</i>	The maximum length of the connection queue.
----------------	---

Exceptions

<i>IOException</i>	if an I/O error occurs while attempting this operation.
--------------------	---

Implemented in **decaf::internal::net::tcp::TcpSocket** (p. 2685).

6.493.3.16 `virtual void decaf::net::SocketImpl::sendUrgentData (int data)` [virtual]

Sends on byte of urgent data to the **Socket** (p. 2452).

Parameters

<i>data</i>	The value to write as urgent data, only the lower eight bits are sent.
-------------	--

Exceptions

<i>IOException</i>	if an I/O error occurs while performing this operation.
--------------------	---

6.493.3.17 `virtual void decaf::net::SocketImpl::setOption (int option, int value)`
`[pure virtual]`

Sets the specified option on the **Socket** (p. 2452) if supported.

Parameters

<i>option</i>	The Socket (p. 2452) option to set.
<i>value</i>	The value of the socket option to apply to the socket.

Exceptions

<i>IOException</i>	if an I/O error occurs while performing this operation.
--------------------	---

Implemented in **decaf::internal::net::tcp::TcpSocket** (p. 2686).

6.493.3.18 `virtual void decaf::net::SocketImpl::shutdownInput ()` `[pure virtual]`

Places the input stream for this socket at "end of stream".

Any data sent to this socket is acknowledged and then silently discarded. If you read from a socket input stream after invoking **shutdownInput()** (p. 2486) on the socket, the stream will return EOF.

Exceptions

<i>IOException</i>	if an I/O error occurs while attempting this operation.
--------------------	---

Implemented in **decaf::internal::net::tcp::TcpSocket** (p. 2687).

6.493.3.19 `virtual void decaf::net::SocketImpl::shutdownOutput ()` `[pure virtual]`

Disables the output stream for this socket.

For a TCP socket, any previously written data will be sent followed by TCP's normal connection termination sequence. If you write to a socket output stream after invoking **shutdownOutput()** (p. 2486) on the socket, the stream will throw an **IOException**.

Exceptions

<i>IOException</i>	if an I/O error occurs while attempting this operation.
--------------------	---

Implemented in **decaf::internal::net::tcp::TcpSocket** (p. 2687).

6.493.3.20 `virtual bool decaf::net::SocketImpl::supportsUrgentData () const`
`[inline, virtual]`

Returns

true if this **SocketImpl** (p. 2479) supports sending Urgent Data. The default implementation always returns false.

6.493.3.21 `std::string decaf::net::SocketImpl::toString () const`

Returns a string containing the address and port of this **Socket** (p. 2452) instance.

Returns

a string containing the address and port of this socket.

6.493.4 Field Documentation

6.493.4.1 `std::string decaf::net::SocketImpl::address` `[protected]`

The Remote Address that the **Socket** (p. 2452) is connected to.

6.493.4.2 `io::FileDescriptor* decaf::net::SocketImpl::fd` `[protected]`

The File Descriptor for this **Socket** (p. 2452).

6.493.4.3 `int decaf::net::SocketImpl::localPort` `[protected]`

The port on the Local Machine that this **Socket** (p. 2452) is Bound to.

6.493.4.4 `int decaf::net::SocketImpl::port` `[protected]`

The remote port that this **Socket** (p. 2452) is connected to.

The documentation for this class was generated from the following file:

- `src/main/decaf/net/SocketImpl.h`

6.494 decaf::net::SocketImplFactory Class Reference

Factory class interface for a Factory that creates SocketImpl objects.

```
#include <src/main/decaf/net/SocketImplFactory.h>
```

Public Member Functions

- virtual `~SocketImplFactory()`
- virtual `SocketImpl * createSocketImpl()=0`

*Creates a new SokcetImpl instance and returns it, the caller then owns the instance and must delete it when finished with the **SocketImpl** (p. 2479).*

6.494.1 Detailed Description

Factory class interface for a Factory that creates SocketImpl objects.

These factories can be used to create various types of Sockets, e.g. Streaming, - Multicast, SSL, or platform specific variations of these types.

See also

decaf::net::Socket (p. 2452)

decaf::net::ServerSocket (p. 2342)

Since

1.0

6.494.2 Constructor & Destructor Documentation

- 6.494.2.1 `virtual decaf::net::SocketImplFactory::~~SocketImplFactory()`
`[inline, virtual]`

6.494.3 Member Function Documentation

- 6.494.3.1 `virtual SocketImpl* decaf::net::SocketImplFactory::createSocketImpl()`
`[pure virtual]`

Creates a new SokcetImpl instance and returns it, the caller then owns the instance and must delete it when finished with the **SocketImpl** (p. 2479).

Returns

new **SocketImpl** (p. 2479) instance that is owned by the caller.

The documentation for this class was generated from the following file:

- `src/main/decaf/net/SocketImplFactory.h`

6.495 decaf::net::SocketOptions Class Reference

```
#include <src/main/decaf/net/SocketOptions.h>
```

Inheritance diagram for decaf::net::SocketOptions:

Public Member Functions

- virtual `~SocketOptions()`

Static Public Attributes

- static const int **SOCKET_OPTION_TCP_NODELAY**
Disable Nagle's algorithm for this connection.
- static const int **SOCKET_OPTION_BINDADDR**
Fetch the local address binding of a socket (this option cannot be "set" only "gotten", since sockets are bound at creation time, and so the locally bound address cannot be changed).
- static const int **SOCKET_OPTION_REUSEADDR**
Sets SO_REUSEADDR for a socket.
- static const int **SOCKET_OPTION_BROADCAST**
Sets SO_BROADCAST for a socket.
- static const int **SOCKET_OPTION_IP_MULTICAST_IF**
Set which outgoing interface on which to send multicast packets.
- static const int **SOCKET_OPTION_IP_MULTICAST_IF2**
Same as above.
- static const int **SOCKET_OPTION_IP_MULTICAST_LOOP**
This option enables or disables local loopback of multicast datagrams.
- static const int **SOCKET_OPTION_IP_TOS**
This option sets the type-of-service or traffic class field in the IP header for a TCP or UDP socket.
- static const int **SOCKET_OPTION_LINGER**
Specify a linger-on-close timeout.
- static const int **SOCKET_OPTION_TIMEOUT**
*Set a timeout on blocking **Socket** (p. 2452) operations.*
- static const int **SOCKET_OPTION_SNDBUF**
Set a hint the size of the underlying buffers used by the platform for outgoing network I/O.
- static const int **SOCKET_OPTION_RCVBUF**
Set a hint the size of the underlying buffers used by the platform for incoming network I/O.
- static const int **SOCKET_OPTION_KEEPAIVE**
When the keepalive option is set for a TCP socket and no data has been exchanged across the socket in either direction for 2 hours (NOTE: the actual value is implementation dependent), TCP automatically sends a keepalive probe to the peer.
- static const int **SOCKET_OPTION_OOBINLINE**
When the OOBINLINE option is set, any TCP urgent data received on the socket will be received through the socket input stream.

6.495.1 Detailed Description

Since

1.0

6.495.2 Constructor & Destructor Documentation

6.495.2.1 `virtual decaf::net::SocketOptions::~~SocketOptions ()` [virtual]

6.495.3 Field Documentation

6.495.3.1 `const int decaf::net::SocketOptions::SOCKET_OPTION_BINDADDR`
[static]

Fetch the local address binding of a socket (this option cannot be "set" only "gotten", since sockets are bound at creation time, and so the locally bound address cannot be changed).

The default local address of a socket is `INADDR_ANY`, meaning any local address on a multi-homed host. A multi-homed host can use this option to accept connections to only one of its addresses (in the case of a **ServerSocket** (p. 2342) or `DatagramSocket`), or to specify its return address to the peer (for a **Socket** (p. 2452) or `DatagramSocket`). The parameter of this option is an **InetAddress** (p. 1437).

6.495.3.2 `const int decaf::net::SocketOptions::SOCKET_OPTION_BROADCAST`
[static]

Sets `SO_BROADCAST` for a socket.

This option enables and disables the ability of the process to send broadcast messages. It is supported for only datagram sockets and only on networks that support the concept of a broadcast message (e.g. Ethernet, token ring, etc.), and it is set by default for `DatagramSockets`.

6.495.3.3 `const int decaf::net::SocketOptions::SOCKET_OPTION_IP_MULTICAST-
_IF` [static]

Set which outgoing interface on which to send multicast packets.

Useful on hosts with multiple network interfaces, where applications want to use other than the system default. Takes/returns an **InetAddress** (p. 1437).

Valid for Multicast: `DatagramSocketImpl`.

6.495.3.4 `const int decaf::net::SocketOptions::SOCKET_OPTION_IP_MULTICAST-
_IF2` [static]

Same as above.

This option is introduced so that the behaviour with `IP_MULTICAST_IF` will be kept the same as before, while this new option can support setting outgoing interfaces with either IPv4 and IPv6 addresses.

6.495.3.5 `const int decaf::net::SocketOptions::SOCKET_OPTION_IP_MULTICAST_LOOP` `[static]`

This option enables or disables local loopback of multicast datagrams.

This option is enabled by default for Multicast Sockets.

6.495.3.6 `const int decaf::net::SocketOptions::SOCKET_OPTION_IP_TOS` `[static]`

This option sets the type-of-service or traffic class field in the IP header for a TCP or UDP socket.

6.495.3.7 `const int decaf::net::SocketOptions::SOCKET_OPTION_KEEPAIVE` `[static]`

When the keepalive option is set for a TCP socket and no data has been exchanged across the socket in either direction for 2 hours (NOTE: the actual value is implementation dependent), TCP automatically sends a keepalive probe to the peer.

This probe is a TCP segment to which the peer must respond. One of three responses is expected: 1. The peer responds with the expected ACK. The application is not notified (since everything is OK). TCP will send another probe following another 2 hours of inactivity. 2. The peer responds with an RST, which tells the local TCP that the peer host has crashed and rebooted. The socket is closed. 3. There is no response from the peer. The socket is closed. The purpose of this option is to detect if the peer host crashes.

Valid only for TCP socket: **SocketImpl** (p. 2479)

6.495.3.8 `const int decaf::net::SocketOptions::SOCKET_OPTION_LINGER` `[static]`

Specify a linger-on-close timeout.

This option disables/enables immediate return from a `close()` of a TCP **Socket** (p. 2452). Enabling this option with a non-zero Integer timeout means that a `close()` will block pending the transmission and acknowledgment of all data written to the peer, at which point the socket is closed gracefully. Upon reaching the linger timeout, the socket is closed forcefully, with a TCP RST. Enabling the option with a timeout of zero does a forceful close immediately. If the specified timeout value exceeds 65,535 it will be reduced to 65,535.

Valid only for TCP: **SocketImpl** (p. 2479)

6.495.3.9 `const int decaf::net::SocketOptions::SOCKET_OPTION_OOBLINE`
[static]

When the OOBLINE option is set, any TCP urgent data received on the socket will be received through the socket input stream.

When the option is disabled (which is the default) urgent data is silently discarded.

6.495.3.10 `const int decaf::net::SocketOptions::SOCKET_OPTION_RCVBUF`
[static]

Set a hint the size of the underlying buffers used by the platform for incoming network I/O.

When used in set, this is a suggestion to the kernel from the application about the size of buffers to use for the data to be received over the socket. When used in get, this must return the size of the buffer actually used by the platform when receiving in data on this socket. Valid for all sockets: **SocketImpl** (p. 2479), **DatagramSocketImpl**.

6.495.3.11 `const int decaf::net::SocketOptions::SOCKET_OPTION_REUSEADDR`
[static]

Sets SO_REUSEADDR for a socket.

This is used only for MulticastSockets in decaf, and it is set by default for MulticastSockets.

6.495.3.12 `const int decaf::net::SocketOptions::SOCKET_OPTION_SNDBUF`
[static]

Set a hint the size of the underlying buffers used by the platform for outgoing network I/O.

When used in set, this is a suggestion to the kernel from the application about the size of buffers to use for the data to be sent over the socket. When used in get, this must return the size of the buffer actually used by the platform when sending out data on this socket. Valid for all sockets: **SocketImpl** (p. 2479), **DatagramSocketImpl**

6.495.3.13 `const int decaf::net::SocketOptions::SOCKET_OPTION_TCP_NODELAY`
[static]

Disable Nagle's algorithm for this connection.

Written data to the network is not buffered pending acknowledgment of previously written data. Valid for TCP sockets.

6.495.3.14 `const int decaf::net::SocketOptions::SOCKET_OPTION_TIMEOUT`
`[static]`

Set a timeout on blocking **Socket** (p. 2452) operations.

The option must be set prior to entering a blocking operation to take effect.

The documentation for this class was generated from the following file:

- `src/main/decaf/net/SocketOptions.h`

6.496 decaf::net::SocketTimeoutException Class Reference

```
#include <src/main/decaf/net/SocketTimeoutException.h>
```

Inheritance diagram for `decaf::net::SocketTimeoutException`:

Public Member Functions

- **SocketTimeoutException** () throw ()
Default Constructor.
- **SocketTimeoutException** (const **Exception** &ex) throw ()
Conversion Constructor from some other Exception.
- **SocketTimeoutException** (const **SocketTimeoutException** &ex) throw ()
Copy Constructor.
- **SocketTimeoutException** (const char *file, const int lineNumber, const std::exception *cause, const char *msg,...) throw ()
Constructor - Initializes the file name and line number where this message occurred.
- **SocketTimeoutException** (const std::exception *cause) throw ()
Constructor.
- **SocketTimeoutException** (const char *file, const int lineNumber, const char *msg,...) throw ()
Constructor - Initializes the file name and line number where this message occurred.
- virtual **SocketTimeoutException** * clone () const
Clones this exception.
- virtual ~**SocketTimeoutException** () throw ()

6.496.1 Constructor & Destructor Documentation

6.496.1.1 `decaf::net::SocketTimeoutException::SocketTimeoutException ()`
`throw () [inline]`

Default Constructor.

6.496.1.2 **decaf::net::SocketTimeoutException::SocketTimeoutException (const Exception & ex) throw ()** `[inline]`

Conversion Constructor from some other Exception.

Parameters

<i>ex</i>	An exception that should become this type of Exception
-----------	--

6.496.1.3 **decaf::net::SocketTimeoutException::SocketTimeoutException (const SocketTimeoutException & ex) throw ()** `[inline]`

Copy Constructor.

Parameters

<i>ex</i>	An exception that should become this type of Exception
-----------	--

6.496.1.4 **decaf::net::SocketTimeoutException::SocketTimeoutException (const char * file, const int lineNumber, const std::exception * cause, const char * msg, ...) throw ()** `[inline]`

Constructor - Initializes the file name and line number where this message occurred.

Sets the message to report, using an optional list of arguments to parse into the message

Parameters

<i>file</i>	The file name where exception occurs
<i>lineNumber</i>	The line number where the exception occurred.
<i>cause</i>	The exception that was the cause for this one to be thrown.
<i>msg</i>	The message to report
<i>...</i>	list of primitives that are formatted into the message

6.496.1.5 **decaf::net::SocketTimeoutException::SocketTimeoutException (const std::exception * cause) throw ()** `[inline]`

Constructor.

Parameters

<i>cause</i>	Pointer to the exception that caused this one to be thrown, the object is cloned caller retains ownership.
--------------	--

6.496.1.6 **decaf::net::SocketTimeoutException::SocketTimeoutException** (const char * *file*, const int *lineNumber*, const char * *msg*, ...) throw () [inline]

Constructor - Initializes the file name and line number where this message occurred.

Sets the message to report, using an optional list of arguments to parse into the message

Parameters

<i>file</i>	The file name where exception occurs
<i>lineNumber</i>	The line number where the exception occurred.
<i>msg</i>	The message to report
...	list of primitives that are formatted into the message

6.496.1.7 **virtual decaf::net::SocketTimeoutException::~~SocketTimeoutException** () throw () [inline, virtual]

6.496.2 Member Function Documentation

6.496.2.1 **virtual SocketTimeoutException* decaf::net::SocketTimeoutException::clone** () const [inline, virtual]

Clones this exception.

This is useful for cases where you need to preserve the type of the original exception as well as the message. All subclasses should override.

Returns

a new Exception instance that is a copy of this Exception object.

Reimplemented from **decaf::io::InterruptedIOException** (p. 1533).

The documentation for this class was generated from the following file:

- src/main/decaf/net/**SocketTimeoutException.h**

6.497 decaf::net::ssl::SSLContext Class Reference

Represents on implementation of the Secure **Socket** (p.2452) Layer for streaming based sockets.

```
#include <src/main/decaf/net/ssl/SSLContext.h>
```

Public Member Functions

- **SSLContext** (SSLContextSpi *contextImpl)

- virtual `~SSLContext ()`
- `SocketFactory * getSocketFactory ()`
*Returns an **SocketFactory** (p. 2473) instance for use with this Context, the **SocketFactory** (p. 2473) is owned by the Context and should not be deleted by the caller.*
- `ServerSocketFactory * getServerSocketFactory ()`
*Returns an **ServerSocketFactory** (p. 2352) instance for use with this Context, the **ServerSocketFactory** (p. 2352) is owned by the Context and should not be deleted by the caller.*
- `SSLParameters * getDefaultSSLParameters ()`
- `SSLParameters * getSupportedSSLParameters ()`

Static Public Member Functions

- static `SSLContext * getDefault ()`
*Gets the Default **SSLContext** (p. 2495).*
- static void `setDefault (SSLContext *context)`
*Sets the default **SSLContext** (p. 2495) to be returned from future calls to `getDefault`.*

6.497.1 Detailed Description

Represents an implementation of the Secure **Socket** (p. 2452) Layer for streaming based sockets.

This class serves as a source of factories to be used to create new SSL **Socket** (p. 2452) instances.

Since

1.0

6.497.2 Constructor & Destructor Documentation

6.497.2.1 `decaf::net::ssl::SSLContext::SSLContext (SSLContextSpi * contextImpl)`

6.497.2.2 `virtual decaf::net::ssl::SSLContext::~~SSLContext ()` [virtual]

6.497.3 Member Function Documentation

6.497.3.1 `static SSLContext* decaf::net::ssl::SSLContext::getDefault ()`
 [static]

Gets the Default **SSLContext** (p. 2495).

The default instance of the **SSLContext** (p. 2495) should be immediately usable without any need for the client to initialize this context.

Returns

a pointer to the Default **SSLContext** (p. 2495) instance.

6.497.3.2 **SSLParameters*** decaf::net::ssl::SSLContext::getDefaultSSLParameters ()

Returns

a new instance of an **SSLParameters** (p. 2501) object containing the default set of settings for this **SSLContext** (p. 2495).

Exceptions

<i>UnsupportedOperationException</i>	if the parameters cannot be retrieved.
--------------------------------------	--

6.497.3.3 **ServerSocketFactory*** decaf::net::ssl::SSLContext::getServerSocketFactory ()

Returns an **ServerSocketFactory** (p. 2352) instance for use with this Context, the **ServerSocketFactory** (p. 2352) is owned by the Context and should not be deleted by the caller.

Returns

a pointer to this **SSLContext** (p. 2495)'s **ServerSocketFactory** (p. 2352) for creating **SSLServerSocket** (p. 2504) objects.

Exceptions

<i>IllegalStateException</i>	if the SSLContextSpi (p. 2498) requires initialization but it has not yet been initialized.
------------------------------	--

6.497.3.4 **SocketFactory*** decaf::net::ssl::SSLContext::getSocketFactory ()

Returns an **SocketFactory** (p. 2473) instance for use with this Context, the **SocketFactory** (p. 2473) is owned by the Context and should not be deleted by the caller.

Returns

a pointer to this **SSLContext** (p. 2495)'s **SocketFactory** (p. 2473) for creating **SSLSocket** (p. 2513) objects.

Exceptions

<i>IllegalStateException</i>	if the SSLContextSpi (p. 2498) requires initialization but it has not yet been initialized.
------------------------------	--

6.497.3.5 **SSLParameters*** **decaf::net::ssl::SSLContext::getSupportedSSLParameters ()**

Returns

a new instance of an **SSLParameters** (p. 2501) object containing the complete set of settings for this **SSLContext** (p. 2495).

Exceptions

<i>UnsupportedOperationException</i>	if the parameters cannot be retrieved.
--------------------------------------	--

6.497.3.6 **static void decaf::net::ssl::SSLContext::setDefault (SSLContext * context) [static]**

Sets the default **SSLContext** (p. 2495) to be returned from future calls to `getDefault`.

The set **SSLContext** (p. 2495) must be fully initialized and usable. The caller is responsible for deleting this object before the Library shutdown methods are called.

Exceptions

<i>NullPointerException</i>	if the context passed is NULL.
-----------------------------	--------------------------------

The documentation for this class was generated from the following file:

- `src/main/decaf/net/ssl/SSLContext.h`

6.498 **decaf::net::ssl::SSLContextSpi** Class Reference

Defines the interface that should be provided by an **SSLContext** (p. 2495) provider.

```
#include <src/main/decaf/net/ssl/SSLContextSpi.h>
```

Inheritance diagram for `decaf::net::ssl::SSLContextSpi`:

Public Member Functions

- `virtual ~SSLContextSpi ()`

- virtual void **providerInit** (**security::SecureRandom** *random)=0
Perform the initialization of this Context.
- virtual **SSLParameters** * **providerGetDefaultSSLParameters** ()
*Creates a returns a new **SSLParameters** (p. 2501) instance that contains the default settings for this Providers **SSLContext** (p. 2495).*
- virtual **SSLParameters** * **providerGetSupportedSSLParameters** ()
*Creates and returns a new **SSLParameters** (p. 2501) instance that contains the full set of supported parameters for this SSL Context.*
- virtual **SocketFactory** * **providerGetSocketFactory** ()=0
*Returns a **SocketFactory** (p. 2473) instance that can be used to create new **SSL-Socket** (p. 2513) objects.*
- virtual **ServerSocketFactory** * **providerGetServerSocketFactory** ()=0
*Returns a **ServerSocketFactory** (p. 2352) instance that can be used to create new **SSLServerSocket** (p. 2504) objects.*

6.498.1 Detailed Description

Defines the interface that should be provided by an **SSLContext** (p. 2495) provider.

Since

1.0

6.498.2 Constructor & Destructor Documentation

6.498.2.1 virtual decaf::net::ssl::SSLContextSpi::~SSLContextSpi ()
[virtual]

6.498.3 Member Function Documentation

6.498.3.1 virtual **SSLParameters*** decaf::net::ssl::SSLContextSpi::providerGetDefaultSSLParameters () [virtual]

Creates a returns a new **SSLParameters** (p. 2501) instance that contains the default settings for this Providers **SSLContext** (p. 2495).

The returned **SSLParameters** (p. 2501) instance is requires to have non-empty values in its ciphersuites and protocols.

Returns

new **SSLParameters** (p. 2501) instance with the **SSLContext** (p. 2495) defaults.

Exceptions

<i>Unsupported-OperationException</i>	if the defaults cannot be obtained.
---------------------------------------	-------------------------------------

6.498.3.2 **virtual ServerSocketFactory* decaf::net::ssl::SSLContextSpi::providerGetServerSocketFactory ()** [pure virtual]

Returns a **ServerSocketFactory** (p. 2352) instance that can be used to create new **SSLServerSocket** (p. 2504) objects.

The **ServerSocketFactory** (p. 2352) is owned by the Service Provider and should not be destroyed by the caller.

Returns

SocketFactory (p. 2473) instance that can be used to create new **SSLServerSockets**.

Exceptions

<i>IllegalStateException</i>	if the SSLContextSpi (p. 2498) object requires initialization but has not been initialized yet.
------------------------------	--

Implemented in **decaf::internal::net::ssl::openssl::OpenSSLContextSpi** (p. 2000).

6.498.3.3 **virtual SocketFactory* decaf::net::ssl::SSLContextSpi::providerGetSocketFactory ()** [pure virtual]

Returns a **SocketFactory** (p. 2473) instance that can be used to create new **SSLSocket** (p. 2513) objects.

The **SocketFactory** (p. 2473) is owned by the Service Provider and should not be destroyed by the caller.

Returns

SocketFactory (p. 2473) instance that can be used to create new **SSLSockets**.

Exceptions

<i>IllegalStateException</i>	if the SSLContextSpi (p. 2498) object requires initialization but has not been initialized yet.
------------------------------	--

Implemented in **decaf::internal::net::ssl::openssl::OpenSSLContextSpi** (p. 2000).

6.498.3.4 **virtual SSLParameters* decaf::net::ssl::SSLContextSpi::providerGetSupportedSSLParameters ()** [virtual]

Creates and returns a new **SSLParameters** (p. 2501) instance that contains the full set of supported parameters for this SSL Context.

The returned **SSLParameters** (p. 2501) instance is requires to have non-empty values in its ciphersuites and protocols.

Returns

a new **SSLParameters** (p. 2501) instance with the full set of settings that are supported.

Exceptions

<i>Unsupported-OperationException</i>	if the supported parameters cannot be obtained.
---------------------------------------	---

6.498.3.5 virtual void decaf::net::ssl::SSLContextSpi::providerInit (security::SecureRandom * random) [pure virtual]

Perform the initialization of this Context.

Parameters

<i>random</i>	Pointer to an instance of a secure random number generator.
---------------	---

Exceptions

<i>NullPointerException</i>	if the SecureRandom instance is NULL.
<i>KeyManagement-Exception</i>	if an error occurs while initializing the context.

Implemented in **decaf::internal::net::ssl::openssl::OpenSSLContextSpi** (p. 2000).

The documentation for this class was generated from the following file:

- src/main/decaf/net/ssl/**SSLContextSpi.h**

6.499 decaf::net::ssl::SSLParameters Class Reference

```
#include <src/main/decaf/net/ssl/SSLParameters.h>
```

Public Member Functions

- **SSLParameters** ()
Creates a new **SSLParameters** (p. 2501) instance with empty vectors for the protocols and the cipherSuites, the wantClientAuth and needClientAuth flags are set to false.
- **SSLParameters** (const std::vector< std::string > &cipherSuites)

Creates a new **SSLParameters** (p. 2501) instance with the given cipherSuites value, the protocols vector is empty and the wantClientAuth and needClientAuth flags are set to false.

- **SSLParameters** (const std::vector< std::string > &cipherSuites, const std::vector< std::string > &protocols)

Creates a new **SSLParameters** (p. 2501) instance with the given cipherSuites value and protocols value, the wantClientAuth and needClientAuth flags are set to false.

- virtual ~**SSLParameters** ()
- std::vector< std::string > **getCipherSuites** () const
- void **setCipherSuites** (const std::vector< std::string > &cipherSuites)

Sets the vector of ciphersuites.

- std::vector< std::string > **getProtocols** () const
- void **setProtocols** (const std::vector< std::string > &protocols)

Sets the vector of protocols.

- bool **getWantClientAuth** () const
- void **setWantClientAuth** (bool wantClientAuth)

Sets whether client authentication should be requested.

- bool **getNeedClientAuth** () const
- void **setNeedClientAuth** (bool needClientAuth)

Sets whether client authentication should be required.

6.499.1 Constructor & Destructor Documentation

6.499.1.1 decaf::net::ssl::SSLParameters::SSLParameters ()

Creates a new **SSLParameters** (p. 2501) instance with empty vectors for the protocols and the cipherSuites, the wantClientAuth and needClientAuth flags are set to false.

6.499.1.2 decaf::net::ssl::SSLParameters::SSLParameters (const std::vector< std::string > & cipherSuites)

Creates a new **SSLParameters** (p. 2501) instance with the given cipherSuites value, the protocols vector is empty and the wantClientAuth and needClientAuth flags are set to false.

Parameters

<i>cipherSuites</i>	The vector of cipherSuites for this SSLParameters (p. 2501) instance (can be empty).
---------------------	---

6.499.1.3 decaf::net::ssl::SSLParameters::SSLParameters (const std::vector< std::string > & cipherSuites, const std::vector< std::string > & protocols)

Creates a new **SSLParameters** (p. 2501) instance with the given cipherSuites value and protocols value, the wantClientAuth and needClientAuth flags are set to false.

Parameters

<i>cipherSuites</i>	The vector of cipherSuites for this SSLParameters (p. 2501) instance (can be empty).
<i>protocols</i>	The vector of protocols for this SSLParameters (p. 2501) instance (can be empty).

6.499.1.4 **virtual decaf::net::ssl::SSLParameters::~~SSLParameters ()**
[virtual]

6.499.2 Member Function Documentation

6.499.2.1 **std::vector<std::string> decaf::net::ssl::SSLParameters::getCipherSuites () const** [inline]

Returns

a copy of the vector of ciphersuites or an empty vector if none have been set.

6.499.2.2 **bool decaf::net::ssl::SSLParameters::getNeedClientAuth () const**
[inline]

Returns

whether client authentication should be required.

6.499.2.3 **std::vector<std::string> decaf::net::ssl::SSLParameters::getProtocols () const** [inline]

Returns

a copy of the vector of protocols or an empty vector if none have been set.

6.499.2.4 **bool decaf::net::ssl::SSLParameters::getWantClientAuth () const**
[inline]

Returns

whether client authentication should be requested.

6.499.2.5 **void decaf::net::ssl::SSLParameters::setCipherSuites (const std::vector<std::string> & cipherSuites)** [inline]

Sets the vector of ciphersuites.

Parameters

<i>cipherSuites</i>	The vector of cipherSuites (can be an empty vector).
---------------------	--

6.499.2.6 void decaf::net::ssl::SSLParameters::setNeedClientAuth (bool
needClientAuth) [inline]

Sets whether client authentication should be required.

Calling this method clears the wantClientAuth flag.

Parameters

<i>needClientAuth</i>	whether client authentication should be required.
-----------------------	---

6.499.2.7 void decaf::net::ssl::SSLParameters::setProtocols (const std::vector<
std::string > & protocols) [inline]

Sets the vector of protocols.

Parameters

<i>protocols</i>	the vector of protocols (or an empty vector)
------------------	--

6.499.2.8 void decaf::net::ssl::SSLParameters::setWantClientAuth (bool
wantClientAuth) [inline]

Sets whether client authentication should be requested.

Calling this method clears the needClientAuth flag.

Parameters

<i>whether</i>	client authentication should be requested.
----------------	--

The documentation for this class was generated from the following file:

- src/main/decaf/net/ssl/SSLParameters.h

6.500 decaf::net::ssl::SSLServerSocket Class Reference

Represents a server socket that is used to accept connections from clients using the Secure Sockets protocol or the Top Level Security protocol.

```
#include <src/main/decaf/net/ssl/SSLServerSocket.h>
```

Inheritance diagram for decaf::net::ssl::SSLServerSocket:

Public Member Functions

- virtual **~SSLServerSocket** ()
- virtual std::vector< std::string > **getSupportedCipherSuites** () const =0
*Gets a vector containing the names of all the cipher suites that are supported by this **SSLServerSocket** (p. 2504).*
- virtual std::vector< std::string > **getSupportedProtocols** () const =0
*Gets a vector containing the names of all the protocols that could be enabled for this **SSLServerSocket** (p. 2504) instance.*
- virtual std::vector< std::string > **getEnabledCipherSuites** () const =0
*Returns a vector containing the names of all the currently enabled Cipher Suites for this **SSLServerSocket** (p. 2504).*
- virtual void **setEnabledCipherSuites** (const std::vector< std::string > &suites)=0
*Sets the Cipher Suites that are to be enabled on the **SSLServerSocket** (p. 2504) connection.*
- virtual std::vector< std::string > **getEnabledProtocols** () const =0
*Returns a vector containing the names of all the currently enabled Protocols for this **SSLServerSocket** (p. 2504).*
- virtual void **setEnabledProtocols** (const std::vector< std::string > &protocols)=0
*Sets the Protocols that are to be enabled on the **SSLServerSocket** (p. 2504) connection.*
- virtual bool **getWantClientAuth** () const =0
- virtual void **setWantClientAuth** (bool value)=0
*Sets whether or not this **Socket** (p. 2452) will request Client Authentication.*
- virtual bool **getNeedClientAuth** () const =0
- virtual void **setNeedClientAuth** (bool value)=0
*Sets whether or not this **Socket** (p. 2452) will require Client Authentication.*

Protected Member Functions

- **SSLServerSocket** ()
Creates a non-bound server socket.
- **SSLServerSocket** (int port)
*Creates a new **ServerSocket** (p. 2342) bound to the specified port, if the value of port is 0, then any free port is chosen.*
- **SSLServerSocket** (int port, int backlog)
*Creates a new **ServerSocket** (p. 2342) bound to the specified port, if the value of port is 0, then any free port is chosen.*

- **SSLServerSocket** (int port, int backlog, const **decaf::net::InetAddress** *address)

*Creates a new **ServerSocket** (p. 2342) bound to the specified port, if the value of port is 0, then any free port is chosen.*

6.500.1 Detailed Description

Represents a server socket that is used to accept connections from clients using the Secure Sockets protocol or the Top Level Security protocol.

The main function of this class is to create **SSLSocket** (p. 2513) objects by accepting connections from client sockets over SSL.

Since

1.0

6.500.2 Constructor & Destructor Documentation

6.500.2.1 **decaf::net::ssl::SSLServerSocket::SSLServerSocket** () [protected]

Creates a non-bound server socket.

6.500.2.2 **decaf::net::ssl::SSLServerSocket::SSLServerSocket** (int port) [protected]

Creates a new **ServerSocket** (p. 2342) bound to the specified port, if the value of port is 0, then any free port is chosen.

When this constructor is called the size of the backlog queue is set at 50, connections that arrive after the backlog has been reached are refused.

If a **SocketImplFactory** (p. 2487) is registered then the createSocketImpl method on the factory will be called otherwise a default **SocketImpl** (p. 2479) is created.

Parameters

<i>port</i>	The port to bind the ServerSocket (p. 2342) to.
-------------	--

Exceptions

<i>IOException</i>	if there is an I/O error while performing this operation.
<i>IllegalArgument-Exception</i>	if the port value is negative or greater than 65535.

6.500.2.3 **decaf::net::ssl::SSLServerSocket::SSLServerSocket** (*int port*, *int backlog*) [protected]

Creates a new **ServerSocket** (p. 2342) bound to the specified port, if the value of port is 0, then any free port is chosen.

When this constructor is called the size of the backlog queue is set at backlog, connections that arrive after the backlog has been reached are refused. If backlog is zero or negative then the default backlog value of 50 is used.

If a **SocketImplFactory** (p. 2487) is registered then the createSocketImpl method on the factory will be called otherwise a default **SocketImpl** (p. 2479) is created.

Parameters

<i>port</i>	The port to bind the ServerSocket (p. 2342) to.
<i>backlog</i>	The the number of incoming connection attempts to queue before connections are refused.

Exceptions

<i>IOException</i>	if there is an I/O error while performing this operation.
<i>IllegalArgument-Exception</i>	if the port value is negative or greater than 65535.

6.500.2.4 **decaf::net::ssl::SSLServerSocket::SSLServerSocket** (*int port*, *int backlog*, *const decaf::net::InetAddress * address*) [protected]

Creates a new **ServerSocket** (p. 2342) bound to the specified port, if the value of port is 0, then any free port is chosen.

If the value of the ifAddress is empty or NULL then the ANY address is used.

When this constructor is called the size of the backlog queue is set at backlog, connections that arrive after the backlog has been reached are refused. If backlog is zero or negative then the default backlog value of 50 is used.

If a **SocketImplFactory** (p. 2487) is registered then the createSocketImpl method on the factory will be called otherwise a default **SocketImpl** (p. 2479) is created.

Parameters

<i>port</i>	The port to bind the ServerSocket (p. 2342) to.
<i>backlog</i>	The the number of incoming connection attempts to queue before connections are refused.
<i>ifAddress</i>	The IP Address to bind to on the local machine.

Exceptions

<i>IOException</i>	if there is an I/O error while performing this operation.
<i>IllegalArgument-Exception</i>	if the port value is negative or greater than 65535.

6.500.2.5 `virtual decaf::net::ssl::SSLServerSocket::~SSLServerSocket ()`
[virtual]

6.500.3 Member Function Documentation

6.500.3.1 `virtual std::vector<std::string> decaf::net::ssl::SSLServerSocket::getEnabledCipherSuites () const` [pure virtual]

Returns a vector containing the names of all the currently enabled Cipher Suites for this **SSLServerSocket** (p. 2504).

Returns

vector of the names of all enabled Cipher Suites.

Implemented in **decaf::internal::net::ssl::openssl::OpenSSLServerSocket** (p. 2006).

6.500.3.2 `virtual std::vector<std::string> decaf::net::ssl::SSLServerSocket::getEnabledProtocols () const` [pure virtual]

Returns a vector containing the names of all the currently enabled Protocols for this **SSLServerSocket** (p. 2504).

Returns

vector of the names of all enabled Protocols.

Implemented in **decaf::internal::net::ssl::openssl::OpenSSLServerSocket** (p. 2006).

6.500.3.3 `virtual bool decaf::net::ssl::SSLServerSocket::getNeedClientAuth () const` [pure virtual]

Returns

true if the **Socket** (p. 2452) requires client Authentication.

Implemented in **decaf::internal::net::ssl::openssl::OpenSSLServerSocket** (p. 2007).

6.500.3.4 `virtual std::vector<std::string> decaf::net::ssl::SSLServerSocket::getSupportedCipherSuites () const [pure virtual]`

Gets a vector containing the names of all the cipher suites that are supported by this **SSLServerSocket** (p. 2504).

Normally not all of these cipher suites will be enabled on the **Socket** (p. 2452).

Returns

a vector containing the names of all the supported cipher suites.

Implemented in **decaf::internal::net::ssl::openssl::OpenSSLServerSocket** (p. 2007).

6.500.3.5 `virtual std::vector<std::string> decaf::net::ssl::SSLServerSocket::getSupportedProtocols () const [pure virtual]`

Gets a vector containing the names of all the protocols that could be enabled for this **SSLServerSocket** (p. 2504) instance.

Returns

a vector containing the names of all the supported protocols.

Implemented in **decaf::internal::net::ssl::openssl::OpenSSLServerSocket** (p. 2007).

6.500.3.6 `virtual bool decaf::net::ssl::SSLServerSocket::getWantClientAuth () const [pure virtual]`

Returns

true if the **Socket** (p. 2452) request client Authentication.

Implemented in **decaf::internal::net::ssl::openssl::OpenSSLServerSocket** (p. 2007).

6.500.3.7 `virtual void decaf::net::ssl::SSLServerSocket::setEnabledCipherSuites (const std::vector< std::string > & suites) [pure virtual]`

Sets the Cipher Suites that are to be enabled on the **SSLServerSocket** (p. 2504) connection.

Each of the named Cipher Suites must appear in the list of supported cipher suites for this connection or an exception will be thrown.

Parameters

<i>suites</i>	An Vector of names for all the Cipher Suites that are to be enabled.
---------------	--

Exceptions

<i>IllegalArgument-Exception</i>	if the vector is empty or one of the names is invalid.
----------------------------------	--

Implemented in **decaf::internal::net::ssl::openssl::OpenSSLServerSocket** (p. 2008).

6.500.3.8 `virtual void decaf::net::ssl::SSLServerSocket::setEnabledProtocols (const std::vector< std::string > & protocols) [pure virtual]`

Sets the Protocols that are to be enabled on the **SSLServerSocket** (p. 2504) connection.

Each of the named Protocols must appear in the list of supported protocols suites for this connection or an exception will be thrown.

Parameters

<i>protocols</i>	An Vector of names for all the Protocols that are to be enabled.
------------------	--

Exceptions

<i>IllegalArgument-Exception</i>	if the vector is empty or one of the names is invalid.
----------------------------------	--

Implemented in **decaf::internal::net::ssl::openssl::OpenSSLServerSocket** (p. 2008).

6.500.3.9 `virtual void decaf::net::ssl::SSLServerSocket::setNeedClientAuth (bool value) [pure virtual]`

Sets whether or not this **Socket** (p. 2452) will require Client Authentication.

If set to true the **Socket** (p. 2452) (when used in server mode) will require that the client authenticate itself, if the client doesn't send authentication the socket will not allow negotiation to continue.

Parameters

<i>value</i>	Whether the server socket should require client authentication.
--------------	---

Implemented in **decaf::internal::net::ssl::openssl::OpenSSLServerSocket** (p. 2009).

6.500.3.10 `virtual void decaf::net::ssl::SSLServerSocket::setWantClientAuth (bool value) [pure virtual]`

Sets whether or not this **Socket** (p. 2452) will request Client Authentication.

If set to true the **Socket** (p. 2452) (when used in server mode) will request that the client authenticate itself, if the client doesn't send authentication the socket will still

allow negotiation to continue.

Parameters

<i>value</i>	Whether the server socket should request client authentication.
--------------	---

Implemented in **decaf::internal::net::ssl::openssl::OpenSSLServerSocket** (p. 2009).

The documentation for this class was generated from the following file:

- src/main/decaf/net/ssl/**SSLServerSocket.h**

6.501 decaf::net::ssl::SSLServerSocketFactory Class Reference

Factory class interface that provides methods to create SSL Server Sockets.

```
#include <src/main/decaf/net/ssl/SSLServerSocketFactory.h>
```

Inheritance diagram for decaf::net::ssl::SSLServerSocketFactory:

Public Member Functions

- virtual **~SSLServerSocketFactory** ()
- virtual std::vector< std::string > **getDefaultCipherSuites** ()=0
Returns the list of cipher suites which are enabled by default.
- virtual std::vector< std::string > **getSupportedCipherSuites** ()=0
Returns the names of the cipher suites which could be enabled for use on an SSL connection.

Static Public Member Functions

- static **ServerSocketFactory** * **getDefault** ()
*Returns the current default SSL **ServerSocketFactory** (p. 2352), the factory is returned as a pointer however the caller does not own this pointer and should not delete it.*

Protected Member Functions

- **SSLServerSocketFactory** ()

6.501.1 Detailed Description

Factory class interface that provides methods to create SSL Server Sockets.

Since

1.0

6.501.2 Constructor & Destructor Documentation

6.501.2.1 `decaf::net::ssl::SSLServerSocketFactory::SSLServerSocketFactory ()`
[protected]

6.501.2.2 `virtual decaf::net::ssl::SSLServerSocketFactory::~~SSLServerSocketFactory ()` [virtual]

6.501.3 Member Function Documentation

6.501.3.1 `static ServerSocketFactory* decaf::net::ssl::SSLServerSocketFactory::getDefault ()` [static]

Returns the current default SSL **ServerSocketFactory** (p. 2352), the factory is returned as a pointer however the caller does not own this pointer and should not delete it.

This method returns **SSLContext::getDefault()** (p. 2496)->**getServerSocketFactory()**. If that call fails, a non-functional factory is returned.

Returns

the default SSL **ServerSocketFactory** (p. 2352) pointer.

See also

decaf::net::ssl::SSLContext::getDefault() (p. 2496)

Reimplemented from **decaf::net::ServerSocketFactory** (p. 2355).

6.501.3.2 `virtual std::vector<std::string> decaf::net::ssl::SSLServerSocketFactory::getDefaultCipherSuites ()` [pure virtual]

Returns the list of cipher suites which are enabled by default.

Unless a different list is enabled, handshaking on an SSL connection will use one of these cipher suites. The minimum quality of service for these defaults requires confidentiality protection and server authentication (that is, no anonymous cipher suites).

Returns

an STL vector containing the list of cipher suites enabled by default.

See also

getSupportedCipherSuites() (p. 2512)

Implemented in **decaf::internal::net::ssl::openssl::OpenSSLServerSocketFactory** (p. 2014), and **decaf::internal::net::ssl::DefaultSSLServerSocketFactory** (p. 1169).

6.501.3.3 `virtual std::vector<std::string> decaf::net::ssl::SSLServer-
SocketFactory::getSupportedCipherSuites () [pure
virtual]`

Returns the names of the cipher suites which could be enabled for use on an SSL connection.

Normally, only a subset of these will actually be enabled by default, since this list may include cipher suites which do not meet quality of service requirements for those defaults. Such cipher suites are useful in specialized applications.

Returns

an STL vector containing the list of supported cipher suites.

See also

getDefaultCipherSuites() (p. 2512)

Implemented in **decaf::internal::net::ssl::openssl::OpenSSLServerSocketFactory** (p. 2014), and **decaf::internal::net::ssl::DefaultSSLServerSocketFactory** (p. 1170).

The documentation for this class was generated from the following file:

- `src/main/decaf/net/ssl/SSLServerSocketFactory.h`

6.502 decaf::net::ssl::SSLSocket Class Reference

```
#include <src/main/decaf/net/ssl/SSLSocket.h>
```

Inheritance diagram for decaf::net::ssl::SSLSocket:

Public Member Functions

- **SSLSocket ()**

- **SSLSocket** (const **InetAddress** *address, int port)
*Creates a new **SSLSocket** (p. 2513) instance and connects it to the given address and port.*
- **SSLSocket** (const **InetAddress** *address, int port, const **InetAddress** *localAddress, int localPort)
*Creates a new **SSLSocket** (p. 2513) instance and connects it to the given address and port.*
- **SSLSocket** (const std::string &host, int port)
*Creates a new **SSLSocket** (p. 2513) instance and connects it to the given host and port.*
- **SSLSocket** (const std::string &host, int port, const **InetAddress** *localAddress, int localPort)
*Creates a new **SSLSocket** (p. 2513) instance and connects it to the given host and port.*
- virtual ~**SSLSocket** ()
- virtual std::vector< std::string > **getSupportedCipherSuites** () const =0
*Gets a vector containing the names of all the cipher suites that are supported by this **SSLSocket** (p. 2513).*
- virtual std::vector< std::string > **getSupportedProtocols** () const =0
*Gets a vector containing the names of all the protocols that could be enabled for this **SSLSocket** (p. 2513) instance.*
- virtual std::vector< std::string > **getEnabledCipherSuites** () const =0
*Returns a vector containing the names of all the currently enabled Cipher Suites for this **SSL Socket** (p. 2452).*
- virtual void **setEnabledCipherSuites** (const std::vector< std::string > &suites)=0
*Sets the Cipher Suites that are to be enabled on the **SSL Socket** (p. 2452) connection.*
- virtual std::vector< std::string > **getEnabledProtocols** () const =0
*Returns a vector containing the names of all the currently enabled Protocols for this **SSL Socket** (p. 2452).*
- virtual void **setEnabledProtocols** (const std::vector< std::string > &protocols)=0
*Sets the Protocols that are to be enabled on the **SSL Socket** (p. 2452) connection.*
- virtual **SSLParameters** **getSSLParameters** () const
*Returns an **SSLParameters** (p. 2501) object for this **SSLSocket** (p. 2513) instance.*
- virtual void **setSSLParameters** (const **SSLParameters** &value)
*Sets the **SSLParameters** (p. 2501) for this **SSLSocket** (p. 2513) using the supplied **SSLParameters** (p. 2501) instance.*
- virtual void **startHandshake** ()=0
Initiates a handshake for this SSL Connection, this can be necessary for several reasons such as using new encryption keys, or starting a new session.
- virtual void **setUseClientMode** (bool value)=0
Determines the mode that the socket uses when a handshake is initiated, client or server.
- virtual bool **getUseClientMode** () const =0

*Gets whether this **Socket** (p. 2452) is in Client or Server mode, true indicates that the mode is set to Client.*

- virtual void **setNeedClientAuth** (bool value)=0

*Sets the **Socket** (p. 2452) to require that a client authenticate itself by sending a valid Certificate that is trusted by this Server mode socket.*

- virtual bool **getNeedClientAuth** () const =0

Returns if this socket is configured to require client authentication, true means that is has and that clients that failed to authenticate will be rejected.

- virtual void **setWantClientAuth** (bool value)=0

*Sets the **Socket** (p. 2452) to request that a client authenticate itself by sending a valid Certificate that is trusted by this Server mode socket.*

- virtual bool **getWantClientAuth** () const =0

Returns if this socket is configured to request client authentication, true means that is has and that clients that failed to authenticate will be rejected but that clients that do not send a certificate are not considered to have failed authentication.

6.502.1 Detailed Description

Since

1.0

6.502.2 Constructor & Destructor Documentation

6.502.2.1 decaf::net::ssl::SSLSocket::SSLSocket ()

6.502.2.2 decaf::net::ssl::SSLSocket::SSLSocket (const InetAddress * address, int port)

Creates a new **SSLSocket** (p. 2513) instance and connects it to the given address and port.

If the host parameter is empty then the loop back address is used.

Parameters

<i>address</i>	The address to connect to.
<i>port</i>	The port number to connect to [0...65535]

Exceptions

UnknownHostException (p. 2816)	if the host cannot be resolved.
<i>IOException</i>	if an I/O error occurs while connecting the Socket (p. 2452).
<i>NullPointerException</i>	if the InetAddress (p. 1437) instance is NULL.
<i>IllegalArgumentException</i>	if the port is not in range [0...65535]

6.502.2.3 `decaf::net::ssl::SSLSocket::SSLSocket (const InetAddress * address, int port, const InetAddress * localAddress, int localPort)`

Creates a new **SSLSocket** (p. 2513) instance and connects it to the given address and port.

The **Socket** (p. 2452) will also bind to the local address and port specified.

Parameters

<i>address</i>	The address to connect to.
<i>port</i>	The port number to connect to [0...65535]
<i>localAddress</i>	The IP address on the local machine to bind to.
<i>localPort</i>	The port on the local machine to bind to.

Exceptions

UnknownHostException (p. 2816)	if the host cannot be resolved.
<i>IOException</i>	if an I/O error occurs while connecting the Socket (p. 2452).
<i>NullPointerException</i>	if the InetAddress (p. 1437) instance is NULL.
<i>IllegalArgumentException</i>	if the port is not in range [0...65535]

6.502.2.4 `decaf::net::ssl::SSLSocket::SSLSocket (const std::string & host, int port)`

Creates a new **SSLSocket** (p. 2513) instance and connects it to the given host and port.

If the host parameter is empty then the loop back address is used.

Parameters

<i>host</i>	The host name or IP address to connect to, empty string means loop-back.
<i>port</i>	The port number to connect to [0...65535]

Exceptions

UnknownHostException (p. 2816)	if the host cannot be resolved.
<i>IOException</i>	if an I/O error occurs while connecting the Socket (p. 2452).
<i>IllegalArgumentException</i>	if the port is not in range [0...65535]

6.502.2.5 **decaf::net::ssl::SSLSocket::SSLSocket** (const std::string & *host*, int *port*, const InetAddress * *localAddress*, int *localPort*)

Creates a new **SSLSocket** (p. 2513) instance and connects it to the given host and port. If the host parameter is empty then the loop back address is used.

Parameters

<i>host</i>	The host name or IP address to connect to, empty string means loop-back.
<i>port</i>	The port number to connect to [0...65535]
<i>localAddress</i>	The IP address on the local machine to bind to.
<i>localPort</i>	The port on the local machine to bind to.

Exceptions

UnknownHostException (p. 2816)	if the host cannot be resolved.
<i>IOException</i>	if an I/O error occurs while connecting the Socket (p. 2452).
<i>IllegalArgument-Exception</i>	if the port is not in range [0...65535]

6.502.2.6 virtual **decaf::net::ssl::SSLSocket::~~SSLSocket** () [virtual]

6.502.3 Member Function Documentation

6.502.3.1 virtual std::vector<std::string> **decaf::net::ssl::SSL-Socket::getEnabledCipherSuites** () const [pure virtual]

Returns a vector containing the names of all the currently enabled Cipher Suites for this SSL **Socket** (p. 2452).

Returns

vector of the names of all enabled Cipher Suites.

Implemented in **decaf::internal::net::ssl::openssl::OpenSSLSocket** (p. 2021).

6.502.3.2 virtual std::vector<std::string> **decaf::net::ssl::SSL-Socket::getEnabledProtocols** () const [pure virtual]

Returns a vector containing the names of all the currently enabled Protocols for this SSL **Socket** (p. 2452).

Returns

vector of the names of all enabled Protocols.

Implemented in **decaf::internal::net::ssl::openssl::OpenSSLSocket** (p. 2021).

6.502.3.3 `virtual bool decaf::net::ssl::SSLSocket::getNeedClientAuth () const`
`[pure virtual]`

Returns if this socket is configured to require client authentication, true means that is has and that clients that failed to authenticate will be rejected.

This option is only useful when the socket is operating in server mode.

Returns

true if client authentication is required.

Implemented in **decaf::internal::net::ssl::openssl::OpenSSLSocket** (p. 2022).

6.502.3.4 `virtual SSLParameters decaf::net::ssl::SSLSocket::getSSLParameters () const`
`[virtual]`

Returns an **SSLParameters** (p. 2501) object for this **SSLSocket** (p. 2513) instance.

The cipherSuites and protocols vectors in the returned **SSLParameters** (p. 2501) reference will never be empty.

Returns

an **SSLParameters** (p. 2501) object with the settings in use for the **SSLSocket** (p. 2513).

6.502.3.5 `virtual std::vector<std::string> decaf::net::ssl::SSL-`
`Socket::getSupportedCipherSuites () const` `[pure`
`virtual]`

Gets a vector containing the names of all the cipher suites that are supported by this **SSLSocket** (p. 2513).

Normally not all of these cipher suites will be enabled on the **Socket** (p. 2452).

Returns

a vector containing the names of all the supported cipher suites.

Implemented in **decaf::internal::net::ssl::openssl::OpenSSLSocket** (p. 2023).

6.502.3.6 `virtual std::vector<std::string> decaf::net::ssl::SSL-
Socket::getSupportedProtocols () const [pure
virtual]`

Gets a vector containing the names of all the protocols that could be enabled for this **SSLSocket** (p. 2513) instance.

Returns

a vector containing the names of all the supported protocols.

Implemented in **decaf::internal::net::ssl::openssl::OpenSSLSocket** (p. 2023).

6.502.3.7 `virtual bool decaf::net::ssl::SSLSocket::getUseClientMode () const
[pure virtual]`

Gets whether this **Socket** (p. 2452) is in Client or Server mode, true indicates that the mode is set to Client.

Returns

true if the **Socket** (p. 2452) is in Client mode, false otherwise.

Implemented in **decaf::internal::net::ssl::openssl::OpenSSLSocket** (p. 2023).

6.502.3.8 `virtual bool decaf::net::ssl::SSLSocket::getWantClientAuth () const
[pure virtual]`

Returns if this socket is configured to request client authentication, true means that is has and that clients that failed to authenticate will be rejected but that clients that do not send a certificate are not considered to have failed authentication.

This option is only useful when the socket is operating in server mode.

Returns

true if client authentication is required.

Implemented in **decaf::internal::net::ssl::openssl::OpenSSLSocket** (p. 2024).

6.502.3.9 `virtual void decaf::net::ssl::SSLSocket::setEnabledCipherSuites (const
std::vector< std::string > & suites) [pure virtual]`

Sets the Cipher Suites that are to be enabled on the SSL **Socket** (p. 2452) connection.

Each of the named Cipher Suites must appear in the list of supported cipher suites for this connection or an exception will be thrown.

Parameters

<i>suites</i>	An Vector of names for all the Cipher Suites that are to be enabled.
---------------	--

Exceptions

<i>IllegalArgument-Exception</i>	if the vector is empty or one of the names is invalid.
----------------------------------	--

Implemented in **decaf::internal::net::ssl::openssl::OpenSSLSocket** (p. 2025).

6.502.3.10 `virtual void decaf::net::ssl::SSLSocket::setEnabledProtocols (const std::vector< std::string > & protocols) [pure virtual]`

Sets the Protocols that are to be enabled on the SSL **Socket** (p. 2452) connection.

Each of the named Protocols must appear in the list of supported protocols suites for this connection or an exception will be thrown.

Parameters

<i>protocols</i>	An Vector of names for all the Protocols that are to be enabled.
------------------	--

Exceptions

<i>IllegalArgument-Exception</i>	if the vector is empty or one of the names is invalid.
----------------------------------	--

Implemented in **decaf::internal::net::ssl::openssl::OpenSSLSocket** (p. 2025).

6.502.3.11 `virtual void decaf::net::ssl::SSLSocket::setNeedClientAuth (bool value) [pure virtual]`

Sets the **Socket** (p. 2452) to require that a client authenticate itself by sending a valid Certificate that is trusted by this Server mode socket.

This option only applies to sockets in the Server mode.

If the option is enabled an the client does not provide a certificate then the handshake is considered failed and the connection is refused. Calling this method resets any previous value for this option as well as clears any value set in the setWantClientAuth method.

Parameters

<i>value</i>	The value indicating if a client is required to authenticate itself or not.
--------------	---

Implemented in **decaf::internal::net::ssl::openssl::OpenSSLSocket** (p. 2026).

6.502.3.12 virtual void decaf::net::ssl::SSLSocket::setSSLParameters (const
SSLParameters & value) [virtual]

Sets the **SSLParameters** (p. 2501) for this **SSLSocket** (p. 2513) using the supplied **SSLParameters** (p. 2501) instance.

If the cipherSuites vector in the **SSLParameters** (p. 2501) instance is not empty then the setEnabledCipherSuites method is called with that vector, if the protocols vector in the **SSLParameters** (p. 2501) instance is not empty then the setEnabledProtocols method is called with that vector. If the needClientAuth value or the wantClientAuth value is true then the setNeedClientAuth and setWantClientAuth methods are called respectively with a value of true, otherwise the setWantClientAuth method is called with a value of false.

Parameters

value	The SSLParameters (p. 2501) instance that is used to update this SSLSocket (p. 2513)'s settings.
-------	--

Exceptions

<i>IllegalArgument-Exception</i>	if an error occurs while calling setEnabledCipherSuites or setEnabledProtocols.
----------------------------------	---

6.502.3.13 virtual void decaf::net::ssl::SSLSocket::setUseClientMode (bool value)
[pure virtual]

Determines the mode that the socket uses when a handshake is initiated, client or server.

This method must be called prior to any handshake attempts on this **Socket** (p. 2452), once a handshake has been initiated this socket remains in the set mode; client or server, for the life of this object.

Parameters

value	The mode setting, true for client or false for server.
-------	--

Exceptions

<i>IllegalArgument-Exception</i>	if the handshake process has begun and mode is locked.
----------------------------------	--

Implemented in **decaf::internal::net::ssl::openssl::OpenSSLSocket** (p. 2026).

6.502.3.14 `virtual void decaf::net::ssl::SSLSocket::setWantClientAuth (bool value)`
`[pure virtual]`

Sets the **Socket** (p.2452) to request that a client authenticate itself by sending a valid Certificate that is trusted by this Server mode socket.

This option only applies to sockets in the Server mode.

If the option is enabled and the client does not provide a certificate then the handshake is considered to have succeeded, if it does send a certificate and that certificate is invalid the the handshake will fail. Calling this method resets any previous value for this option as well as clears any value set in the `setNeedClientAuth` method.

Parameters

<i>value</i>	The value indicating if a client is requested to authenticate itself or not.
--------------	--

Implemented in `decaf::internal::net::ssl::openssl::OpenSSLSocket` (p.2027).

6.502.3.15 `virtual void decaf::net::ssl::SSLSocket::startHandshake ()` `[pure virtual]`

Initiates a handshake for this SSL Connection, this can be necessary for several reasons such as using new encryption keys, or starting a new session.

When called for the first time after the socket connects this method blocks until the handshake is completed. The provider is not required to support multiple handshakes and can throw an `IOException` to indicate an error.

Exceptions

<i>IOException</i>	if an I/O error occurs while performing the Handshake
--------------------	---

Implemented in `decaf::internal::net::ssl::openssl::OpenSSLSocket` (p.2028).

The documentation for this class was generated from the following file:

- `src/main/decaf/net/ssl/SSLSocket.h`

6.503 decaf::net::ssl::SSLSocketFactory Class Reference

Factory class interface for a **SocketFactory** (p.2473) that can create **SSLSocket** (p.2513) objects.

```
#include <src/main/decaf/net/ssl/SSLSocketFactory.h>
```

Inheritance diagram for `decaf::net::ssl::SSLSocketFactory`:

Public Member Functions

- virtual `~SSLSocketFactory()`
- virtual `std::vector< std::string > getDefaultCipherSuites()`
Returns the list of cipher suites which are enabled by default.
- virtual `std::vector< std::string > getSupportedCipherSuites()`
Returns the names of the cipher suites which could be enabled for use on an SSL connection.
- virtual `Socket * createSocket (Socket *socket, std::string host, int port, bool autoClose)=0`
Returns a socket layered over an existing socket connected to the named host, at the given port.

Static Public Member Functions

- static `SocketFactory * getDefault()`
*Returns the current default SSL **SocketFactory** (p. 2473), the factory is returned as a pointer however the caller does not own this pointer and should not delete it.*

Protected Member Functions

- `SSLSocketFactory()`

6.503.1 Detailed Description

Factory class interface for a **SocketFactory** (p.2473) that can create **SSLSocket** (p.2513) objects.

Since

1.0

6.503.2 Constructor & Destructor Documentation

6.503.2.1 `decaf::net::ssl::SSLSocketFactory::SSLSocketFactory ()`
[protected]

6.503.2.2 `virtual decaf::net::ssl::SSLSocketFactory::~~SSLSocketFactory ()`
[virtual]

6.503.3 Member Function Documentation

6.503.3.1 **virtual Socket*** **decaf::net::ssl::SSLSocketFactory::createSocket**
(Socket * socket, std::string host, int port, bool autoClose) [pure
 virtual]

Returns a socket layered over an existing socket connected to the named host, at the given port.

This constructor can be used when tunneling SSL through a proxy or when negotiating the use of SSL over an existing socket. The host and port refer to the logical peer destination. This socket is configured using the socket options established for this factory.

Parameters

<i>socket</i>	The existing socket to layer over.
<i>host</i>	The server host the original Socket (p. 2452) is connected to.
<i>port</i>	The server port the original Socket (p. 2452) is connected to.
<i>autoClose</i>	Should the layered over Socket (p. 2452) be closed when the topmost socket is closed.

Returns

a new **Socket** (p. 2452) instance that wraps the given **Socket** (p. 2452).

Exceptions

<i>IOException</i>	if an I/O exception occurs while performing this operation.
UnknownHostException (p. 2816)	if the host is unknown.

Implemented in **decaf::internal::net::ssl::openssl::OpenSSLSocketFactory** (p. 2038), and **decaf::internal::net::ssl::DefaultSSLSocketFactory** (p. 1176).

6.503.3.2 **static SocketFactory*** **decaf::net::ssl::SSLSocketFactory::getDefault ()**
 [static]

Returns the current default SSL **SocketFactory** (p. 2473), the factory is returned as a pointer however the caller does not own this pointer and should not delete it.

This method returns **SSLContext::getDefault()** (p. 2496)->getSocketFactory(). If that call fails, a non-functional factory is returned.

Returns

the default SSL **SocketFactory** (p. 2473) pointer.

See also

decaf::net::ssl::SSLContext::getDefault() (p. 2496)

Reimplemented from **decaf::net::SocketFactory** (p. 2478).

6.503.3.3 `virtual std::vector<std::string> decaf::net::ssl::SSL-
SocketFactory::getDefaultCipherSuites () [pure
virtual]`

Returns the list of cipher suites which are enabled by default.

Unless a different list is enabled, handshaking on an SSL connection will use one of these cipher suites. The minimum quality of service for these defaults requires confidentiality protection and server authentication (that is, no anonymous cipher suites).

Returns

an STL vector containing the list of cipher suites enabled by default.

See also

getSupportedCipherSuites() (p. 2524)

Implemented in **decaf::internal::net::ssl::openssl::OpenSSLSocketFactory** (p. 2039), and **decaf::internal::net::ssl::DefaultSSLSocketFactory** (p. 1177).

6.503.3.4 `virtual std::vector<std::string> decaf::net::ssl::SSLSocket-
Factory::getSupportedCipherSuites () [pure
virtual]`

Returns the names of the cipher suites which could be enabled for use on an SSL connection.

Normally, only a subset of these will actually be enabled by default, since this list may include cipher suites which do not meet quality of service requirements for those defaults. Such cipher suites are useful in specialized applications.

Returns

an STL vector containing the list of supported cipher suites.

See also

getDefaultCipherSuites() (p. 2524)

Implemented in **decaf::internal::net::ssl::openssl::OpenSSLSocketFactory** (p. 2040), and **decaf::internal::net::ssl::DefaultSSLSocketFactory** (p. 1177).

The documentation for this class was generated from the following file:

- `src/main/decaf/net/ssl/SSLSocketFactory.h`

6.504 activemq::transport::tcp::SslTransport Class Reference

Transport (p. 2790) for connecting to a Broker using an SSL Socket.

```
#include <src/main/activemq/transport/tcp/SslTransport.h>
```

Inheritance diagram for activemq::transport::tcp::SslTransport:

Public Member Functions

- **SslTransport** (const **Pointer**< **Transport** > &next)
*Creates a new instance of the **SslTransport** (p. 2525), the transport will not attempt to connect to a remote host until the connect method is called.*
- virtual ~**SslTransport** ()

Protected Member Functions

- virtual **decaf::net::Socket** * **createSocket** ()
Create an unconnected Socket instance to be used by the transport to communicate with the broker.
Returns

a newly created unconnected Socket instance.

Exceptions

IOException	<i>if there is an error while creating the unconnected Socket.</i>
--------------------	--

- virtual void **configureSocket** (**decaf::net::Socket** *socket, **decaf::util::Properties** &properties)

6.504.1 Detailed Description

Transport (p. 2790) for connecting to a Broker using an SSL Socket.

This transport simply wraps the **TcpTransport** (p. 2693) and provides the **TcpTransport** (p. 2693) an SSL based Socket pointer allowing the core **TcpTransport** (p. 2693) logic to be reused.

Since

3.2.0

6.504.2 Constructor & Destructor Documentation

6.504.2.1 `activemq::transport::tcp::SslTransport::SslTransport (const Pointer< Transport > & next)`

Creates a new instance of the **SslTransport** (p. 2525), the transport will not attempt to connect to a remote host until the connect method is called.

Parameters

<i>next</i>	the next transport in the chain
-------------	---------------------------------

6.504.2.2 `virtual activemq::transport::tcp::SslTransport::~SslTransport ()`
[virtual]

6.504.3 Member Function Documentation

6.504.3.1 `virtual void activemq::transport::tcp::SslTransport::configureSocket (decaf::net::Socket * socket, decaf::util::Properties & properties)`
[protected, virtual]

6.504.3.2 `virtual decaf::net::Socket* activemq::transport::tcp::SslTransport::createSocket ()` [protected, virtual]

Create an unconnected Socket instance to be used by the transport to communicate with the broker.

Returns

a newly created unconnected Socket instance.

Exceptions

<i>IOException</i>	if there is an error while creating the unconnected Socket.
--------------------	---

Reimplemented from **activemq::transport::tcp::TcpTransport** (p. 2695).

The documentation for this class was generated from the following file:

- `src/main/activemq/transport/tcp/SslTransport.h`

6.505 activemq::transport::tcp::SslTransportFactory Class - Reference

```
#include <src/main/activemq/transport/tcp/SslTransport-Factory.h>
```

Inheritance diagram for `activemq::transport::tcp::SslTransportFactory`:

Public Member Functions

- virtual `~SslTransportFactory()`

Protected Member Functions

- virtual `Pointer<Transport> doCreateComposite` (const `decaf::net::URI` &location, const `Pointer<wireformat::WireFormat>` &wireFormat, const `decaf::util::Properties` &properties)

6.505.1 Constructor & Destructor Documentation

- 6.505.1.1 virtual `activemq::transport::tcp::SslTransportFactory::~~SslTransportFactory()` [virtual]

6.505.2 Member Function Documentation

- 6.505.2.1 virtual `Pointer<Transport> activemq::transport::tcp::SslTransportFactory::doCreateComposite` (const `decaf::net::URI` & location, const `Pointer< wireformat::WireFormat >` & wireFormat, const `decaf::util::Properties` & properties) [protected, virtual]

Reimplemented from `activemq::transport::tcp::TcpTransportFactory` (p. 2698).

The documentation for this class was generated from the following file:

- `src/main/activemq/transport/tcp/SslTransportFactory.h`

6.506 `activemq::commands::BrokerError::StackTraceElement` - Struct Reference

```
#include <src/main/activemq/commands/BrokerError.h>
```

Public Member Functions

- `StackTraceElement()`

Data Fields

- std::string **ClassName**
- std::string **FileName**
- std::string **MethodName**
- int **LineNumber**

6.506.1 Constructor & Destructor Documentation

6.506.1.1 `activemq::commands::BrokerError::StackTraceElement::Stack-
TraceElement() [inline]`

6.506.2 Field Documentation

6.506.2.1 `std::string activemq::commands::BrokerError::StackTraceElement::-
ClassName`

6.506.2.2 `std::string activemq::commands::BrokerError::StackTraceElement::File-
Name`

6.506.2.3 `int activemq::commands::BrokerError::StackTraceElement::Line-
Number`

6.506.2.4 `std::string activemq::commands::BrokerError::StackTraceElement::-
MethodName`

The documentation for this struct was generated from the following file:

- `src/main/activemq/commands/BrokerError.h`

6.507 decaf::internal::io::StandardErrorOutputStream Class - Reference

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

```
#include <src/main/decaf/internal/io/StandardErrorOutput-  
Stream.h>
```

Inheritance diagram for `decaf::internal::io::StandardErrorOutputStream`:

Public Member Functions

- `StandardErrorOutputStream ()`
- `virtual ~StandardErrorOutputStream ()`

- virtual void **flush** ()

Flushes this stream by writing any buffered output to the underlying stream.

Exceptions

IOException (p. 1545)	if an I/O error occurs.
------------------------------	-------------------------

The default implementation of this method does nothing.

- virtual void **close** ()

Closes this object and deallocates the appropriate resources.

The object is generally no longer usable after calling close.

Exceptions

IOException (p. 1545)	if an error occurs while closing.
------------------------------	-----------------------------------

The default implementation of this method does nothing.

Protected Member Functions

- virtual void **doWriteByte** (unsigned char value)
- virtual void **doWriteArrayBounded** (const unsigned char *buffer, int size, int offset, int length)

6.507.1 Detailed Description

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

This allows for the use of alternate output methods on platforms or compilers that do not support `std::cerr`.

6.507.2 Constructor & Destructor Documentation

6.507.2.1 **decaf::internal::io::StandardErrorOutputStream::StandardErrorOutputStream ()**

6.507.2.2 **virtual decaf::internal::io::StandardErrorOutputStream::~~StandardErrorOutputStream ()** [virtual]

6.507.3 Member Function Documentation

6.507.3.1 **virtual void decaf::internal::io::StandardErrorOutputStream::close ()**
[virtual]

Closes this object and deallocates the appropriate resources.

The object is generally no longer usable after calling close.

Exceptions

IOException (p. 1545)	if an error occurs while closing.
---------------------------------	-----------------------------------

The default implementation of this method does nothing.

The default implementation of this method does nothing.

Reimplemented from **decaf::io::OutputStream** (p. 2068).

6.507.3.2 virtual void decaf::internal::io::StandardErrorOutputStream::doWriteArrayBounded (const unsigned char * *buffer*, int *size*, int *offset*, int *length*)
[protected, virtual]

Reimplemented from **decaf::io::OutputStream** (p. 2068).

6.507.3.3 virtual void decaf::internal::io::StandardErrorOutputStream::doWriteByte (unsigned char *value*) [protected, virtual]

Implements **decaf::io::OutputStream** (p. 2069).

6.507.3.4 virtual void decaf::internal::io::StandardErrorOutputStream::flush ()
[virtual]

Flushes this stream by writing any buffered output to the underlying stream.

Exceptions

<i>IOException</i> (p. 1545)	if an I/O error occurs.
--	-------------------------

The default implementation of this method does nothing.

The default implementation of this method does nothing.

Reimplemented from **decaf::io::OutputStream** (p. 2069).

The documentation for this class was generated from the following file:

- src/main/decaf/internal/io/**StandardErrorOutputStream.h**

6.508 decaf::internal::io::StandardInputStream Class Reference

```
#include <src/main/decaf/internal/io/StandardInputStream.h>
```

Inheritance diagram for decaf::internal::io::StandardInputStream:

Public Member Functions

- **StandardInputStream** ()

- virtual `~StandardInputStream ()`
- virtual int **available** () const
Indicates the number of bytes available.

Protected Member Functions

- virtual int **doReadByte** ()

6.508.1 Constructor & Destructor Documentation

6.508.1.1 `decaf::internal::io::StandardInputStream::StandardInputStream ()`

6.508.1.2 `virtual decaf::internal::io::StandardInputStream::~~StandardInputStream () [virtual]`

6.508.2 Member Function Documentation

6.508.2.1 `virtual int decaf::internal::io::StandardInputStream::available () const [virtual]`

Indicates the number of bytes available.

The default implementation of this methods returns 0. Classes that override this method may return the total number of bytes that are currently available for reading and others may simply return a value of one indicating that there is some data available. The caller should view the result of this method as an absolute.

The default implementation of this method returns zero.

Returns

the number of bytes available on this input stream.

Exceptions

<i>IOException</i> if an I/O error occurs.
--

Reimplemented from `decaf::io::InputStream` (p. 1466).

6.508.2.2 `virtual int decaf::internal::io::StandardInputStream::doReadByte () [protected, virtual]`

Implements `decaf::io::InputStream` (p. 1467).

The documentation for this class was generated from the following file:

- `src/main/decaf/internal/io/StandardInputStream.h`

6.509 decaf::internal::io::StandardOutputStream Class Reference

```
#include <src/main/decaf/internal/io/StandardOutputStream.h>
```

Inheritance diagram for decaf::internal::io::StandardOutputStream:

Public Member Functions

- **StandardOutputStream** ()
- virtual **~StandardOutputStream** ()
- virtual void **flush** ()

Flushes this stream by writing any buffered output to the underlying stream.

Exceptions

IOException (p. 1545)	<i>if an I/O error occurs.</i>
------------------------------	--------------------------------

The default implementation of this method does nothing.

- virtual void **close** ()

Closes this object and deallocates the appropriate resources.

The object is generally no longer usable after calling close.

Exceptions

IOException (p. 1545)	<i>if an error occurs while closing.</i>
------------------------------	--

The default implementation of this method does nothing.

Protected Member Functions

- virtual void **doWriteByte** (unsigned char value)
- virtual void **doWriteArrayBounded** (const unsigned char *buffer, int size, int offset, int length)

6.509.1 Constructor & Destructor Documentation

6.509.1.1 **decaf::internal::io::StandardOutputStream::StandardOutputStream** ()

6.509.1.2 **virtual decaf::internal::io::StandardOutputStream::~~StandardOutputStream** () [virtual]

6.509.2 Member Function Documentation

6.509.2.1 **virtual void decaf::internal::io::StandardOutputStream::close** () [virtual]

Closes this object and deallocates the appropriate resources.

The object is generally no longer usable after calling close.

Exceptions

<i>IOException</i> (p. 1545)	if an error occurs while closing.
--	-----------------------------------

The default implementation of this method does nothing.

The default implementation of this method does nothing.

Reimplemented from **decaf::io::OutputStream** (p. 2068).

6.509.2.2 virtual void **decaf::internal::io::StandardOutputStream::doWriteArrayBounded** (const unsigned char * *buffer*, int *size*, int *offset*, int *length*)
[protected, virtual]

Reimplemented from **decaf::io::OutputStream** (p. 2068).

6.509.2.3 virtual void **decaf::internal::io::StandardOutputStream::doWriteByte** (unsigned char *value*) [protected, virtual]

Implements **decaf::io::OutputStream** (p. 2069).

6.509.2.4 virtual void **decaf::internal::io::StandardOutputStream::flush** ()
[virtual]

Flushes this stream by writing any buffered output to the underlying stream.

Exceptions

<i>IOException</i> (p. 1545)	if an I/O error occurs.
--	-------------------------

The default implementation of this method does nothing.

The default implementation of this method does nothing.

Reimplemented from **decaf::io::OutputStream** (p. 2069).

The documentation for this class was generated from the following file:

- src/main/decaf/internal/io/**StandardOutputStream.h**

6.510 cms::Startable Class Reference

Interface for a class that implements the start method.

```
#include <src/main/cms/Startable.h>
```

Inheritance diagram for cms::Startable:

Public Member Functions

- virtual **~Startable** () throw ()
- virtual void **start** ()=0

Starts the service.

6.510.1 Detailed Description

Interface for a class that implements the start method.

An object that implements the **Startable** (p.2534) interface implies that until its start method is called it will be considered to be in a closed or stopped state and will throw an Exception to indicate that it is not in an started state if one of its methods is called.

Since

1.0

6.510.2 Constructor & Destructor Documentation

6.510.2.1 virtual cms::Startable::~**~Startable** () throw () [virtual]

6.510.3 Member Function Documentation

6.510.3.1 virtual void cms::Startable::**start** () [pure virtual]

Starts the service.

Exceptions

CMSException (p. 826)	if an internal error occurs while starting.
---------------------------------	---

Implemented in **activemq::core::ActiveMQConnection** (p.211), **activemq::core::ActiveMQSession** (p.352), **activemq::core::ActiveMQConsumer** (p.244), **activemq::cmsutil::PooledSession** (p.2106), and **activemq::cmsutil::CachedConsumer** (p.738).

Referenced by activemq::cmsutil::PooledSession::start().

The documentation for this class was generated from the following file:

- src/main/cms/**Startable.h**

6.511 decaf::lang::STATIC_CAST_TOKEN Struct Reference

```
#include <src/main/decaf/lang/Pointer.h>
```

The documentation for this struct was generated from the following file:

- src/main/decaf/lang/**Pointer.h**

6.512 activemq::core::ActiveMQConstants::StaticInitializer Class - Reference

```
#include <src/main/activemq/core/ActiveMQConstants.h>
```

Public Member Functions

- **StaticInitializer** ()
- virtual **~StaticInitializer** ()

Static Public Attributes

- static std::string **destOptions** [NUM_OPTIONS]
- static std::string **uriParams** [NUM_PARAMS]
- static std::map< std::string, **DestinationOption** > **destOptionMap**
- static std::map< std::string, **URIParam** > **uriParamsMap**

6.512.1 Constructor & Destructor Documentation

6.512.1.1 **activemq::core::ActiveMQConstants::StaticInitializer::StaticInitializer** ()

6.512.1.2 **virtual activemq::core::ActiveMQConstants::StaticInitializer::~~StaticInitializer** () [inline, virtual]

6.512.2 Field Documentation

6.512.2.1 **std::map<std::string, DestinationOption> activemq::core::ActiveMQConstants::StaticInitializer::destOptionMap** [static]

6.512.2.2 **std::string activemq::core::ActiveMQConstants::StaticInitializer::destOptions[NUM_OPTIONS]** [static]

6.512.2.3 `std::string activemq::core::ActiveMQConstants::StaticInitializer::uriParams[NUM_PARAMS]` `[static]`

6.512.2.4 `std::map<std::string, URIParam> activemq::core::ActiveMQConstants::StaticInitializer::uriParamsMap` `[static]`

The documentation for this class was generated from the following file:

- `src/main/activemq/core/ActiveMQConstants.h`

6.513 decaf::util::StlList< E > Class Template Reference

List (p. 1658) class that wraps the STL list object to provide a simpler interface and additional methods not provided by the STL type.

```
#include <src/main/decaf/util/StlList.h>
```

Inheritance diagram for `decaf::util::StlList< E >`:

Data Structures

- class **ConstStlListIterator**
- class **StlListIterator**

Public Member Functions

- **StlList** ()
Default constructor - does nothing.
- **StlList** (const **StlList** &source)
Copy constructor - copies the content of the given set into this one.
- **StlList** (const **Collection**< E > &source)
Copy constructor - copies the content of the given set into this one.
- virtual bool **equals** (const **Collection**< E > &collection) const
*Answers true if this **Collection** (p. 851) and the one given are the same size and if each element contained in the **Collection** (p. 851) given is equal to an element contained in this collection.*

Parameters

collection	- The Collection (p. 851) to be compared to this one.
------------	--

Returns

*true if this **Collection** (p. 851) is equal to the one given.*

- virtual void **copy** (const **Collection**< E > &collection)

*Renders this **Collection** (p. 851) as a Copy of the given **Collection** (p. 851).*

*The default implementation iterates over the contents of the given collection adding each to this collection after first calling this **Collection** (p. 851)'s clear method.*

Parameters

collection	<i>The collection to mirror.</i>
------------	----------------------------------

Exceptions

UnsupportedOperation-Exception	<i>if this is an unmodifiable collection.</i>
IllegalStateException	<i>if the elements cannot be added at this time due to insertion restrictions.</i>

- virtual **Iterator**< E > * **iterator** ()

Returns

an iterator over a set of elements of type T.

- virtual **Iterator**< E > * **iterator** () const
- virtual **ListIterator**< E > * **listIterator** ()

Returns

a list iterator over the elements in this list (in proper sequence).

- virtual **ListIterator**< E > * **listIterator** () const
- virtual **ListIterator**< E > * **listIterator** (int index)

Parameters

index	<i>index of first element to be returned from the list iterator (by a call to the next method).</i>
-------	---

Returns

a list iterator of the elements in this list (in proper sequence), starting at the specified position in this list. The specified index indicates the first element that would be returned by an initial call to next. An initial call to previous would return the element with the specified index minus one.

Exceptions

IndexOutOfBoundsException	<i>if the index is out of range ($index < 0 \parallel index > \mathbf{size}()$ (p. 864))</i>
---------------------------	---

- virtual **ListIterator**< E > * **listIterator** (int index) const
- virtual void **clear** ()

Removes all of the elements from this collection (optional operation).

The collection will be empty after this method returns.

*This implementation iterates over this collection, removing each element using the - **Iterator.remove** (p. 1560) operation. Most implementations will probably choose to override this method for efficiency.*

Note that this implementation will throw an UnsupportedOperationException if the iterator returned by this collection's iterator method does not implement the remove method and this collection is non-empty.

Exceptions

UnsupportedOperation-Exception	<i>if the clear operation is not supported by this collection</i>
--------------------------------	---

- virtual bool **isEmpty** () const

Returns true if this collection contains no elements.

*This implementation returns **size()** (p. 864) == 0.*

Returns

true if the size method return 0.

- virtual int **size** () const

Returns the number of elements in this collection.

If this collection contains more than Integer::MAX_VALUE elements, returns Integer::MAX_VALUE.

Returns

the number of elements in this collection

- virtual E **get** (int index) const

Gets the element contained at position passed.

Parameters

index	<i>The position to get.</i>
-------	-----------------------------

Returns

value at index specified.

Exceptions

IndexOutOfBounds-Exception	<i>if the index given is less than zero or greater than the List (p. 1658) size.</i>
----------------------------	---

- virtual E **set** (int index, const E &element)

Replaces the element at the specified position in this list with the specified element.

Parameters

index	<i>The index of the element to replace.</i>
element	<i>The element to be stored at the specified position.</i>

Returns

the element previously at the specified position.

Exceptions

IndexOutOfBounds-Exception	<i>if the index given is less than zero or greater than the List (p. 1658) size.</i>
UnsupportedOperation-Exception	<i>if this is an unmodifiable collection.</i>
NullPointerException	<i>if the Collection (p. 851) is a container of pointers and does not allow NULL values.</i>
IllegalArgumentException	<i>if some property of the element prevents it from being added to this collection</i>
IllegalStateException	<i>if the element cannot be added at this time due to insertion restrictions.</i>

- virtual void **add** (int index, const E &element)

Inserts the specified element at the specified position in this list.

- virtual bool **add** (const E &value)

Returns true if this collection changed as a result of the call.

- virtual bool **addAll** (const **Collection**< E > &collection)

Adds all of the elements in the specified collection to this collection.

The behavior of this operation is undefined if the specified collection is modified while the operation is in progress. (This implies that the behavior of this call is undefined if the specified collection is this collection, and this collection is nonempty.)

Parameters

collection	The Collection (p. 851) whose elements are added to this one.
------------	--

Returns

true if this collection changed as a result of the call

Exceptions

UnsupportedOperationException	<i>if this is an unmodifiable collection.</i>
NullPointerException	<i>if the Collection (p. 851) is a container of pointers and does not allow NULL values.</i>
IllegalArgumentException	<i>if some property of an element prevents it from being added to this collection</i>
IllegalStateException	<i>if an element cannot be added at this time due to insertion restrictions.</i>

This implementation iterates over the specified collection, and adds each object returned by the iterator to this collection, in turn.

Note that this implementation will throw an UnsupportedOperationException unless add is overridden (assuming the specified collection is non-empty).

- virtual bool **addAll** (int index, const **Collection**< E > &collection)

Inserts all of the elements in the specified collection into this list at the specified position (optional operation).

Shifts the element currently at that position (if any) and any subsequent elements to the right (increases their indices). The new elements will appear in this list in the order that they are returned by the specified collection's iterator. The behavior of this operation is undefined if the specified collection is modified while the operation is in progress. (Note that this will occur if the specified collection is this list, and it's nonempty.)

Parameters

index	The index at which to insert the first element from the specified collection
source	The Collection (p. 851) containing elements to be added to this list

Returns

true if this list changed as a result of the call

Exceptions

IndexOutOfBoundsException	<i>if the index given is less than zero or greater than the List (p. 1658) size.</i>
UnsupportedOperationException	<i>if this is an unmodifiable collection.</i>
NullPointerException	<i>if the Collection (p. 851) is a container of pointers and does not allow NULL values.</i>
IllegalArgumentException	<i>if some property of the element prevents it from being added to this collection</i>
IllegalStateException	<i>if the element cannot be added at this time due to insertion restrictions.</i>

- virtual bool **remove** (const E &value)

Removes a single instance of the specified element from the collection.

More formally, removes an element e such that $(value == NULL ? e == NULL : value == e)$, if this collection contains one or more such elements. Returns true if this collection contained the specified element (or equivalently, if this collection changed as a result of the call).

Parameters

value	<i>The reference to the element to remove from this Collection (p. 851).</i>
-------	---

Returns

true if the collection was changed, false otherwise.

Exceptions

UnsupportedOperationException	<i>if this is an unmodifiable collection.</i>
NullPointerException	<i>if the Collection (p. 851) is a container of pointers and does not allow NULL values.</i>

This implementation iterates over the collection looking for the specified element. - If it finds the element, it removes the element from the collection using the iterator's remove method.

Note that this implementation throws an UnsupportedOperationException if the iterator returned by this collection's iterator method does not implement the remove method and this collection contains the specified object.

- virtual E **removeAt** (int index)

Removes the element at the specified position in this list.

Shifts any subsequent elements to the left (subtracts one from their indices). Returns the element that was removed from the list.

Parameters

index	<i>- the index of the element to be removed.</i>
-------	--

Returns

the element previously at the specified position.

Exceptions

IndexOutOfBoundsException	<i>if the index given is less than zero or greater than the List (p. 1658) size.</i>
UnsupportedOperationException	<i>if this is an unmodifiable collection.</i>

- virtual int **indexOf** (const E &value) const

Returns the index of the first occurrence of the specified element in this list, or -1 if this list does not contain the element.

- virtual int **lastIndexOf** (const E &value) const

Returns the index of the last occurrence of the specified element in this list, or -1 if this list does not contain the element.

- virtual bool **contains** (const E &value) const

Returns true if this collection contains the specified element.

More formally, returns true if and only if this collection contains at least one element e such that $(value == NULL ? e == NULL : value == e)$.

Parameters

value	<i>The value to check for presence in the collection.</i>
-------	---

Returns

true if there is at least one of the elements in the collection

Exceptions

NullPointerException	<i>if the Collection (p. 851) contains pointers and the Collection (p. 851) does not allow for NULL elements (optional check).</i>
----------------------	--

This implementation iterates over the elements in the collection, checking each element in turn for equality with the specified element.

6.513.1 Detailed Description

```
template<typename E>class decaf::util::StlList< E >
```

List (p. 1658) class that wraps the STL list object to provide a simpler interface and additional methods not provided by the STL type.

6.513.2 Constructor & Destructor Documentation

6.513.2.1 `template<typename E> decaf::util::StlList< E >::StlList () [inline]`

Default constructor - does nothing.

6.513.2.2 `template<typename E> decaf::util::StlList< E >::StlList (const StlList< E > & source) [inline]`

Copy constructor - copies the content of the given set into this one.

Parameters

<i>source</i>	The source set.
---------------	-----------------

References `decaf::util::StlList< E >::copy()`.

6.513.2.3 `template<typename E> decaf::util::StlList< E >::StlList (const Collection< E > & source) [inline]`

Copy constructor - copies the content of the given set into this one.

Parameters

<i>source</i>	The source set.
---------------	-----------------

References `decaf::util::StlList< E >::copy()`.

6.513.3 Member Function Documentation

6.513.3.1 `template<typename E> virtual void decaf::util::StlList< E >::add (int index, const E & element) [inline, virtual]`

Inserts the specified element at the specified position in this list.

Shifts the element currently at that position (if any) and any subsequent elements to the right (adds one to their indices).

Parameters

<i>index</i>	The index at which the specified element is to be inserted.
<i>element</i>	The element to be inserted in this List (p. 1658).

Exceptions

<i>IndexOutOfBoundsException</i>	if the index is greater than size of the List (p. 1658).
<i>UnsupportedOperationException</i>	if this is an unmodifiable collection.
<i>NullPointerException</i>	if the Collection (p. 851) is a container of pointers and does not allow NULL values.
<i>IllegalArgumentException</i>	if some property of the element prevents it from being added to this collection
<i>IllegalStateException</i>	if the element cannot be added at this time due to insertion restrictions.

Implements `decaf::util::List< E >` (p. 1660).

References `decaf::util::StlList< E >::size()`.

6.513.3.2 `template<typename E> virtual bool decaf::util::StlList< E >::add (const E & value) [inline, virtual]`

Returns true if this collection changed as a result of the call.

(Returns false if this collection does not permit duplicates and already contains the specified element.)

Collections that support this operation may place limitations on what elements may be added to this collection. In particular, some collections will refuse to add null elements, and others will impose restrictions on the type of elements that may be added. **Collection** (p. 851) classes should clearly specify in their documentation any restrictions on what elements may be added.

If a collection refuses to add a particular element for any reason other than that it already contains the element, it must throw an exception (rather than returning false). This preserves the invariant that a collection always contains the specified element after this call returns.

For non-pointer values, i.e. class instances or string's the object will be copied into the

collection, thus the object must support being copied, must not hide the copy constructor and assignment operator.

Parameters

<i>value</i>	The reference to the element to add to this Collection (p. 851).
--------------	---

Returns

true if the element was added to this **Collection** (p. 851).

Exceptions

<i>UnsupportedOperationException</i>	if this is an unmodifiable collection.
<i>NullPointerException</i>	if the Collection (p. 851) is a container of pointers and does not allow NULL values.
<i>IllegalArgumentException</i>	if some property of the element prevents it from being added to this collection
<i>IllegalStateException</i>	if the element cannot be added at this time due to insertion restrictions.

Reimplemented from **decaf::util::AbstractList< E >** (p. 125).

6.513.3.3 `template<typename E> virtual bool decaf::util::StlList< E >::addAll(const Collection< E > & collection) [inline, virtual]`

Adds all of the elements in the specified collection to this collection.

The behavior of this operation is undefined if the specified collection is modified while the operation is in progress. (This implies that the behavior of this call is undefined if the specified collection is this collection, and this collection is nonempty.)

Parameters

<i>collection</i>	The Collection (p. 851) whose elements are added to this one.
-------------------	--

Returns

true if this collection changed as a result of the call

Exceptions

<i>UnsupportedOperationException</i>	if this is an unmodifiable collection.
<i>NullPointerException</i>	if the Collection (p. 851) is a container of pointers and does not allow NULL values.
<i>IllegalArgumentException</i>	if some property of an element prevents it from being added to this collection

<i>IllegalStateException</i>	if an element cannot be added at this time due to insertion restrictions.
------------------------------	---

This implementation iterates over the specified collection, and adds each object returned by the iterator to this collection, in turn.

Note that this implementation will throw an `UnsupportedOperationException` unless `add` is overridden (assuming the specified collection is non-empty).

This implementation iterates over the specified collection, and adds each object returned by the iterator to this collection, in turn.

Note that this implementation will throw an `UnsupportedOperationException` unless `add` is overridden (assuming the specified collection is non-empty).

Reimplemented from **decaf::util::AbstractCollection< E >** (p. 110).

References `decaf::util::Collection< E >::isEmpty()`, `decaf::util::StlList< E >::listIterator()`, and `decaf::util::Collection< E >::toArray()`.

6.513.3.4 `template<typename E> virtual bool decaf::util::StlList< E >::addAll (int index, const Collection< E > & collection) [inline, virtual]`

Inserts all of the elements in the specified collection into this list at the specified position (optional operation).

Shifts the element currently at that position (if any) and any subsequent elements to the right (increases their indices). The new elements will appear in this list in the order that they are returned by the specified collection's iterator. The behavior of this operation is undefined if the specified collection is modified while the operation is in progress. (Note that this will occur if the specified collection is this list, and it's nonempty.)

Parameters

<i>index</i>	The index at which to insert the first element from the specified collection
<i>source</i>	The Collection (p. 851) containing elements to be added to this list

Returns

true if this list changed as a result of the call

Exceptions

<i>IndexOutOfBoundsException</i>	if the index given is less than zero or greater than the List (p. 1658) size.
<i>UnsupportedOperationException</i>	if this is an unmodifiable collection.
<i>NullPointerException</i>	if the Collection (p. 851) is a container of pointers and does not allow NULL values.

<i>IllegalArgument-Exception</i>	if some property of the element prevents it from being added to this collection
<i>IllegalStateException</i>	if the element cannot be added at this time due to insertion restrictions.

Reimplemented from **decaf::util::AbstractList< E >** (p. 126).

References **decaf::util::Collection< E >::isEmpty()**, **decaf::util::StlList< E >::list-iterator()**, **decaf::util::StlList< E >::size()**, and **decaf::util::Collection< E >::toArray()**.

6.513.3.5 `template<typename E> virtual void decaf::util::StlList< E >::clear ()`
`[inline, virtual]`

Removes all of the elements from this collection (optional operation).

The collection will be empty after this method returns.

This implementation iterates over this collection, removing each element using the **Iterator.remove** (p. 1560) operation. Most implementations will probably choose to override this method for efficiency.

Note that this implementation will throw an `UnsupportedOperationException` if the iterator returned by this collection's iterator method does not implement the remove method and this collection is non-empty.

Exceptions

<i>Unsupported-OperationException</i>	if the clear operation is not supported by this collection
---------------------------------------	--

Reimplemented from **decaf::util::AbstractList< E >** (p. 127).

6.513.3.6 `template<typename E> virtual bool decaf::util::StlList< E >::contains (const E & value) const` `[inline, virtual]`

Returns true if this collection contains the specified element.

More formally, returns true if and only if this collection contains at least one element `e` such that `(value == NULL ? e == NULL : value == e)`.

Parameters

<i>value</i>	The value to check for presence in the collection.
--------------	--

Returns

true if there is at least one of the elements in the collection

Exceptions

<i>NullPointerException</i>	if the Collection (p. 851) contains pointers and the Collection (p. 851) does not allow for NULL elements (optional check).
-----------------------------	---

This implementation iterates over the elements in the collection, checking each element in turn for equality with the specified element.

This implementation iterates over the elements in the collection, checking each element in turn for equality with the specified element.

Reimplemented from **decaf::util::AbstractCollection< E >** (p. 112).

6.513.3.7 `template<typename E> virtual void decaf::util::StlList< E >::copy (const Collection< E > & collection) [inline, virtual]`

Renders this **Collection** (p. 851) as a Copy of the given **Collection** (p. 851).

The default implementation iterates over the contents of the given collection adding each to this collection after first calling this **Collection** (p. 851)'s clear method.

Parameters

<i>collection</i>	The collection to mirror.
-------------------	---------------------------

Exceptions

<i>UnsupportedOperationException</i>	if this is an unmodifiable collection.
<i>IllegalStateException</i>	if the elements cannot be added at this time due to insertion restrictions.

Reimplemented from **decaf::util::AbstractCollection< E >** (p. 113).

Referenced by decaf::util::StlList< E >::StlList().

6.513.3.8 `template<typename E> virtual bool decaf::util::StlList< E >::equals (const Collection< E > & collection) const [inline, virtual]`

Answers true if this **Collection** (p. 851) and the one given are the same size and if each element contained in the **Collection** (p. 851) given is equal to an element contained in this collection.

Parameters

<i>collection</i>	- The Collection (p. 851) to be compared to this one.
-------------------	--

Returns

true if this **Collection** (p. 851) is equal to the one given.

Reimplemented from **decaf::util::AbstractCollection**< **E** > (p. 114).

6.513.3.9 `template<typename E> virtual E decaf::util::StlList< E >::get (int index)
const [inline, virtual]`

Gets the element contained at position passed.

Parameters

<i>index</i>	The position to get.
--------------	----------------------

Returns

value at index specified.

Exceptions

<i>IndexOutOfBoundsException</i>	if the index given is less than zero or greater than the List (p. 1658) size.
----------------------------------	--

Implements **decaf::util::List**< **E** > (p. 1662).

References **decaf::util::StlList**< **E** >::size().

6.513.3.10 `template<typename E> virtual int decaf::util::StlList< E >::indexOf (const E
& value) const [inline, virtual]`

Returns the index of the first occurrence of the specified element in this list, or -1 if this list does not contain the element.

More formally, returns the lowest index *i* such that `get(i) == value`, or -1 if there is no such index.

Parameters

<i>value</i>	The element to search for in this List (p. 1658).
--------------	--

Returns

the index of the first occurrence of the specified element in this list,

Exceptions

<i>NullPointerException</i>	if the Collection (p. 851) is a container of pointers and does not allow NULL values.
-----------------------------	--

Reimplemented from **decaf::util::AbstractList< E >** (p. 128).

6.513.3.11 `template<typename E> virtual bool decaf::util::StlList< E >::isEmpty ()
const [inline, virtual]`

Returns true if this collection contains no elements.

This implementation returns **size()** (p. 864) == 0.

Returns

true if the size method return 0.

Reimplemented from **decaf::util::AbstractCollection< E >** (p. 114).

6.513.3.12 `template<typename E> virtual Iterator<E>* decaf::util::StlList< E
>::iterator () [inline, virtual]`

Returns

an iterator over a set of elements of type T.

Reimplemented from **decaf::util::AbstractList< E >** (p. 128).

6.513.3.13 `template<typename E> virtual Iterator<E>* decaf::util::StlList< E
>::iterator () const [inline, virtual]`

Reimplemented from **decaf::util::AbstractList< E >** (p. 129).

6.513.3.14 `template<typename E> virtual int decaf::util::StlList< E >::lastIndexOf (
const E & value) const [inline, virtual]`

Returns the index of the last occurrence of the specified element in this list, or -1 if this list does not contain the element.

More formally, returns the highest index i such that **get(i)** == value or -1 if there is no such index.

Parameters

<i>value</i>	The element to search for in this List (p. 1658).
--------------	--

Returns

the index of the last occurrence of the specified element in this list.

Exceptions

<i>NullPointerException</i>	if the Collection (p. 851) is a container of pointers and does not allow NULL values.
-----------------------------	--

Reimplemented from **decaf::util::AbstractList< E >** (p. 129).

References decaf::util::StlList< E >::size().

6.513.3.15 `template<typename E> virtual ListIterator<E>* decaf::util::StlList< E >::listIterator () [inline, virtual]`

Returns

a list iterator over the elements in this list (in proper sequence).

Reimplemented from **decaf::util::AbstractList< E >** (p. 130).

Referenced by decaf::util::StlList< E >::addAll().

6.513.3.16 `template<typename E> virtual ListIterator<E>* decaf::util::StlList< E >::listIterator () const [inline, virtual]`

Reimplemented from **decaf::util::AbstractList< E >** (p. 131).

6.513.3.17 `template<typename E> virtual ListIterator<E>* decaf::util::StlList< E >::listIterator (int index) [inline, virtual]`

Parameters

<i>index</i>	index of first element to be returned from the list iterator (by a call to the next method).
--------------	--

Returns

a list iterator of the elements in this list (in proper sequence), starting at the specified position in this list. The specified index indicates the first element that would be returned by an initial call to next. An initial call to previous would return the element with the specified index minus one.

Exceptions

<i>IndexOutOfBoundsException</i>	if the index is out of range (<code>index < 0 index > size()</code>) (p. 864))
----------------------------------	---

Reimplemented from **decaf::util::AbstractList< E >** (p. 131).

References decaf::util::StlList< E >::size().

6.513.3.18 `template<typename E> virtual ListIterator<E>* decaf::util::StlList< E >::listIterator (int index) const [inline, virtual]`

Reimplemented from **decaf::util::AbstractList< E >** (p. 132).

References **decaf::util::StlList< E >::size()**.

6.513.3.19 `template<typename E> virtual bool decaf::util::StlList< E >::remove (const E & value) [inline, virtual]`

Removes a single instance of the specified element from the collection.

More formally, removes an element *e* such that (*value* == NULL ? *e* == NULL : *value* == *e*), if this collection contains one or more such elements. Returns true if this collection contained the specified element (or equivalently, if this collection changed as a result of the call).

Parameters

<i>value</i>	The reference to the element to remove from this Collection (p. 851).
--------------	--

Returns

true if the collection was changed, false otherwise.

Exceptions

<i>UnsupportedOperationException</i>	if this is an unmodifiable collection.
<i>NullPointerException</i>	if the Collection (p. 851) is a container of pointers and does not allow NULL values.

This implementation iterates over the collection looking for the specified element. If it finds the element, it removes the element from the collection using the iterator's remove method.

Note that this implementation throws an `UnsupportedOperationException` if the iterator returned by this collection's iterator method does not implement the remove method and this collection contains the specified object.

Reimplemented from **decaf::util::AbstractCollection< E >** (p. 116).

References **decaf::util::StlList< E >::size()**.

6.513.3.20 `template<typename E> virtual E decaf::util::StlList< E >::removeAt (int index) [inline, virtual]`

Removes the element at the specified position in this list.

Shifts any subsequent elements to the left (subtracts one from their indices). Returns the element that was removed from the list.

Parameters

<i>index</i>	- the index of the element to be removed.
--------------	---

Returns

the element previously at the specified position.

Exceptions

<i>IndexOutOfBoundsException</i>	if the index given is less than zero or greater than the List (p. 1658) size.
<i>UnsupportedOperationException</i>	if this is an unmodifiable collection.

Reimplemented from **decaf::util::AbstractList< E >** (p. 133).

References **decaf::util::StlList< E >::size()**.

6.513.3.21 `template<typename E> virtual E decaf::util::StlList< E >::set (int index ,
const E & element) [inline, virtual]`

Replaces the element at the specified position in this list with the specified element.

Parameters

<i>index</i>	The index of the element to replace.
<i>element</i>	The element to be stored at the specified position.

Returns

the element previously at the specified position.

Exceptions

<i>IndexOutOfBoundsException</i>	if the index given is less than zero or greater than the List (p. 1658) size.
<i>UnsupportedOperationException</i>	if this is an unmodifiable collection.
<i>NullPointerException</i>	if the Collection (p. 851) is a container of pointers and does not allow NULL values.
<i>IllegalArgumentException</i>	if some property of the element prevents it from being added to this collection
<i>IllegalStateException</i>	if the element cannot be added at this time due to insertion restrictions.

Implements **decaf::util::List< E >** (p. 1669).

References **decaf::util::StlList< E >::size()**.

6.514 decaf::util::StlMap< K, V, COMPARATOR > Class Template Reference 2559

6.513.3.22 `template<typename E> virtual int decaf::util::StlList< E >::size () const`
`[inline, virtual]`

Returns the number of elements in this collection.

If this collection contains more than Integer::MAX_VALUE elements, returns Integer::MAX_VALUE.

Returns

the number of elements in this collection

Implements **decaf::util::Collection**< E > (p. 864).

Referenced by `decaf::util::StlList< E >::add()`, `decaf::util::StlList< E >::addAll()`, `decaf::util::StlList< E >::get()`, `decaf::util::StlList< E >::lastIndexOf()`, `decaf::util::StlList< E >::listIterator()`, `decaf::util::StlList< E >::remove()`, `decaf::util::StlList< E >::removeAt()`, and `decaf::util::StlList< E >::set()`.

The documentation for this class was generated from the following file:

- `src/main/decaf/util/StlList.h`

6.514 decaf::util::StlMap< K, V, COMPARATOR > Class Template Reference

Map (p. 1768) template that wraps around a `std::map` to provide a more user-friendly interface and to provide common functions that do not exist in `std::map`.

```
#include <src/main/decaf/util/StlMap.h>
```

Inheritance diagram for `decaf::util::StlMap< K, V, COMPARATOR >`:

Public Member Functions

- **StlMap** ()
Default constructor - does nothing.
- **StlMap** (const **StlMap** &source)
Copy constructor - copies the content of the given map into this one.
- **StlMap** (const **Map**< K, V, COMPARATOR > &source)
Copy constructor - copies the content of the given map into this one.
- virtual **~StlMap** ()
- virtual bool **equals** (const **StlMap** &source) const
- virtual bool **equals** (const **Map**< K, V, COMPARATOR > &source) const
Comparison, equality is dependent on the method of determining if the element are equal.

Parameters

source	- Map (p. 1768) to compare to this one.
--------	--

Returns

*true if the **Map** (p. 1768) passed is equal in value to this one.*

- virtual void **copy** (const **StlMap** &source)
- virtual void **copy** (const **Map**< K, V, COMPARATOR > &source)

Copies the content of the source map into this map.

Erases all existing data in this map.

Parameters

source	The source object to copy from.
--------	---------------------------------

- virtual void **clear** ()

Removes all keys and values from this map.

Exceptions

UnsupportedOperation-Exception	if this map is unmodifiable.
--------------------------------	------------------------------

- virtual bool **containsKey** (const K &key) const

Indicates whether or this map contains a value for the given key.

Parameters

key	The key to look up.
-----	---------------------

Returns

true if this map contains the value, otherwise false.

- virtual bool **containsValue** (const V &value) const

Indicates whether or this map contains a value for the given value, i.e.

they are equal, this is done by operator== so the types must pass equivalence testing in this manner.

Parameters

value	The Value to look up.
-------	-----------------------

Returns

true if this map contains the value, otherwise false.

- virtual bool **isEmpty** () const

Returns

*if the **Map** (p. 1768) contains any element or not, TRUE or FALSE*

- virtual int **size** () const

Returns

The number of elements (key/value pairs) in this map.

- virtual V & **get** (const K &key)

*Gets the value mapped to the specified key in the **Map** (p. 1768).*

*If there is no element in the map whose key is equivalent to the key provided then a **NoSuchElementException** (p. 1984) is thrown.*

6.514 decaf::util::StlMap< K, V, COMPARATOR > Class Template Reference2561

Parameters

key	<i>The search key.</i>
-----	------------------------

Returns

A reference to the value for the given key.

Exceptions

NoSuchElementException (p. 1984)	<i>if the key requests doesn't exist in the Map (p. 1768).</i>
--	---

- virtual const V & **get** (const K &key) const

*Gets the value mapped to the specified key in the **Map** (p. 1768).*

*If there is no element in the map whose key is equivalent to the key provided then a **NoSuchElementException** (p. 1984) is thrown.*

Parameters

key	<i>The search key.</i>
-----	------------------------

Returns

A {const} reference to the value for the given key.

Exceptions

NoSuchElementException (p. 1984)	<i>if the key requests doesn't exist in the Map (p. 1768).</i>
--	---

- virtual void **put** (const K &key, const V &value)

Sets the value for the specified key.

Parameters

key	<i>The target key.</i>
value	<i>The value to be set.</i>

Exceptions

UnsupportedOperationException	<i>if this map is unmodifiable.</i>
--------------------------------------	-------------------------------------

- virtual void **putAll** (const **StlMap**< K, V, COMPARATOR > &other)

- virtual void **putAll** (const **Map**< K, V, COMPARATOR > &other)

*Stores a copy of the Mappings contained in the other **Map** (p. 1768) in this one.*

Parameters

other	<i>A Map (p. 1768) instance whose elements are to all be inserted in this Map (p. 1768).</i>
-------	--

Exceptions

UnsupportedOperationException	<i>If the implementing class does not support the putAll operation.</i>
--------------------------------------	---

- virtual V **remove** (const K &key)

Removes the value (key/value pair) for the specified key from the map, returns a copy

of the value that was mapped to the key.

Parameters

key	The search key.
-----	-----------------

Returns

a copy of the element that was previously mapped to the given key

Exceptions

NoSuchElementException (p. 1984)	if this key is not in the Map (p. 1768).
UnsupportedOperationException	if this map is unmodifiable.

- virtual `std::vector< K > keySet ()` const

Returns a **Set** (p. 2397) view of the mappings contained in this map.

The set is backed by the map, so changes to the map are reflected in the set, and vice-versa. If the map is modified while an iteration over the set is in progress (except through the iterator's own remove operation, or through the `setValue` operation on a map entry returned by the iterator) the results of the iteration are undefined. The set supports element removal, which removes the corresponding mapping from the map, via the **Iterator.remove** (p. 1560), **Set.remove** (p. 861), `removeAll`, `retainAll` and `clear` operations. It does not support the `add` or `addAll` operations.

Returns

the entire set of keys in this map as a `std::vector`.

- virtual `std::vector< V > values ()` const

Returns

the entire set of values in this map as a `std::vector`.

- virtual void **lock ()**

Locks the object.

- virtual bool **tryLock ()**

Attempts to Lock the object, if the lock is already held by another thread than this method returns false.

- virtual void **unlock ()**

Unlocks the object.

- virtual void **wait ()**

Waits on a signal from this object, which is generated by a call to `Notify`.

- virtual void **wait** (long long millisecs)

Waits on a signal from this object, which is generated by a call to `Notify`.

- virtual void **wait** (long long millisecs, int nanos)

Waits on a signal from this object, which is generated by a call to `Notify`.

- virtual void **notify ()**

Signals a waiter on this object that it can now wake up and continue.

- virtual void **notifyAll ()**

Signals the waiters on this object that it can now wake up and continue.

6.514 decaf::util::StlMap< K, V, COMPARATOR > Class Template Reference 2563

6.514.1 Detailed Description

template<typename K, typename V, typename COMPARATOR = std::less<K>> class decaf::util::StlMap< K, V, COMPARATOR >

Map (p. 1768) template that wraps around a std::map to provide a more user-friendly interface and to provide common functions that do not exist in std::map.

Since

1.0

6.514.2 Constructor & Destructor Documentation

6.514.2.1 template<typename K, typename V, typename COMPARATOR = std::less<K>>
decaf::util::StlMap< K, V, COMPARATOR >::StlMap () [inline]

Default constructor - does nothing.

6.514.2.2 template<typename K, typename V, typename COMPARATOR = std::less<K>>
decaf::util::StlMap< K, V, COMPARATOR >::StlMap (const StlMap< K, V,
COMPARATOR > & source) [inline]

Copy constructor - copies the content of the given map into this one.

Parameters

<i>source</i>	The source map.
---------------	-----------------

6.514.2.3 template<typename K, typename V, typename COMPARATOR = std::less<K>>
decaf::util::StlMap< K, V, COMPARATOR >::StlMap (const Map< K, V,
COMPARATOR > & source) [inline]

Copy constructor - copies the content of the given map into this one.

Parameters

<i>source</i>	The source map.
---------------	-----------------

6.514.2.4 template<typename K, typename V, typename COMPARATOR = std::less<K>>
virtual decaf::util::StlMap< K, V, COMPARATOR >::~~StlMap ()
[inline, virtual]

6.514.3 Member Function Documentation

```
6.514.3.1  template<typename K, typename V, typename COMPARATOR = std::less<K>>
            virtual void decaf::util::StlMap< K, V, COMPARATOR >::clear ( )
            [inline, virtual]
```

Removes all keys and values from this map.

Exceptions

<i>Unsupported- OperationException</i>	if this map is unmodifiable.
--	------------------------------

Implements **decaf::util::Map< K, V, COMPARATOR >** (p. 1770).

Referenced by decaf::util::StlMap< std::string, cms::Topic * >::copy().

```
6.514.3.2  template<typename K, typename V, typename COMPARATOR = std::less<K>>
            virtual bool decaf::util::StlMap< K, V, COMPARATOR >::containsKey ( const K
            & key ) const [inline, virtual]
```

Indicates whether or this map contains a value for the given key.

Parameters

<i>key</i>	The key to look up.
------------	---------------------

Returns

true if this map contains the value, otherwise false.

Implements **decaf::util::Map< K, V, COMPARATOR >** (p. 1771).

Referenced by decaf::util::StlMap< std::string, cms::Topic * >::equals().

```
6.514.3.3  template<typename K, typename V, typename COMPARATOR = std::less<K>>
            virtual bool decaf::util::StlMap< K, V, COMPARATOR >::containsValue ( const
            V & value ) const [inline, virtual]
```

Indicates whether or this map contains a value for the given value, i.e.

they are equal, this is done by operator== so the types must pass equivalence testing in this manner.

Parameters

<i>value</i>	The Value to look up.
--------------	-----------------------

6.514 decaf::util::StlMap< K, V, COMPARATOR > Class Template Reference 2565

Returns

true if this map contains the value, otherwise false.

Implements **decaf::util::Map**< K, V, COMPARATOR > (p. 1771).

```
6.514.3.4  template<typename K, typename V, typename COMPARATOR = std::less<K>>
           virtual void decaf::util::StlMap< K, V, COMPARATOR >::copy ( const StlMap<
           K, V, COMPARATOR > & source ) [inline, virtual]
```

Referenced by decaf::util::StlMap< std::string, cms::Topic * >::StlMap().

```
6.514.3.5  template<typename K, typename V, typename COMPARATOR = std::less<K>>
           virtual void decaf::util::StlMap< K, V, COMPARATOR >::copy ( const Map< K,
           V, COMPARATOR > & source ) [inline, virtual]
```

Copies the content of the source map into this map.

Erases all existing data in this map.

Parameters

<i>source</i>	The source object to copy from.
---------------	---------------------------------

Implements **decaf::util::Map**< K, V, COMPARATOR > (p. 1772).

```
6.514.3.6  template<typename K, typename V, typename COMPARATOR = std::less<K>>
           virtual bool decaf::util::StlMap< K, V, COMPARATOR >::equals ( const
           StlMap< K, V, COMPARATOR > & source ) const [inline, virtual]
```

```
6.514.3.7  template<typename K, typename V, typename COMPARATOR = std::less<K>>
           virtual bool decaf::util::StlMap< K, V, COMPARATOR >::equals ( const Map<
           K, V, COMPARATOR > & source ) const [inline, virtual]
```

Comparison, equality is dependent on the method of determining if the element are equal.

Parameters

<i>source</i>	- Map (p. 1768) to compare to this one.
---------------	--

Returns

true if the **Map** (p. 1768) passed is equal in value to this one.

Implements **decaf::util::Map**< K, V, COMPARATOR > (p. 1772).

```
6.514.3.8  template<typename K, typename V, typename COMPARATOR = std::less<K>>
            virtual V& decaf::util::StlMap< K, V, COMPARATOR >::get ( const K & key )
            [inline, virtual]
```

Gets the value mapped to the specified key in the **Map** (p. 1768).

If there is no element in the map whose key is equivalent to the key provided then a **NoSuchElementException** (p. 1984) is thrown.

Parameters

<i>key</i>	The search key.
------------	-----------------

Returns

A reference to the value for the given key.

Exceptions

<i>NoSuchElementException</i> (p. 1984)	if the key requests doesn't exist in the Map (p. 1768).
---	--

Implements **decaf::util::Map< K, V, COMPARATOR >** (p. 1773).

```
6.514.3.9  template<typename K, typename V, typename COMPARATOR = std::less<K>>
            virtual const V& decaf::util::StlMap< K, V, COMPARATOR >::get ( const K & key
            )const [inline, virtual]
```

Gets the value mapped to the specified key in the **Map** (p. 1768).

If there is no element in the map whose key is equivalent to the key provided then a **NoSuchElementException** (p. 1984) is thrown.

Parameters

<i>key</i>	The search key.
------------	-----------------

Returns

A {const} reference to the value for the given key.

Exceptions

<i>NoSuchElementException</i> (p. 1984)	if the key requests doesn't exist in the Map (p. 1768).
---	--

Implements **decaf::util::Map< K, V, COMPARATOR >** (p. 1774).

6.514 decaf::util::StlMap< K, V, COMPARATOR > Class Template Reference 2567

6.514.3.10 `template<typename K, typename V, typename COMPARATOR = std::less<K>>
virtual bool decaf::util::StlMap< K, V, COMPARATOR >::isEmpty () const
[inline, virtual]`

Returns

if the **Map** (p. 1768) contains any element or not, TRUE or FALSE

Implements **decaf::util::Map< K, V, COMPARATOR >** (p. 1774).

6.514.3.11 `template<typename K, typename V, typename COMPARATOR = std::less<K>>
virtual std::vector<K> decaf::util::StlMap< K, V, COMPARATOR >::keySet () const
[inline, virtual]`

Returns a **Set** (p. 2397) view of the mappings contained in this map.

The set is backed by the map, so changes to the map are reflected in the set, and vice-versa. If the map is modified while an iteration over the set is in progress (except through the iterator's own remove operation, or through the setValue operation on a map entry returned by the iterator) the results of the iteration are undefined. The set supports element removal, which removes the corresponding mapping from the map, via the **Iterator.remove** (p. 1560), **Set.remove** (p. 861), removeAll, retainAll and clear operations. It does not support the add or addAll operations.

Returns

the entire set of keys in this map as a std::vector.

Implements **decaf::util::Map< K, V, COMPARATOR >** (p. 1775).

6.514.3.12 `template<typename K, typename V, typename COMPARATOR = std::less<K>>
virtual void decaf::util::StlMap< K, V, COMPARATOR >::lock ()
[inline, virtual]`

Locks the object.

Exceptions

<i>RuntimeException</i>	if an error occurs while locking the object.
-------------------------	--

Implements **decaf::util::concurrent::Synchronizable** (p. 2640).

6.514.3.13 `template<typename K, typename V, typename COMPARATOR = std::less<K>>
virtual void decaf::util::StlMap< K, V, COMPARATOR >::notify ()
[inline, virtual]`

Signals a waiter on this object that it can now wake up and continue.

Must have this object locked before calling.

Exceptions

<i>IllegalMonitorStateException</i>	- if the current thread is not the owner of the the Synchronizable Object.
<i>RuntimeException</i>	if an error occurs while notifying one of the waiting threads.

Implements **decaf::util::concurrent::Synchronizable** (p. 2641).

```
6.514.3.14  template<typename K, typename V, typename COMPARATOR = std::less<K>>
virtual void decaf::util::StlMap< K, V, COMPARATOR >::notifyAll ( )
[inline, virtual]
```

Signals the waiters on this object that it can now wake up and continue.

Must have this object locked before calling.

Exceptions

<i>IllegalMonitorStateException</i>	- if the current thread is not the owner of the the Synchronizable Object.
<i>RuntimeException</i>	if an error occurs while notifying the waiting threads.

Implements **decaf::util::concurrent::Synchronizable** (p. 2642).

```
6.514.3.15  template<typename K, typename V, typename COMPARATOR = std::less<K>>
virtual void decaf::util::StlMap< K, V, COMPARATOR >::put ( const K & key,
const V & value ) [inline, virtual]
```

Sets the value for the specified key.

Parameters

<i>key</i>	The target key.
<i>value</i>	The value to be set.

Exceptions

<i>UnsupportedOperationException</i>	if this map is unmodifiable.
--------------------------------------	------------------------------

Implements **decaf::util::Map< K, V, COMPARATOR >** (p. 1776).

Referenced by **decaf::util::StlMap< std::string, cms::Topic * >::putAll()**.

6.514 decaf::util::StlMap< K, V, COMPARATOR > Class Template Reference 2569

6.514.3.16 `template<typename K, typename V, typename COMPARATOR = std::less<K>>
virtual void decaf::util::StlMap< K, V, COMPARATOR >::putAll (const
StlMap< K, V, COMPARATOR > & other) [inline, virtual]`

Referenced by `decaf::util::StlMap< std::string, cms::Topic * >::copy()`.

6.514.3.17 `template<typename K, typename V, typename COMPARATOR = std::less<K>>
virtual void decaf::util::StlMap< K, V, COMPARATOR >::putAll (const Map<
K, V, COMPARATOR > & other) [inline, virtual]`

Stores a copy of the Mappings contained in the other **Map** (p. 1768) in this one.

Parameters

<i>other</i>	A Map (p. 1768) instance whose elements are to all be inserted in this Map (p. 1768).
--------------	---

Exceptions

<i>Unsupported- OperationException</i>	If the implementing class does not support the putAll operation.
--	--

Implements `decaf::util::Map< K, V, COMPARATOR >` (p. 1777).

6.514.3.18 `template<typename K, typename V, typename COMPARATOR = std::less<K>>
virtual V decaf::util::StlMap< K, V, COMPARATOR >::remove (const K & key)
[inline, virtual]`

Removes the value (key/value pair) for the specified key from the map, returns a copy of the value that was mapped to the key.

Parameters

<i>key</i>	The search key.
------------	-----------------

Returns

a copy of the element that was previously mapped to the given key

Exceptions

<i>NoSuchElement- Exception</i> (p. 1984)	if this key is not in the Map (p. 1768).
<i>Unsupported- OperationException</i>	if this map is unmodifiable.

Implements **decaf::util::Map**< K, V, COMPARATOR > (p. 1777).

```
6.514.3.19  template<typename K, typename V, typename COMPARATOR = std::less<K>>
            virtual int decaf::util::StlMap< K, V, COMPARATOR >::size ( ) const
            [inline, virtual]
```

Returns

The number of elements (key/value pairs) in this map.

Implements **decaf::util::Map**< K, V, COMPARATOR > (p. 1778).

```
6.514.3.20  template<typename K, typename V, typename COMPARATOR = std::less<K>>
            virtual bool decaf::util::StlMap< K, V, COMPARATOR >::tryLock ( )
            [inline, virtual]
```

Attempts to Lock the object, if the lock is already held by another thread than this method returns false.

Returns

true if the lock was acquired, false if it is already held by another thread.

Exceptions

<i>RuntimeException</i>	if an error occurs while locking the object.
-------------------------	--

Implements **decaf::util::concurrent::Synchronizable** (p. 2643).

```
6.514.3.21  template<typename K, typename V, typename COMPARATOR = std::less<K>>
            virtual void decaf::util::StlMap< K, V, COMPARATOR >::unlock ( )
            [inline, virtual]
```

Unlocks the object.

Exceptions

<i>RuntimeException</i>	if an error occurs while unlocking the object.
-------------------------	--

Implements **decaf::util::concurrent::Synchronizable** (p. 2645).

```
6.514.3.22  template<typename K, typename V, typename COMPARATOR = std::less<K>>
            virtual std::vector<V> decaf::util::StlMap< K, V, COMPARATOR >::values (
            ) const [inline, virtual]
```


6.514 decaf::util::StlMap< K, V, COMPARATOR > Class Template Reference 2571

Returns

the entire set of values in this map as a std::vector.

Implements **decaf::util::Map< K, V, COMPARATOR >** (p. 1779).

Referenced by decaf::util::StlMap< std::string, cms::Topic * >::values().

```
6.514.3.23  template<typename K, typename V, typename COMPARATOR = std::less<K>>
            virtual void decaf::util::StlMap< K, V, COMPARATOR >::wait ( )
            [inline, virtual]
```

Waits on a signal from this object, which is generated by a call to Notify.

Must have this object locked before calling.

Exceptions

<i>RuntimeException</i>	if an error occurs while waiting on the object.
<i>InterruptedException</i>	if the wait is interrupted before it completes.
<i>IllegalMonitorState-Exception</i>	- if the current thread is not the owner of the the Synchronizable Object.

Implements **decaf::util::concurrent::Synchronizable** (p. 2646).

```
6.514.3.24  template<typename K, typename V, typename COMPARATOR = std::less<K>>
            virtual void decaf::util::StlMap< K, V, COMPARATOR >::wait ( long long
            millisecs ) [inline, virtual]
```

Waits on a signal from this object, which is generated by a call to Notify.

Must have this object locked before calling. This wait will timeout after the specified time interval.

Parameters

<i>millisecs</i>	the time in milliseconds to wait, or WAIT_INFINITE
------------------	--

Exceptions

<i>RuntimeException</i>	if an error occurs while waiting on the object.
<i>InterruptedException</i>	if the wait is interrupted before it completes.
<i>IllegalMonitorState-Exception</i>	- if the current thread is not the owner of the the Synchronizable Object.

Implements **decaf::util::concurrent::Synchronizable** (p. 2647).

```
6.514.3.25  template<typename K, typename V, typename COMPARATOR = std::less<K>>
            virtual void decaf::util::StlMap< K, V, COMPARATOR >::wait ( long long
            milliseconds, int nanos ) [inline, virtual]
```

Waits on a signal from this object, which is generated by a call to Notify.

Must have this object locked before calling. This wait will timeout after the specified time interval. This method is similar to the one argument wait function except that it add a finer grained control over the amount of time that it waits by adding in the additional nanosecond argument.

NOTE: The ability to wait accurately at a nanosecond scale depends on the platform and OS that the Decaf API is running on, some systems do not provide an accurate enough clock to provide this level of granularity.

Parameters

<i>milliseconds</i>	the time in milliseconds to wait, or WAIT_INFINITE
<i>nanos</i>	additional time in nanoseconds with a range of 0-999999

Exceptions

<i>IllegalArgument-Exception</i>	if an error occurs or the nanos argument is not in the range of [0-999999]
<i>RuntimeException</i>	if an error occurs while waiting on the object.
<i>InterruptedException</i>	if the wait is interrupted before it completes.
<i>IllegalMonitorState-Exception</i>	- if the current thread is not the owner of the the Synchronizable Object.

Implements **decaf::util::concurrent::Synchronizable** (p. 2648).

The documentation for this class was generated from the following file:

- src/main/decaf/util/**StlMap.h**

6.515 decaf::util::StlQueue< T > Class Template Reference

The **Queue** (p. 2222) class accepts messages with an psuh(m) command where m is the message to be queued.

```
#include <src/main/decaf/util/StlQueue.h>
```

Inheritance diagram for decaf::util::StlQueue< T >:

Data Structures

- class **QueueIterator**

Public Member Functions

- **StlQueue** ()
- virtual **~StlQueue** ()
- **Iterator**< T > * **iterator** ()
*Gets an **Iterator** (p. 1559) over this **Queue** (p. 2222).*
- void **clear** ()
Empties this queue.
- T & **front** ()
Returns a Reference to the element at the head of the queue.
- const T & **front** () const
Returns a Reference to the element at the head of the queue.
- T & **back** ()
Returns a Reference to the element at the tail of the queue.
- const T & **back** () const
Returns a Reference to the element at the tail of the queue.
- void **push** (const T &t)
Places a new Object at the Tail of the queue.
- void **enqueueFront** (const T &t)
Places a new Object at the front of the queue.
- T **pop** ()
Removes and returns the element that is at the Head of the queue.
- size_t **size** () const
*Gets the Number of elements currently in the **Queue** (p. 2222).*
- bool **empty** () const
*Checks if this **Queue** (p. 2222) is currently empty.*
- virtual std::vector< T > **toArray** () const
- void **reverse** (**StlQueue**< T > &target) const
Reverses the order of the contents of this queue and stores them in the target queue.
- virtual void **lock** ()
Locks the object.
- virtual bool **tryLock** ()
Attempts to Lock the object, if the lock is already held by another thread than this method returns false.
- virtual void **unlock** ()
Unlocks the object.
- virtual void **wait** ()
Waits on a signal from this object, which is generated by a call to Notify.
- virtual void **wait** (long long millisecs)
Waits on a signal from this object, which is generated by a call to Notify.
- virtual void **wait** (long long millisecs, int nanos)
Waits on a signal from this object, which is generated by a call to Notify.
- virtual void **notify** ()
Signals a waiter on this object that it can now wake up and continue.

- virtual void **notifyAll** ()

Signals the waiters on this object that it can now wake up and continue.

- T & **getSafeValue** ()

Fetch a reference to the safe value this object will return when there is nothing to fetch from the queue.

6.515.1 Detailed Description

```
template<typename T>class decaf::util::StlQueue< T >
```

The **Queue** (p. 2222) class accepts messages with an `push(m)` command where `m` is the message to be queued.

It destructively returns the message with **pop()** (p. 2570). **pop()** (p. 2570) returns messages in the order they were enqueued.

Queue (p. 2222) is implemented with an instance of the STL queue object. The interface is essentially the same as that of the STL queue except that the `pop` method actually reaturns a reference to the element popped. This frees the app from having to call the `front` method before calling `pop`.

```
Queue<string> sq; // make a queue to hold string messages sq.push(s); // enqueues
a message m string s = sq.pop(); // dequeues a message
```

= DESIGN CONSIDERATIONS

The **Queue** (p. 2222) class inherits from the Synchronizable interface and provides methods for locking and unlocking this queue as well as waiting on this queue. In a multi-threaded app this can allow for multiple threads to be reading from and writing to the same **Queue** (p. 2222).

Clients should consider that in a multiple threaded app it is possible that items could be placed on the queue faster than you are taking them off, so protection should be placed in your polling loop to ensure that you don't get stuck there.

6.515.2 Constructor & Destructor Documentation

6.515.2.1 `template<typename T> decaf::util::StlQueue< T >::StlQueue ()`
[inline]

6.515.2.2 `template<typename T> virtual decaf::util::StlQueue< T >::~StlQueue ()`
[inline, virtual]

6.515.3 Member Function Documentation

6.515.3.1 `template<typename T> T& decaf::util::StlQueue< T >::back ()`
[inline]

Returns a Reference to the element at the tail of the queue.

Returns

reference to a queue type object or (safe)

References decaf::util::StlQueue< T >::getSafeValue().

6.515.3.2 `template<typename T> const T& decaf::util::StlQueue< T >::back () const`
[inline]

Returns a Reference to the element at the tail of the queue.

Returns

reference to a queue type object or (safe)

References decaf::util::StlQueue< T >::getSafeValue().

6.515.3.3 `template<typename T> void decaf::util::StlQueue< T >::clear ()`
[inline]

Empties this queue.

6.515.3.4 `template<typename T> bool decaf::util::StlQueue< T >::empty () const`
[inline]

Checks if this **Queue** (p. 2222) is currently empty.

Returns

boolean indicating queue emptiness

6.515.3.5 `template<typename T> void decaf::util::StlQueue< T >::enqueueFront (`
`const T & t)` [inline]

Places a new Object at the front of the queue.

Parameters

<i>t</i> - Queue (p. 2222) Object Type reference.
--

6.515.3.6 `template<typename T> T& decaf::util::StlQueue< T >::front ()`
[inline]

Returns a Reference to the element at the head of the queue.

Returns

reference to a queue type object or (safe)

References `decaf::util::StlQueue< T >::getSafeValue()`.

6.515.3.7 `template<typename T> const T& decaf::util::StlQueue< T >::front () const`
`[inline]`

Returns a Reference to the element at the head of the queue.

Returns

reference to a queue type object or (safe)

References `decaf::util::StlQueue< T >::getSafeValue()`.

6.515.3.8 `template<typename T> T& decaf::util::StlQueue< T >::getSafeValue ()`
`[inline]`

Fetch a reference to the safe value this object will return when there is nothing to fetch from the queue.

Returns

Reference to this Queues safe object

Referenced by `decaf::util::StlQueue< T >::back()`, `decaf::util::StlQueue< T >::front()`, and `decaf::util::StlQueue< T >::pop()`.

6.515.3.9 `template<typename T> Iterator<T>* decaf::util::StlQueue< T >::iterator ()`
`[inline]`

Gets an **Iterator** (p. 1559) over this **Queue** (p. 2222).

Returns

new iterator pointer that is owned by the caller.

6.515.3.10 `template<typename T> virtual void decaf::util::StlQueue< T >::lock ()`
`[inline, virtual]`

Locks the object.

Exceptions

<i>RuntimeException</i>	if an error occurs while locking the object.
-------------------------	--

Implements **decaf::util::concurrent::Synchronizable** (p. 2640).

References decaf::util::concurrent::Mutex::lock().

6.515.3.11 `template<typename T> virtual void decaf::util::StlQueue< T >::notify ()`
`[inline, virtual]`

Signals a waiter on this object that it can now wake up and continue.

Must have this object locked before calling.

Exceptions

<i>IllegalMonitorState-Exception</i>	- if the current thread is not the owner of the the Synchronizable Object.
<i>RuntimeException</i>	if an error occurs while notifying one of the waiting threads.

Implements **decaf::util::concurrent::Synchronizable** (p. 2641).

References decaf::util::concurrent::Mutex::notify().

6.515.3.12 `template<typename T> virtual void decaf::util::StlQueue< T >::notifyAll ()`
`[inline, virtual]`

Signals the waiters on this object that it can now wake up and continue.

Must have this object locked before calling.

Exceptions

<i>IllegalMonitorState-Exception</i>	- if the current thread is not the owner of the the Synchronizable Object.
<i>RuntimeException</i>	if an error occurs while notifying the waiting threads.

Implements **decaf::util::concurrent::Synchronizable** (p. 2642).

References decaf::util::concurrent::Mutex::notifyAll().

6.515.3.13 `template<typename T> T decaf::util::StlQueue< T >::pop ()` `[inline]`

Removes and returns the element that is at the Head of the queue.

Returns

reference to a queue type object or (safe)

References decaf::util::StlQueue< T >::getSafeValue().

6.515.3.14 `template<typename T> void decaf::util::StlQueue< T >::push (const T & t)`
`[inline]`

Places a new Object at the Tail of the queue.

Parameters

<i>t</i>	- Queue (p. 2222) Object Type reference.
----------	---

6.515.3.15 `template<typename T> void decaf::util::StlQueue< T >::reverse (`
`StlQueue< T > & target) const [inline]`

Reverses the order of the contents of this queue and stores them in the target queue.

Parameters

<i>target</i>	- The target queue that will receive the contents of this queue in reverse order.
---------------	---

6.515.3.16 `template<typename T> size_t decaf::util::StlQueue< T >::size () const`
`[inline]`

Gets the Number of elements currently in the **Queue** (p. 2222).

Returns

Queue (p. 2222) Size

6.515.3.17 `template<typename T> virtual std::vector<T> decaf::util::StlQueue< T`
`>::toArray () const [inline, virtual]`

Returns

the all values in this queue as a std::vector.

6.515.3.18 `template<typename T> virtual bool decaf::util::StlQueue< T >::tryLock ()`
`[inline, virtual]`

Attempts to Lock the object, if the lock is already held by another thread than this method returns false.

Returns

true if the lock was acquired, false if it is already held by another thread.

Exceptions

<i>RuntimeException</i>	if an error occurs while locking the object.
-------------------------	--

Implements **decaf::util::concurrent::Synchronizable** (p. 2643).

References decaf::util::concurrent::Mutex::tryLock().

6.515.3.19 `template<typename T> virtual void decaf::util::StlQueue< T >::unlock ()`
`[inline, virtual]`

Unlocks the object.

Exceptions

<i>RuntimeException</i>	if an error occurs while unlocking the object.
-------------------------	--

Implements **decaf::util::concurrent::Synchronizable** (p. 2645).

References decaf::util::concurrent::Mutex::unlock().

6.515.3.20 `template<typename T> virtual void decaf::util::StlQueue< T >::wait ()`
`[inline, virtual]`

Waits on a signal from this object, which is generated by a call to Notify.

Must have this object locked before calling.

Exceptions

<i>RuntimeException</i>	if an error occurs while waiting on the object.
<i>InterruptedException</i>	if the wait is interrupted before it completes.
<i>IllegalMonitorStateException</i>	- if the current thread is not the owner of the the Synchronizable Object.

Implements **decaf::util::concurrent::Synchronizable** (p. 2646).

References decaf::util::concurrent::Mutex::wait().

6.515.3.21 `template<typename T> virtual void decaf::util::StlQueue< T >::wait (long`
`long millisecs) [inline, virtual]`

Waits on a signal from this object, which is generated by a call to Notify.

Must have this object locked before calling. This wait will timeout after the specified time interval.

Parameters

<i>millisecs</i>	the time in milliseconds to wait, or WAIT_INFINITE
------------------	--

Exceptions

<i>RuntimeException</i>	if an error occurs while waiting on the object.
<i>InterruptedException</i>	if the wait is interrupted before it completes.
<i>IllegalMonitorStateException</i>	- if the current thread is not the owner of the the Synchronizable Object.

Implements **decaf::util::concurrent::Synchronizable** (p. 2647).

References `decaf::util::concurrent::Mutex::wait()`.

6.515.3.22 `template<typename T> virtual void decaf::util::StlQueue< T >::wait (long long millisecs, int nanos) [inline, virtual]`

Waits on a signal from this object, which is generated by a call to Notify.

Must have this object locked before calling. This wait will timeout after the specified time interval. This method is similar to the one argument wait function except that it add a finer grained control over the amount of time that it waits by adding in the additional nanosecond argument.

NOTE: The ability to wait accurately at a nanosecond scale depends on the platform and OS that the Decaf API is running on, some systems do not provide an accurate enough clock to provide this level of granularity.

Parameters

<i>millisecs</i>	the time in milliseconds to wait, or WAIT_INFINITE
<i>nanos</i>	additional time in nanoseconds with a range of 0-999999

Exceptions

<i>IllegalArgument-Exception</i>	if an error occurs or the nanos argument is not in the range of [0-999999]
<i>RuntimeException</i>	if an error occurs while waiting on the object.
<i>InterruptedException</i>	if the wait is interrupted before it completes.
<i>IllegalMonitorStateException</i>	- if the current thread is not the owner of the the Synchronizable Object.

Implements **decaf::util::concurrent::Synchronizable** (p. 2648).

References `decaf::util::concurrent::Mutex::wait()`.

The documentation for this class was generated from the following file:

- `src/main/decaf/util/StlQueue.h`

6.516 decaf::util::StlSet< E > Class Template Reference

Set (p. 2397) template that wraps around a std::set to provide a more user-friendly interface and to provide common functions that do not exist in std::set.

```
#include <src/main/decaf/util/StlSet.h>
```

Inheritance diagram for decaf::util::StlSet< E >:

Data Structures

- class **ConstSetIterator**
- class **SetIterator**

Public Member Functions

- **StlSet** ()
Default constructor - does nothing.
- **StlSet** (const **StlSet** &source)
Copy constructor - copies the content of the given set into this one.
- **StlSet** (const **Collection**< E > &source)
Copy constructor - copies the content of the given set into this one.
- virtual ~**StlSet** ()
- **Iterator**< E > * **iterator** ()
Returns
an iterator over a set of elements of type T.
- **Iterator**< E > * **iterator** () const
- virtual bool **equals** (const **Collection**< E > &collection) const
*Answers true if this **Collection** (p. 851) and the one given are the same size and if each element contained in the **Collection** (p. 851) given is equal to an element contained in this collection.*

Parameters

collection	- The Collection (p. 851) to be compared to this one.
------------	--

Returns

*true if this **Collection** (p. 851) is equal to the one given.*

- virtual void **copy** (const **Collection**< E > &collection)
*Renders this **Collection** (p. 851) as a Copy of the given **Collection** (p. 851).
The default implementation iterates over the contents of the given collection adding each to this collection after first calling this **Collection** (p. 851)'s clear method.*

Parameters

collection	The collection to mirror.
------------	---------------------------

Exceptions

UnsupportedOperation-Exception	<i>if this is an unmodifiable collection.</i>
IllegalStateException	<i>if the elements cannot be added at this time due to insertion restrictions.</i>

- virtual void **clear** ()

Removes all of the elements from this collection (optional operation).

The collection will be empty after this method returns.

*This implementation iterates over this collection, removing each element using the **Iterator.remove** (p. 1560) operation. Most implementations will probably choose to override this method for efficiency.*

*Note that this implementation will throw an **UnsupportedOperationException** if the iterator returned by this collection's iterator method does not implement the remove method and this collection is non-empty.*

Exceptions

UnsupportedOperation-Exception	<i>if the clear operation is not supported by this collection</i>
--------------------------------	---

- virtual bool **contains** (const E &value) const

Returns true if this collection contains the specified element.

*More formally, returns true if and only if this collection contains at least one element *e* such that (value == NULL ? e == NULL : value == e).*

Parameters

value	<i>The value to check for presence in the collection.</i>
-------	---

Returns

true if there is at least one of the elements in the collection

Exceptions

NullPointerException	<i>if the Collection (p. 851) contains pointers and the Collection (p. 851) does not allow for NULL elements (optional check).</i>
----------------------	--

This implementation iterates over the elements in the collection, checking each element in turn for equality with the specified element.

- virtual bool **isEmpty** () const
- virtual int **size** () const
- virtual bool **add** (const E &value)

Returns true if this collection changed as a result of the call.

(Returns false if this collection does not permit duplicates and already contains the specified element.)

Collections that support this operation may place limitations on what elements may be added to this collection. In particular, some collections will refuse to add null elements, and others will impose restrictions on the type of elements that may be added.

***Collection** (p. 851) classes should clearly specify in their documentation any restrictions on what elements may be added.*

If a collection refuses to add a particular element for any reason other than that it already contains the element, it must throw an exception (rather than returning false). This preserves the invariant that a collection always contains the specified element after this call returns.

For non-pointer values, i.e. class instances or string's the object will be copied into the collection, thus the object must support being copied, must not hide the copy constructor and assignment operator.

Parameters

value	The reference to the element to add to this Collection (p. 851).
-------	---

Returns

*true if the element was added to this **Collection** (p. 851).*

Exceptions

UnsupportedOperationException	<i>if this is an unmodifiable collection.</i>
NullPointerException	<i>if the Collection (p. 851) is a container of pointers and does not allow NULL values.</i>
IllegalArgumentException	<i>if some property of the element prevents it from being added to this collection</i>
IllegalStateException	<i>if the element cannot be added at this time due to insertion restrictions.</i>

- virtual bool **remove** (const E &value)

*Removes a single instance of the specified element from the collection. More formally, removes an element *e* such that (value == NULL ? e == NULL : value == e), if this collection contains one or more such elements. Returns true if this collection contained the specified element (or equivalently, if this collection changed as a result of the call).*

Parameters

value	The reference to the element to remove from this Collection (p. 851).
-------	--

Returns

true if the collection was changed, false otherwise.

Exceptions

UnsupportedOperationException	<i>if this is an unmodifiable collection.</i>
NullPointerException	<i>if the Collection (p. 851) is a container of pointers and does not allow NULL values.</i>

This implementation iterates over the collection looking for the specified element. - If it finds the element, it removes the element from the collection using the iterator's remove method.

Note that this implementation throws an UnsupportedOperationException if the iterator returned by this collection's iterator method does not implement the remove method and this collection contains the specified object.

6.516.1 Detailed Description

```
template<typename E> class decaf::util::StlSet< E >
```

Set (p. 2397) template that wraps around a std::set to provide a more user-friendly interface and to provide common functions that do not exist in std::set.

6.516.2 Constructor & Destructor Documentation

6.516.2.1 `template<typename E> decaf::util::StlSet< E >::StlSet () [inline]`

Default constructor - does nothing.

6.516.2.2 `template<typename E> decaf::util::StlSet< E >::StlSet (const StlSet< E > & source) [inline]`

Copy constructor - copies the content of the given set into this one.

Parameters

<i>source</i>	The source set.
---------------	-----------------

6.516.2.3 `template<typename E> decaf::util::StlSet< E >::StlSet (const Collection< E > & source) [inline]`

Copy constructor - copies the content of the given set into this one.

Parameters

<i>source</i>	The source set.
---------------	-----------------

6.516.2.4 `template<typename E> virtual decaf::util::StlSet< E >::~StlSet () [inline, virtual]`

6.516.3 Member Function Documentation

6.516.3.1 `template<typename E> virtual bool decaf::util::StlSet< E >::add (const E & value) [inline, virtual]`

Returns true if this collection changed as a result of the call.

(Returns false if this collection does not permit duplicates and already contains the specified element.)

Collections that support this operation may place limitations on what elements may be added to this collection. In particular, some collections will refuse to add null elements, and others will impose restrictions on the type of elements that may be added. **Collection** (p. 851) classes should clearly specify in their documentation any restrictions on what elements may be added.

If a collection refuses to add a particular element for any reason other than that it already contains the element, it must throw an exception (rather than returning false). This preserves the invariant that a collection always contains the specified element after this call returns.

For non-pointer values, i.e. class instances or string's the object will be copied into the collection, thus the object must support being copied, must not hide the copy constructor

and assignment operator.

Parameters

<i>value</i>	The reference to the element to add to this Collection (p. 851).
--------------	---

Returns

true if the element was added to this **Collection** (p. 851).

Exceptions

<i>UnsupportedOperationException</i>	if this is an unmodifiable collection.
<i>NullPointerException</i>	if the Collection (p. 851) is a container of pointers and does not allow NULL values.
<i>IllegalArgumentException</i>	if some property of the element prevents it from being added to this collection
<i>IllegalStateException</i>	if the element cannot be added at this time due to insertion restrictions.

Implements **decaf::util::Collection< E >** (p. 853).

6.516.3.2 `template<typename E> virtual void decaf::util::StlSet< E >::clear ()`
`[inline, virtual]`

Removes all of the elements from this collection (optional operation).

The collection will be empty after this method returns.

This implementation iterates over this collection, removing each element using the **Iterator.remove** (p. 1560) operation. Most implementations will probably choose to override this method for efficiency.

Note that this implementation will throw an `UnsupportedOperationException` if the iterator returned by this collection's iterator method does not implement the remove method and this collection is non-empty.

Exceptions

<i>UnsupportedOperationException</i>	if the clear operation is not supported by this collection
--------------------------------------	--

Reimplemented from **decaf::util::AbstractCollection< E >** (p. 111).

6.516.3.3 `template<typename E> virtual bool decaf::util::StlSet< E >::contains (const`
`E & value) const [inline, virtual]`

Returns true if this collection contains the specified element.

More formally, returns true if and only if this collection contains at least one element *e* such that (value == NULL ? e == NULL : value == e).

Parameters

<i>value</i>	The value to check for presence in the collection.
--------------	--

Returns

true if there is at least one of the elements in the collection

Exceptions

<i>NullPointerException</i>	if the Collection (p. 851) contains pointers and the Collection (p. 851) does not allow for NULL elements (optional check).
-----------------------------	---

This implementation iterates over the elements in the collection, checking each element in turn for equality with the specified element.

Reimplemented from **decaf::util::AbstractCollection**< **E** > (p. 112).

6.516.3.4 `template<typename E> virtual void decaf::util::StlSet< E >::copy (const Collection< E > & collection) [inline, virtual]`

Renders this **Collection** (p. 851) as a Copy of the given **Collection** (p. 851).

The default implementation iterates over the contents of the given collection adding each to this collection after first calling this **Collection** (p. 851)'s clear method.

Parameters

<i>collection</i>	The collection to mirror.
-------------------	---------------------------

Exceptions

<i>UnsupportedOperationException</i>	if this is an unmodifiable collection.
<i>IllegalStateException</i>	if the elements cannot be added at this time due to insertion restrictions.

Reimplemented from **decaf::util::AbstractCollection**< **E** > (p. 113).

Referenced by `decaf::util::StlSet< Resource * >::copy()`, and `decaf::util::StlSet< Resource * >::StlSet()`.

6.516.3.5 `template<typename E> virtual bool decaf::util::StlSet< E >::equals (const Collection< E > & collection) const [inline, virtual]`

Answers true if this **Collection** (p. 851) and the one given are the same size and if each element contained in the **Collection** (p. 851) given is equal to an element contained in this collection.

Parameters

<i>collection</i>	- The Collection (p. 851) to be compared to this one.
-------------------	--

Returns

true if this **Collection** (p. 851) is equal to the one given.

Reimplemented from **decaf::util::AbstractCollection< E >** (p. 114).

Referenced by **decaf::util::StlSet< Resource * >::equals()**.

6.516.3.6 `template<typename E> virtual bool decaf::util::StlSet< E >::isEmpty () const [inline, virtual]`

Returns

if the set contains any element or not, TRUE or FALSE

Reimplemented from **decaf::util::AbstractCollection< E >** (p. 114).

6.516.3.7 `template<typename E> Iterator<E>* decaf::util::StlSet< E >::iterator () [inline, virtual]`

Returns

an iterator over a set of elements of type T.

Implements **decaf::lang::Iterable< E >** (p. 1557).

6.516.3.8 `template<typename E> Iterator<E>* decaf::util::StlSet< E >::iterator () const [inline, virtual]`

Implements **decaf::lang::Iterable< E >** (p. 1558).

6.516.3.9 `template<typename E> virtual bool decaf::util::StlSet< E >::remove (const E & value) [inline, virtual]`

Removes a single instance of the specified element from the collection.

More formally, removes an element e such that (value == NULL ? e == NULL : value == e), if this collection contains one or more such elements. Returns true if this collection

contained the specified element (or equivalently, if this collection changed as a result of the call).

Parameters

<i>value</i>	The reference to the element to remove from this Collection (p. 851).
--------------	--

Returns

true if the collection was changed, false otherwise.

Exceptions

<i>UnsupportedOperationException</i>	if this is an unmodifiable collection.
<i>NullPointerException</i>	if the Collection (p. 851) is a container of pointers and does not allow NULL values.

This implementation iterates over the collection looking for the specified element. If it finds the element, it removes the element from the collection using the iterator's remove method.

Note that this implementation throws an `UnsupportedOperationException` if the iterator returned by this collection's iterator method does not implement the remove method and this collection contains the specified object.

Reimplemented from **decaf::util::AbstractCollection**< **E** > (p. 116).

```
6.516.3.10  template<typename E> virtual int decaf::util::StlSet< E >::size ( ) const
            [inline, virtual]
```

Returns

The number of elements in this set.

Implements **decaf::util::Collection**< **E** > (p. 864).

The documentation for this class was generated from the following file:

- src/main/decaf/util/**StlSet.h**

6.517 activemq::wireformat::stomp::StompCommandConstants - Class Reference

```
#include <src/main/activemq/wireformat/stomp/StompCommand-
Constants.h>
```

Static Public Attributes

- static const std::string **CONNECT**
- static const std::string **CONNECTED**
- static const std::string **DISCONNECT**
- static const std::string **SUBSCRIBE**
- static const std::string **UNSUBSCRIBE**
- static const std::string **MESSAGE**
- static const std::string **SEND**
- static const std::string **BEGIN**
- static const std::string **COMMIT**
- static const std::string **ABORT**
- static const std::string **ACK**
- static const std::string **ERROR_CMD**
- static const std::string **RECEIPT**
- static const std::string **HEADER_DESTINATION**
- static const std::string **HEADER_TRANSACTIONID**
- static const std::string **HEADER_CONTENTLENGTH**
- static const std::string **HEADER_SESSIONID**
- static const std::string **HEADER_RECEIPT_REQUIRED**
- static const std::string **HEADER_RECEIPTID**
- static const std::string **HEADER_MESSAGEID**
- static const std::string **HEADER_ACK**
- static const std::string **HEADER_LOGIN**
- static const std::string **HEADER_PASSWORD**
- static const std::string **HEADER_CLIENT_ID**
- static const std::string **HEADER_MESSAGE**
- static const std::string **HEADER_CORRELATIONID**
- static const std::string **HEADER_REQUESTID**
- static const std::string **HEADER_RESPONSEID**
- static const std::string **HEADER_EXPIRES**
- static const std::string **HEADER_PERSISTENT**
- static const std::string **HEADER_REPLYTO**
- static const std::string **HEADER_TYPE**
- static const std::string **HEADER_DISPATCH_ASYNC**
- static const std::string **HEADER_EXCLUSIVE**
- static const std::string **HEADER_MAXPENDINGMSGLIMIT**
- static const std::string **HEADER_NOLOCAL**
- static const std::string **HEADER_PREFETCHSIZE**
- static const std::string **HEADER_JMSPRIORITY**
- static const std::string **HEADER_CONSUMERPRIORITY**
- static const std::string **HEADER_RETROACTIVE**
- static const std::string **HEADER_SUBSCRIPTIONNAME**
- static const std::string **HEADER_OLDSUBSCRIPTIONNAME**
- static const std::string **HEADER_TIMESTAMP**
- static const std::string **HEADER_REDELIVERED**
- static const std::string **HEADER_REDELIVERYCOUNT**

- static const std::string **HEADER_SELECTOR**
- static const std::string **HEADER_ID**
- static const std::string **HEADER_SUBSCRIPTION**
- static const std::string **HEADER_TRANSFORMATION**
- static const std::string **HEADER_TRANSFORMATION_ERROR**
- static const std::string **ACK_CLIENT**
- static const std::string **ACK_AUTO**
- static const std::string **ACK_INDIVIDUAL**
- static const std::string **TEXT**
- static const std::string **BYTES**
- static const std::string **QUEUE_PREFIX**
- static const std::string **TOPIC_PREFIX**
- static const std::string **TEMPQUEUE_PREFIX**
- static const std::string **TEMPTOPIC_PREFIX**

6.517.1 Field Documentation

- 6.517.1.1 `const std::string activemq::wireformat::stomp::StompCommand-Constants::ABORT` `[static]`
- 6.517.1.2 `const std::string activemq::wireformat::stomp::StompCommand-Constants::ACK` `[static]`
- 6.517.1.3 `const std::string activemq::wireformat::stomp::StompCommand-Constants::ACK_AUTO` `[static]`
- 6.517.1.4 `const std::string activemq::wireformat::stomp::StompCommand-Constants::ACK_CLIENT` `[static]`
- 6.517.1.5 `const std::string activemq::wireformat::stomp::StompCommand-Constants::ACK_INDIVIDUAL` `[static]`
- 6.517.1.6 `const std::string activemq::wireformat::stomp::StompCommand-Constants::BEGIN` `[static]`
- 6.517.1.7 `const std::string activemq::wireformat::stomp::StompCommand-Constants::BYTES` `[static]`
- 6.517.1.8 `const std::string activemq::wireformat::stomp::StompCommand-Constants::COMMIT` `[static]`
- 6.517.1.9 `const std::string activemq::wireformat::stomp::StompCommand-Constants::CONNECT` `[static]`
- 6.517.1.10 `const std::string activemq::wireformat::stomp::StompCommand-Constants::CONNECTED` `[static]`

6.517 `activemq::wireformat::stomp::StompCommandConstants` Class Reference

- 6.517.1.11 `const std::string activemq::wireformat::stomp::StompCommandConstants::DISCONNECT` [static]
- 6.517.1.12 `const std::string activemq::wireformat::stomp::StompCommandConstants::ERROR_CMD` [static]
- 6.517.1.13 `const std::string activemq::wireformat::stomp::StompCommandConstants::HEADER_ACK` [static]
- 6.517.1.14 `const std::string activemq::wireformat::stomp::StompCommandConstants::HEADER_CLIENT_ID` [static]
- 6.517.1.15 `const std::string activemq::wireformat::stomp::StompCommandConstants::HEADER_CONSUMERPRIORITY` [static]
- 6.517.1.16 `const std::string activemq::wireformat::stomp::StompCommandConstants::HEADER_CONTENTLENGTH` [static]
- 6.517.1.17 `const std::string activemq::wireformat::stomp::StompCommandConstants::HEADER_CORRELATIONID` [static]
- 6.517.1.18 `const std::string activemq::wireformat::stomp::StompCommandConstants::HEADER_DESTINATION` [static]
- 6.517.1.19 `const std::string activemq::wireformat::stomp::StompCommandConstants::HEADER_DISPATCH_ASYNC` [static]
- 6.517.1.20 `const std::string activemq::wireformat::stomp::StompCommandConstants::HEADER_EXCLUSIVE` [static]
- 6.517.1.21 `const std::string activemq::wireformat::stomp::StompCommandConstants::HEADER_EXPIRES` [static]
- 6.517.1.22 `const std::string activemq::wireformat::stomp::StompCommandConstants::HEADER_ID` [static]
- 6.517.1.23 `const std::string activemq::wireformat::stomp::StompCommandConstants::HEADER_JMSPRIORITY` [static]

- 6.517.1.24 **const std::string activemq::wireformat::stomp::StompCommand-Constants::HEADER_LOGIN** [static]
- 6.517.1.25 **const std::string activemq::wireformat::stomp::StompCommandConstants::HEADER_MAXPENDINGMSGLIMIT** [static]
- 6.517.1.26 **const std::string activemq::wireformat::stomp::StompCommandConstants::HEADER_MESSAGE** [static]
- 6.517.1.27 **const std::string activemq::wireformat::stomp::StompCommandConstants::HEADER_MESSAGEID** [static]
- 6.517.1.28 **const std::string activemq::wireformat::stomp::StompCommandConstants::HEADER_NOLOCAL** [static]
- 6.517.1.29 **const std::string activemq::wireformat::stomp::StompCommandConstants::HEADER_OLDSUBSCRIPTIONNAME** [static]
- 6.517.1.30 **const std::string activemq::wireformat::stomp::StompCommandConstants::HEADER_PASSWORD** [static]
- 6.517.1.31 **const std::string activemq::wireformat::stomp::StompCommandConstants::HEADER_PERSISTENT** [static]
- 6.517.1.32 **const std::string activemq::wireformat::stomp::StompCommandConstants::HEADER_PREFETCHSIZE** [static]
- 6.517.1.33 **const std::string activemq::wireformat::stomp::StompCommandConstants::HEADER_RECEIPT_REQUIRED** [static]
- 6.517.1.34 **const std::string activemq::wireformat::stomp::StompCommandConstants::HEADER_RECEIPTID** [static]
- 6.517.1.35 **const std::string activemq::wireformat::stomp::StompCommandConstants::HEADER_REDELIVERED** [static]

6.517 **activemq::wireformat::stomp::StompCommandConstants** Class Reference

- 6.517.1.36 **const std::string activemq::wireformat::stomp::StompCommandConstants::HEADER_REDELIVERYCOUNT**
[static]
- 6.517.1.37 **const std::string activemq::wireformat::stomp::StompCommandConstants::HEADER_REPLYTO**
[static]
- 6.517.1.38 **const std::string activemq::wireformat::stomp::StompCommandConstants::HEADER_REQUESTID**
[static]
- 6.517.1.39 **const std::string activemq::wireformat::stomp::StompCommandConstants::HEADER_RESPONSEID**
[static]
- 6.517.1.40 **const std::string activemq::wireformat::stomp::StompCommandConstants::HEADER_RETROACTIVE**
[static]
- 6.517.1.41 **const std::string activemq::wireformat::stomp::StompCommandConstants::HEADER_SELECTOR**
[static]
- 6.517.1.42 **const std::string activemq::wireformat::stomp::StompCommandConstants::HEADER_SESSIONID**
[static]
- 6.517.1.43 **const std::string activemq::wireformat::stomp::StompCommandConstants::HEADER_SUBSCRIPTION**
[static]
- 6.517.1.44 **const std::string activemq::wireformat::stomp::StompCommandConstants::HEADER_SUBSCRIPTIONNAME**
[static]
- 6.517.1.45 **const std::string activemq::wireformat::stomp::StompCommandConstants::HEADER_TIMESTAMP**
[static]
- 6.517.1.46 **const std::string activemq::wireformat::stomp::StompCommandConstants::HEADER_TRANSACTIONID**
[static]
- 6.517.1.47 **const std::string activemq::wireformat::stomp::StompCommandConstants::HEADER_TRANSFORMATION**
[static]

- 6.517.1.48 `const std::string activemq::wireformat::stomp::StompCommandConstants::HEADER_TRANSFORMATION_ERROR`
[static]
- 6.517.1.49 `const std::string activemq::wireformat::stomp::StompCommandConstants::HEADER_TYPE` [static]
- 6.517.1.50 `const std::string activemq::wireformat::stomp::StompCommandConstants::MESSAGE` [static]
- 6.517.1.51 `const std::string activemq::wireformat::stomp::StompCommandConstants::QUEUE_PREFIX` [static]
- 6.517.1.52 `const std::string activemq::wireformat::stomp::StompCommandConstants::RECEIPT` [static]
- 6.517.1.53 `const std::string activemq::wireformat::stomp::StompCommandConstants::SEND` [static]
- 6.517.1.54 `const std::string activemq::wireformat::stomp::StompCommandConstants::SUBSCRIBE` [static]
- 6.517.1.55 `const std::string activemq::wireformat::stomp::StompCommandConstants::TEMPQUEUE_PREFIX`
[static]
- 6.517.1.56 `const std::string activemq::wireformat::stomp::StompCommandConstants::TEMPTOPIC_PREFIX`
[static]
- 6.517.1.57 `const std::string activemq::wireformat::stomp::StompCommandConstants::TEXT` [static]
- 6.517.1.58 `const std::string activemq::wireformat::stomp::StompCommandConstants::TOPIC_PREFIX` [static]
- 6.517.1.59 `const std::string activemq::wireformat::stomp::StompCommandConstants::UNSUBSCRIBE` [static]

The documentation for this class was generated from the following file:

- `src/main/activemq/wireformat/stomp/StompCommandConstants.h`

6.518 `activemq::wireformat::stomp::StompFrame` Class Reference

A Stomp-level message frame that encloses all messages to and from the broker.


```
#include <src/main/activemq/wireformat/stomp/StompFrame.h>
```

Public Member Functions

- **StompFrame** ()
Default constructor.
- virtual **~StompFrame** ()
Destruction.
- **StompFrame * clone** () const
Clone this message exactly, returns a new instance that the caller is required to delete.
- void **copy** (const **StompFrame** *src)
Copies the contents of the passed Frame to this one.
- void **setCommand** (const std::string &cmd)
Sets the command for this stomp frame.
- const std::string & **getCommand** () const
Accessor for this frame's command field.
- bool **hasProperty** (const std::string &name) const
Checks if the given property is present in the Frame.
- std::string **getProperty** (const std::string &name, const std::string &fallback="") const
Gets a property from this Frame's properties and returns it, or the default value given.
- std::string **removeProperty** (const std::string &name)
Gets and remove the property specified, if the property is not set, this method returns the empty string.
- void **setProperty** (const std::string &name, const std::string &value)
Sets the property given to the value specified in this Frame's Properties.
- **decaf::util::Properties & getProperties** ()
Gets access to the header properties for this frame.
- const **decaf::util::Properties & getProperties** () const
- const std::vector< unsigned char > & **getBody** () const
Accessor for the body data of this frame.
- std::vector< unsigned char > & **getBody** ()
Non-const version of the body accessor.
- std::size_t **getBodyLength** () const
Return the number of bytes contained in this frames body.
- void **setBody** (const unsigned char *bytes, std::size_t numBytes)
Sets the body data of this frame as a byte sequence.
- void **toStream** (**decaf::io::DataOutputStream** *stream) const
Writes this Frame to an OuputStream in the Stomp Wire Format.
- void **fromStream** (**decaf::io::DataInputStream** *stream)
Reads a Stop Frame from a DataInputStream in the Stomp Wire format.

6.518.1 Detailed Description

A Stomp-level message frame that encloses all messages to and from the broker.

6.518.2 Constructor & Destructor Documentation

6.518.2.1 `activemq::wireformat::stomp::StompFrame::StompFrame ()`

Default constructor.

6.518.2.2 `virtual activemq::wireformat::stomp::StompFrame::~~StompFrame ()` [virtual]

Destruction.

6.518.3 Member Function Documentation

6.518.3.1 `StompFrame* activemq::wireformat::stomp::StompFrame::clone ()` `const`

Clone this message exactly, returns a new instance that the caller is required to delete.

Returns

new copy of this message

6.518.3.2 `void activemq::wireformat::stomp::StompFrame::copy (const` `StompFrame * src)`

Copies the contents of the passed Frame to this one.

Parameters

<i>src</i>	- Frame to copy
------------	-----------------

6.518.3.3 `void activemq::wireformat::stomp::StompFrame::fromStream (` `decaf::io::DataInputStream * stream)`

Reads a Stop Frame from a DataInputStream in the Stomp Wire format.

Parameters

<i>stream</i>	- The stream to read the Frame from.
---------------	--------------------------------------

Exceptions

<i>IOException</i>	if an error occurs while writing the Frame.
--------------------	---

6.518.3.4 `const std::vector<unsigned char>& activemq::wireformat::stomp::StompFrame::getBody () const [inline]`

Accessor for the body data of this frame.

Returns

char pointer to body data

6.518.3.5 `std::vector<unsigned char>& activemq::wireformat::stomp::StompFrame::getBody () [inline]`

Non-const version of the body accessor.

6.518.3.6 `std::size_t activemq::wireformat::stomp::StompFrame::getBodyLength () const [inline]`

Return the number of bytes contained in this frames body.

Returns

Body bytes length.

6.518.3.7 `const std::string& activemq::wireformat::stomp::StompFrame::getCommand () const [inline]`

Accessor for this frame's command field.

6.518.3.8 `decaf::util::Properties& activemq::wireformat::stomp::StompFrame::getProperties () [inline]`

Gets access to the header properties for this frame.

Returns

the Properties object owned by this Frame

6.518.3.9 **const decaf::util::Properties& activemq::wireformat-
::stomp::StompFrame::getProperties () const**
[inline]

6.518.3.10 **std::string activemq::wireformat::stomp::StompFrame::getProperty (**
const std::string & name, const std::string & fallback = " ") const [inline]

Gets a property from this Frame's properties and returns it, or the default value given.

Parameters

<i>name</i>	- The name of the property to lookup
<i>fallback</i>	- The default value to return if this value isn't set

Returns

string value of the property asked for.

6.518.3.11 **bool activemq::wireformat::stomp::StompFrame::hasProperty (const**
std::string & name) const [inline]

Checks if the given property is present in the Frame.

Parameters

<i>name</i>	- The name of the property to check for.
-------------	--

6.518.3.12 **std::string activemq::wireformat::stomp::StompFrame::removeProperty**
(const std::string & name) [inline]

Gets and remove the property specified, if the property is not set, this method returns the empty string.

Parameters

<i>name</i>	- the Name of the property to get and return.
-------------	---

6.518.3.13 **void activemq::wireformat::stomp::StompFrame::setBody (const**
unsigned char * bytes, std::size_t numBytes)

Sets the body data of this frame as a byte sequence.

Parameters

<i>bytes</i>	The byte buffer to be set in the body.
<i>numBytes</i>	The number of bytes in the buffer.

6.518.3.14 `void activemq::wireformat::stomp::StompFrame::setCommand (const std::string & cmd) [inline]`

Sets the command for this stomp frame.

Parameters

<i>cmd</i>	command The command to be set.
------------	--------------------------------

6.518.3.15 `void activemq::wireformat::stomp::StompFrame::setProperty (const std::string & name, const std::string & value) [inline]`

Sets the property given to the value specified in this Frame's Properties.

Parameters

<i>name</i>	- Name of the property.
<i>value</i>	- Value to set the property to.

6.518.3.16 `void activemq::wireformat::stomp::StompFrame::toStream (decaf::io::DataOutputStream * stream) const`

Writes this Frame to an OutputStream in the Stomp Wire Format.

Parameters

<i>stream</i>	- The stream to write the Frame to.
---------------	-------------------------------------

Exceptions

<i>IOException</i>	if an error occurs while reading the Frame.
--------------------	---

The documentation for this class was generated from the following file:

- `src/main/activemq/wireformat/stomp/StompFrame.h`

6.519 activemq::wireformat::stomp::StompHelper Class Reference

Utility Methods used when marshaling to and from **StompFrame** (p. 2587)'s.

```
#include <src/main/activemq/wireformat/stomp/StompHelper.h>
```

Public Member Functions

- **StompHelper** ()
- virtual **~StompHelper** ()
- void **convertProperties** (const **Pointer**< **StompFrame** > &frame, const **Pointer**< **Message** > &message)
Converts the Headers in a Stomp Frame into Headers in the given Message - Command.
- void **convertProperties** (const **Pointer**< **Message** > &message, const **Pointer**< **StompFrame** > &frame)
*Converts the Properties in a Message Command to Valid Headers and Properties in the **StompFrame** (p. 2587).*
- **Pointer**< **ActiveMQDestination** > **convertDestination** (const std::string &destination)
Converts from a Stomp Destination to an ActiveMQDestination.
- std::string **convertDestination** (const **Pointer**< **ActiveMQDestination** > &destination)
Converts from a ActiveMQDestination to a Stomp Destination Name.
- std::string **convertMessageId** (const **Pointer**< **MessageId** > &messageId)
Converts a MessageId instance to a Stomp MessageId String.
- **Pointer**< **MessageId** > **convertMessageId** (const std::string &messageId)
Converts a Stomp MessageId string to a MessageId.
- std::string **convertConsumerId** (const **Pointer**< **ConsumerId** > &consumerId)
Converts a ConsumerId instance to a Stomp ConsumerId String.
- **Pointer**< **ConsumerId** > **convertConsumerId** (const std::string &consumerId)
Converts a Stomp ConsumerId string to a ConsumerId.
- std::string **convertProducerId** (const **Pointer**< **ProducerId** > &producerId)
Converts a ProducerId instance to a Stomp ProducerId String.
- **Pointer**< **ProducerId** > **convertProducerId** (const std::string &producerId)
Converts a Stomp ProducerId string to a ProducerId.
- std::string **convertTransactionId** (const **Pointer**< **TransactionId** > &transactionId)
Converts a TransactionId instance to a Stomp TransactionId String.
- **Pointer**< **TransactionId** > **convertTransactionId** (const std::string &transactionId)
Converts a Stomp TransactionId string to a TransactionId.

6.519.1 Detailed Description

Utility Methods used when marshaling to and from **StompFrame** (p. 2587)'s.

Since

3.0

6.519.2 Constructor & Destructor Documentation

6.519.2.1 **activemq::wireformat::stomp::StompHelper::StompHelper ()**

6.519.2.2 **virtual activemq::wireformat::stomp::StompHelper::~~StompHelper ()**
[virtual]

6.519.3 Member Function Documentation

6.519.3.1 **std::string activemq::wireformat::stomp::StompHelper::convert-ConsumerId (const Pointer< ConsumerId > & consumerId)**

Converts a ConsumerId instance to a Stomp ConsumerId String.

Parameters

<i>consumerId</i>	- the Consumer instance to convert.
-------------------	-------------------------------------

Returns

a Stomp Consumer Id String.

6.519.3.2 **Pointer<ConsumerId> activemq::wireformat::stomp::StompHelper::convertConsumerId (const std::string & consumerId)**

Converts a Stomp ConsumerId string to a ConsumerId.

Parameters

<i>consumerId</i>	- the String Consumer Id to convert.
-------------------	--------------------------------------

Returns

Pointer to a new ConsumerId.

6.519.3.3 **Pointer<ActiveMQDestination> activemq::wireformat::stomp::StompHelper::convertDestination (const std::string & destination)**

Converts from a Stomp Destination to an ActiveMQDestination.

Parameters

<i>destination</i>	- The Stomp Destination name string.
--------------------	--------------------------------------

Returns

Pointer to a new ActiveMQDestination.

6.519.3.4 `std::string activemq::wireformat::stomp::StompHelper::convert-Destination (const Pointer< ActiveMQDestination > & destination)`

Converts from a ActiveMQDestination to a Stomp Destination Name.

Parameters

<i>destination</i>	- The ActiveMQDestination to Convert
--------------------	--------------------------------------

Returns

the Stomp String name that defines the destination.

6.519.3.5 `std::string activemq::wireformat::stomp::StompHelper::convertMessageld (const Pointer< Messageld > & messageld)`

Converts a Messageld instance to a Stomp Messageld String.

Parameters

<i>messageld</i>	- the Messageld instance to convert.
------------------	--------------------------------------

Returns

a Stomp Message Id String.

6.519.3.6 `Pointer<Messageld> activemq::wireformat::stomp::StompHelper::convertMessageld (const std::string & messageld)`

Converts a Stomp Messageld string to a Messageld.

Parameters

<i>messageld</i>	- the String message Id to convert.
------------------	-------------------------------------

Returns

Pointer to a new Messageld.

6.519.3.7 `std::string activemq::wireformat::stomp::StompHelper::convertProducerId (const Pointer< ProducerId > & producerId)`

Converts a ProducerId instance to a Stomp ProducerId String.

Parameters

<i>producerId</i>	- the Producer instance to convert.
-------------------	-------------------------------------

Returns

a Stomp Producer Id String.

6.519.3.8 `Pointer<ProducerId> activemq::wireformat::stomp::StompHelper::convertProducerId (const std::string & producerId)`

Converts a Stomp ProducerId string to a ProducerId.

Parameters

<i>producerId</i>	- the String Producer Id to convert.
-------------------	--------------------------------------

Returns

Pointer to a new ProducerId.

6.519.3.9 `void activemq::wireformat::stomp::StompHelper::convertProperties (const Pointer< StompFrame > & frame, const Pointer< Message > & message)`

Converts the Headers in a Stomp Frame into Headers in the given Message Command.

Parameters

<i>frame</i>	- The frame to extract headers from.
<i>message</i>	- The message to move the Headers to.

6.519.3.10 `void activemq::wireformat::stomp::StompHelper::convertProperties (const Pointer< Message > & message, const Pointer< StompFrame > & frame)`

Converts the Properties in a Message Command to Valid Headers and Properties in the **StompFrame** (p. 2587).

Parameters

<i>message</i>	- The message to move the Headers to.
<i>frame</i>	- The frame to extract headers from.

6.519.3.11 `std::string activemq::wireformat::stomp::StompHelper::convert-TransactionId (const Pointer< TransactionId > & transactionId)`

Converts a TransactionId instance to a Stomp TransactionId String.

Parameters

<i>transactionId</i>	- the Transaction instance to convert.
----------------------	--

Returns

a Stomp Transaction Id String.

6.519.3.12 `Pointer<TransactionId> activemq::wireformat::stomp::StompHelper::convertTransactionId (const std::string & transactionId)`

Converts a Stomp TransactionId string to a TransactionId.

Parameters

<i>transactionId</i>	- the String Transaction Id to convert.
----------------------	---

Returns

Pointer to a new TransactionId.

The documentation for this class was generated from the following file:

- src/main/activemq/wireformat/stomp/**StompHelper.h**

6.520 activemq::wireformat::stomp::StompWireFormat Class - Reference

```
#include <src/main/activemq/wireformat/stomp/StompWireFormat.h>
```

Inheritance diagram for `activemq::wireformat::stomp::StompWireFormat`:

Public Member Functions

- **StompWireFormat** ()
- virtual **~StompWireFormat** ()
- virtual void **marshal** (const **Pointer**< **commands::Command** > &command, const **activemq::transport::Transport** *transport, **decaf::io::DataOutputStream** *out)

Stream based marshaling of a Command, this method blocks until the entire Command has been written out to the output stream.
- virtual **Pointer** < **commands::Command** > **unmarshal** (const **activemq::transport::Transport** *transport, **decaf::io::DataInputStream** *in)

Stream based un-marshaling, blocks on reads on the input stream until a complete command has been read and unmarshaled into the correct form.
- virtual void **setVersion** (int version AMQCPP_UNUSED)

Set the Version.
- virtual int **getVersion** () const

Get the Version.
- virtual bool **inReceive** () const

Is there a Message being unmarshaled?
- virtual bool **hasNegotiator** () const

*Returns true if this **WireFormat** (p. 2884) has a Negotiator that needs to wrap the Transport that uses it.*
- virtual **Pointer** < **transport::Transport** > **createNegotiator** (const **Pointer**< **transport::Transport** > &transport)

If the Transport Provides a Negotiator this method will create and return a news instance of the Negotiator.

6.520.1 Constructor & Destructor Documentation

6.520.1.1 **activemq::wireformat::stomp::StompWireFormat::StompWireFormat** ()

6.520.1.2 virtual **activemq::wireformat::stomp::StompWireFormat::~~StompWireFormat** () [virtual]

6.520.2 Member Function Documentation

6.520.2.1 virtual **Pointer**<**transport::Transport**> **activemq::wireformat::stomp::StompWireFormat::createNegotiator** (const **Pointer**< **transport::Transport** > & *transport*) [virtual]

If the Transport Provides a Negotiator this method will create and return a news instance of the Negotiator.

Returns

new instance of a **WireFormatNegotiator** (p. 2904).

Exceptions

<i>Unsupported-OperationException</i>	if the WireFormat (p. 2884) doesn't have a Negotiator.
---------------------------------------	---

Implements **activemq::wireformat::WireFormat** (p. 2886).

6.520.2.2 `virtual int activemq::wireformat::stomp::StompWireFormat::getVersion () const [inline, virtual]`

Get the Version.

Returns

the version of the wire format

Implements **activemq::wireformat::WireFormat** (p. 2886).

6.520.2.3 `virtual bool activemq::wireformat::stomp::StompWireFormat::hasNegotiator () const [inline, virtual]`

Returns true if this **WireFormat** (p. 2884) has a Negotiator that needs to wrap the - Transport that uses it.

Returns

true if the **WireFormat** (p. 2884) provides a Negotiator.

Implements **activemq::wireformat::WireFormat** (p. 2886).

6.520.2.4 `virtual bool activemq::wireformat::stomp::StompWireFormat::inReceive () const [inline, virtual]`

Is there a Message being unmarshaled?

Returns

true while in the doUnmarshal method.

Implements **activemq::wireformat::WireFormat** (p. 2887).

6.520.2.5 `virtual void activemq::wireformat::stomp::StompWireFormat::marshal (const Pointer< commands::Command > & command, const activemq::transport::Transport * transport, decaf::io::DataOutputStream * out) [virtual]`

Stream based marshaling of a Command, this method blocks until the entire Command has been written out to the output stream.

Parameters

<i>command</i>	The Command to Marshal to the output stream.
<i>transport</i>	The Transport that initiated this marshal call.
<i>out</i>	The output stream to write the command to.

Exceptions

<i>IOException</i>	
--------------------	--

Implements **activemq::wireformat::WireFormat** (p. 2887).

6.520.2.6 virtual void **activemq::wireformat::stomp::StompWireFormat::setVersion** (int version *AMQCPP_UNUSED*) [*inline, virtual*]

Set the Version.

Parameters

<i>the</i>	version of the wire format
------------	----------------------------

Implements **activemq::wireformat::WireFormat** (p. 2887).

6.520.2.7 virtual **Pointer<commands::Command>** **activemq::wireformat::stomp::StompWireFormat::unmarshal** (const **activemq::transport::Transport** * *transport*, **decaf::io::DataInputStream** * *in*) [*virtual*]

Stream based un-marshaling, blocks on reads on the input stream until a complete command has been read and unmarshaled into the correct form.

Returns a Pointer to the newly unmarshaled Command.

Parameters

<i>transport</i>	- Pointer to the transport that is making this request.
<i>in</i>	- the input stream to read the command from.

Returns

the newly marshaled Command, caller owns the pointer

Exceptions

<i>IOException</i>	
--------------------	--

Implements **activemq::wireformat::WireFormat** (p. 2888).

The documentation for this class was generated from the following file:

- src/main/activemq/wireformat/stomp/**StompWireFormat.h**

6.521 activemq::wireformat::stomp::StompWireFormatFactory - Class Reference

Factory used to create the Stomp Wire Format instance.

```
#include <src/main/activemq/wireformat/stomp/StompWire-  
FormatFactory.h>
```

Inheritance diagram for activemq::wireformat::stomp::StompWireFormatFactory:

Public Member Functions

- **StompWireFormatFactory** ()
- virtual **~StompWireFormatFactory** ()
- virtual **Pointer< WireFormat > createWireFormat** (const **decaf::util::Properties** &properties)

*Creates a new **WireFormat** (p. 2884) Object passing it a set of properties from which it can obtain any optional settings.*

6.521.1 Detailed Description

Factory used to create the Stomp Wire Format instance.

6.521.2 Constructor & Destructor Documentation

6.521.2.1 **activemq::wireformat::stomp::StompWireFormatFactory::StompWireFormatFactory** () [inline]

6.521.2.2 **virtual activemq::wireformat::stomp::StompWireFormatFactory::~~StompWireFormatFactory** () [inline, virtual]

6.521.3 Member Function Documentation

6.521.3.1 **virtual Pointer<WireFormat> activemq::wireformat::stomp::StompWireFormatFactory::createWireFormat** (const **decaf::util::Properties** & *properties*) [virtual]

Creates a new **WireFormat** (p. 2884) Object passing it a set of properties from which it can obtain any optional settings.

Parameters

<i>properties</i>	- the Properties for this WireFormat (p. 2884)
-------------------	---

Implements **activemq::wireformat::WireFormatFactory** (p. 2889).

The documentation for this class was generated from the following file:

- src/main/activemq/wireformat/stomp/**StompWireFormatFactory.h**

6.522 cms::Stoppable Class Reference

Interface for a class that implements the stop method.

```
#include <src/main/cms/Stoppable.h>
```

Inheritance diagram for cms::Stoppable:

Public Member Functions

- virtual **~Stoppable** ()
- virtual void **stop** ()=0
Stops this service.

6.522.1 Detailed Description

Interface for a class that implements the stop method.

An object that implements this interface implies that it will halt all operations that result in events being propagated to external users, internally the Object can continue to process data but not events will be generated to clients and methods that return data will not return valid results until the object is started again.

Since

1.0

6.522.2 Constructor & Destructor Documentation

6.522.2.1 virtual cms::Stoppable::~**~Stoppable** () [virtual]

6.522.3 Member Function Documentation

6.522.3.1 virtual void cms::Stoppable::stop () [pure virtual]

Stops this service.

Exceptions

CMSException (p. 826)	- if an internal error occurs while stopping the Service.
---------------------------------	---

Implemented in **activemq::core::ActiveMQConnection** (p.212), **activemq::core::ActiveMQSession** (p.352), **activemq::core::ActiveMQConsumer** (p.244), **activemq::cmsutil::PooledSession** (p.2106), and **activemq::cmsutil::CachedConsumer** (p.738).

Referenced by **activemq::cmsutil::PooledSession::stop()**.

The documentation for this class was generated from the following file:

- **src/main/cms/Stopable.h**

6.523 decaf::util::logging::StreamHandler Class Reference

Stream based logging **Handler** (p. 1401).

```
#include <src/main/decaf/util/logging/StreamHandler.h>
```

Inheritance diagram for **decaf::util::logging::StreamHandler**:

Public Member Functions

- **StreamHandler ()**
*Create a **StreamHandler** (p. 2603), with no current output stream.*
- **StreamHandler (decaf::io::OutputStream *stream, Formatter *formatter)**
*Create a **StreamHandler** (p. 2603), with no current output stream.*
- virtual **~StreamHandler ()**
- virtual void **close ()**
Close the current output stream.
- virtual void **flush ()**
*Flush the **Handler** (p. 1401)'s output, clears any buffers.*
- virtual void **publish (const LogRecord &record)**
*Publish the Log Record to this **Handler** (p. 1401).*
- virtual bool **isLoggable (const LogRecord &record) const**
*Check if this **Handler** (p. 1401) would actually log a given **LogRecord** (p. 1719).*

Protected Member Functions

- virtual void **setOutputStream (decaf::io::OutputStream *stream)**
Change the output stream.

- void **close** (bool closeStream)

Closes this handler, but the underlying output stream is only closed if closeStream is true.

6.523.1 Detailed Description

Stream based logging **Handler** (p. 1401).

This is primarily intended as a base class or support class to be used in implementing other logging Handlers.

LogRecords are published to a given **decaf::io::OutputStream** (p. 2066).

Configuration: By default each **StreamHandler** (p. 2603) is initialized using the following **LogManager** (p. 1712) configuration properties. If properties are not defined (or have invalid values) then the specified default values are used.

* decaf.util.logging.StreamHandler.level specifies the default level for the **Handler** (p. 1401) (defaults to **Level.INFO** (p. 1620)). * decaf.util.logging.StreamHandler.filter specifies the name of a **Filter** (p. 1333) class to use (defaults to no **Filter** (p. 1333)). * decaf.util.logging.StreamHandler.formatter specifies the name of a **Formatter** (p. 1387) class to use (defaults to **decaf.util.logging.SimpleFormatter** (p. 2441)).

Since

1.0

6.523.2 Constructor & Destructor Documentation

6.523.2.1 decaf::util::logging::StreamHandler::StreamHandler ()

Create a **StreamHandler** (p. 2603), with no current output stream.

6.523.2.2 decaf::util::logging::StreamHandler::StreamHandler (decaf::io::OutputStream * stream, Formatter * formatter)

Create a **StreamHandler** (p. 2603), with no current output stream.

6.523.2.3 virtual decaf::util::logging::StreamHandler::~~StreamHandler () [virtual]

6.523.3 Member Function Documentation

6.523.3.1 virtual void decaf::util::logging::StreamHandler::close () [virtual]

Close the current output stream.

The close method will perform a flush and then close the **Handler** (p. 1401). After close has been called this **Handler** (p. 1401) should no longer be used. Method calls may either be silently ignored or may throw runtime exceptions.

Exceptions

<i>IOException</i>	if an I/O error occurs.
--------------------	-------------------------

Implements **decaf::io::Closeable** (p. 817).

Reimplemented in **decaf::util::logging::ConsoleHandler** (p. 991).

6.523.3.2 void **decaf::util::logging::StreamHandler::close** (bool *closeStream*)
[protected]

Closes this handler, but the underlying output stream is only closed if closeStream is true.

Parameters

<i>closeStream</i>	whether to close the underlying output stream.
--------------------	--

6.523.3.3 virtual void **decaf::util::logging::StreamHandler::flush** () [virtual]

Flush the **Handler** (p. 1401)'s output, clears any buffers.

Implements **decaf::util::logging::Handler** (p. 1403).

6.523.3.4 virtual bool **decaf::util::logging::StreamHandler::isLoggable** (const **LogRecord** & *record*) const [virtual]

Check if this **Handler** (p. 1401) would actually log a given **LogRecord** (p. 1719).

Parameters

<i>record</i>	LogRecord (p. 1719) to check
---------------	-------------------------------------

Returns

true if the record can be logged with current settings.

Reimplemented from **decaf::util::logging::Handler** (p. 1404).

6.523.3.5 virtual void **decaf::util::logging::StreamHandler::publish** (const **LogRecord** & *record*) [virtual]

Publish the Log Record to this **Handler** (p. 1401).

Parameters

<i>record</i>	The LogRecord (p. 1719) to Publish
---------------	---

Implements **decaf::util::logging::Handler** (p. 1404).

Reimplemented in **decaf::util::logging::ConsoleHandler** (p. 991).

6.523.3.6 virtual void **decaf::util::logging::StreamHandler::setOutputStream** (
decaf::io::OutputStream * stream) [protected, virtual]

Change the output stream.

If there is a current output stream then the **Formatter** (p. 1387)'s tail string is written and the stream is flushed and closed. Then the output stream is replaced with the new output stream.

Parameters

<i>stream</i>	The new output stream. May not be NULL.
---------------	---

Exceptions

<i>NullPointerException</i>	if the passed stream is NULL.
-----------------------------	-------------------------------

The documentation for this class was generated from the following file:

- src/main/decaf/util/logging/**StreamHandler.h**

6.524 cms::StreamMessage Class Reference

Interface for a **StreamMessage** (p. 2606).

```
#include <src/main/cms/StreamMessage.h>
```

Inheritance diagram for cms::StreamMessage:

Public Member Functions

- virtual **~StreamMessage** () throw ()
- virtual bool **readBoolean** () const =0
Reads a Boolean from the Stream message stream.
- virtual void **writeBoolean** (bool value)=0
Writes a boolean to the Stream message stream as a 1-byte value.
- virtual unsigned char **readByte** () const =0
Reads a Byte from the Stream message stream.

- virtual void **writeByte** (unsigned char value)=0
Writes a byte to the Stream message stream as a 1-byte value.
- virtual int **readBytes** (std::vector< unsigned char > &value) const =0
Reads a byte array from the Stream message stream.
- virtual void **writeBytes** (const std::vector< unsigned char > &value)=0
Writes a byte array to the Stream message stream using the vector size as the number of bytes to write.
- virtual int **readBytes** (unsigned char *buffer, int length) const =0
Reads a portion of the Stream message stream.
- virtual void **writeBytes** (const unsigned char *value, int offset, int length)=0
Writes a portion of a byte array to the Stream message stream.
- virtual char **readChar** () const =0
Reads a Char from the Stream message stream.
- virtual void **writeChar** (char value)=0
Writes a char to the Stream message stream as a 1-byte value.
- virtual float **readFloat** () const =0
Reads a 32 bit float from the Stream message stream.
- virtual void **writeFloat** (float value)=0
Writes a float to the Stream message stream as a 4 byte value.
- virtual double **readDouble** () const =0
Reads a 64 bit double from the Stream message stream.
- virtual void **writeDouble** (double value)=0
Writes a double to the Stream message stream as a 8 byte value.
- virtual short **readShort** () const =0
Reads a 16 bit signed short from the Stream message stream.
- virtual void **writeShort** (short value)=0
Writes a signed short to the Stream message stream as a 2 byte value.
- virtual unsigned short **readUnsignedShort** () const =0
Reads a 16 bit unsigned short from the Stream message stream.
- virtual void **writeUnsignedShort** (unsigned short value)=0
Writes a unsigned short to the Stream message stream as a 2 byte value.
- virtual int **readInt** () const =0
Reads a 32 bit signed integer from the Stream message stream.
- virtual void **writeInt** (int value)=0
Writes a signed int to the Stream message stream as a 4 byte value.
- virtual long long **readLong** () const =0
Reads a 64 bit long from the Stream message stream.
- virtual void **writeLong** (long long value)=0
Writes a long long to the Stream message stream as a 8 byte value.
- virtual std::string **readString** () const =0
Reads an ASCII String from the Stream message stream.
- virtual void **writeString** (const std::string &value)=0
Writes an ASCII String to the Stream message stream.

6.524.1 Detailed Description

Interface for a **StreamMessage** (p. 2606).

The stream Messages provides a **Message** (p. 1839) type whose body is a stream of self describing primitive types. The primitive values are read and written using accessors specific to the given types.

StreamMessage (p. 2606) objects support the following conversion table. The marked cases must be supported. The unmarked cases must throw a **CMSException** (p. 826). The string-to- primitive conversions may throw a runtime exception if the primitive's `valueOf()` method does not accept it as a valid String representation of the primitive.

A value written as the row type can be read as the column type.

	boolean	byte	short	char	int	long	float	double	String	byte[]
boolean	X									X
byte		X	X		X	X				X
short			X		X	X				X
char				X						X
int					X	X				X
long						X				X
float							X	X		X
double								X	X	X
String	X	X	X		X	X	X	X	X	X
byte[]										X

Since

1.3

6.524.2 Constructor & Destructor Documentation

6.524.2.1 `virtual cms::StreamMessage::~StreamMessage () throw ()` [virtual]

6.524.3 Member Function Documentation

6.524.3.1 `virtual bool cms::StreamMessage::readBoolean () const` [pure virtual]

Reads a Boolean from the Stream message stream.

Returns

boolean value from stream

Exceptions

<i>CMSException</i> (p. 826)	- if the CMS provider fails to read the message due to some internal error.
<i>MessageEOF-Exception</i> (p. 1905)	- if unexpected end of message stream has been reached.
<i>MessageFormat-Exception</i> (p. 1906)	- if this type conversion is invalid.
<i>MessageNot-ReadableException</i> (p. 1922)	- if the message is in write-only mode.

Implemented in **activemq::commands::ActiveMQStreamMessage** (p. 363).

6.524.3.2 `virtual unsigned char cms::StreamMessage::readByte () const [pure virtual]`

Reads a Byte from the Stream message stream.

Returns

unsigned char value from stream

Exceptions

<i>CMSException</i> (p. 826)	- if the CMS provider fails to read the message due to some internal error.
<i>MessageEOF-Exception</i> (p. 1905)	- if unexpected end of message stream has been reached.
<i>MessageFormat-Exception</i> (p. 1906)	- if this type conversion is invalid.
<i>MessageNot-ReadableException</i> (p. 1922)	- if the message is in write-only mode.

Implemented in **activemq::commands::ActiveMQStreamMessage** (p. 363).

6.524.3.3 `virtual int cms::StreamMessage::readBytes (std::vector< unsigned char > &value) const [pure virtual]`

Reads a byte array from the Stream message stream.

If the length of vector value is less than the number of bytes remaining to be read from the stream, the vector should be filled. A subsequent call reads the next increment, and so on.

If the number of bytes remaining in the stream is less than the length of vector value, the bytes should be read into the vector. The return value of the total number of bytes read will be less than the length of the vector, indicating that there are no more bytes left to be read from the stream. The next read of the stream returns -1.

Parameters

<i>value</i>	buffer to place data in
--------------	-------------------------

Returns

the total number of bytes read into the buffer, or -1 if there is no more data because the end of the stream has been reached

Exceptions

CMSException (p. 826)	- if the CMS provider fails to read the message due to some internal error.
MessageEOF-Exception (p. 1905)	- if unexpected end of message stream has been reached.
MessageFormat-Exception (p. 1906)	- if this type conversion is invalid.
MessageNot-ReadableException (p. 1922)	- if the message is in write-only mode.

Implemented in **activemq::commands::ActiveMQStreamMessage** (p. 364).

6.524.3.4 virtual int cms::StreamMessage::readBytes (unsigned char * *buffer*, int *length*) const [pure virtual]

Reads a portion of the Stream message stream.

If the length of array value is less than the number of bytes remaining to be read from the stream, the array should be filled. A subsequent call reads the next increment, and so on.

If the number of bytes remaining in the stream is less than the length of array value, the bytes should be read into the array. The return value of the total number of bytes read will be less than the length of the array, indicating that there are no more bytes left to be read from the stream. The next read of the stream returns -1.

If length is negative, or length is greater than the length of the array value, then an **CMSException** (p. 826) is thrown. No bytes will be read from the stream for this exception case.

Parameters

<i>buffer</i>	the buffer into which the data is read
<i>length</i>	the number of bytes to read; must be less than or equal to value.length

Returns

the total number of bytes read into the buffer, or -1 if there is no more data because the end of the stream has been reached

Exceptions

<i>CMSException</i> (p. 826)	- if the CMS provider fails to read the message due to some internal error.
<i>MessageEOF-Exception</i> (p. 1905)	- if unexpected end of message stream has been reached.
<i>MessageFormatException</i> (p. 1906)	- if this type conversion is invalid.
<i>MessageNotReadableException</i> (p. 1922)	- if the message is in write-only mode.

Implemented in **activemq::commands::ActiveMQStreamMessage** (p. 365).

6.524.3.5 virtual char **cms::StreamMessage::readChar** () const [pure virtual]

Reads a Char from the Stream message stream.

Returns

char value from stream

Exceptions

<i>CMSException</i> (p. 826)	- if the CMS provider fails to read the message due to some internal error.
<i>MessageEOF-Exception</i> (p. 1905)	- if unexpected end of message stream has been reached.
<i>MessageFormatException</i> (p. 1906)	- if this type conversion is invalid.
<i>MessageNotReadableException</i> (p. 1922)	- if the message is in write-only mode.

Implemented in **activemq::commands::ActiveMQStreamMessage** (p. 365).

6.524.3.6 virtual double cms::StreamMessage::readDouble () const [pure virtual]

Reads a 64 bit double from the Stream message stream.

Returns

double value from stream

Exceptions

CMSException (p. 826)	- if the CMS provider fails to read the message due to some internal error.
MessageEOF-Exception (p. 1905)	- if unexpected end of message stream has been reached.
MessageFormatException (p. 1906)	- if this type conversion is invalid.
MessageNotReadableException (p. 1922)	- if the message is in write-only mode.

Implemented in **activemq::commands::ActiveMQStreamMessage** (p. 366).

6.524.3.7 virtual float cms::StreamMessage::readFloat () const [pure virtual]

Reads a 32 bit float from the Stream message stream.

Returns

double value from stream

Exceptions

CMSException (p. 826)	- if the CMS provider fails to read the message due to some internal error.
MessageEOF-Exception (p. 1905)	- if unexpected end of message stream has been reached.
MessageFormatException (p. 1906)	- if this type conversion is invalid.
MessageNotReadableException (p. 1922)	- if the message is in write-only mode.

Implemented in `activemq::commands::ActiveMQStreamMessage` (p. 366).

6.524.3.8 `virtual int cms::StreamMessage::readInt () const` [pure virtual]

Reads a 32 bit signed integer from the Stream message stream.

Returns

int value from stream

Exceptions

<i>CMSException</i> (p. 826)	- if the CMS provider fails to read the message due to some internal error.
<i>MessageEOF-Exception</i> (p. 1905)	- if unexpected end of message stream has been reached.
<i>MessageFormat-Exception</i> (p. 1906)	- if this type conversion is invalid.
<i>MessageNot-ReadableException</i> (p. 1922)	- if the message is in write-only mode.

Implemented in `activemq::commands::ActiveMQStreamMessage` (p. 367).

6.524.3.9 `virtual long long cms::StreamMessage::readLong () const` [pure virtual]

Reads a 64 bit long from the Stream message stream.

Returns

long long value from stream

Exceptions

<i>CMSException</i> (p. 826)	- if the CMS provider fails to read the message due to some internal error.
<i>MessageEOF-Exception</i> (p. 1905)	- if unexpected end of message stream has been reached.
<i>MessageFormat-Exception</i> (p. 1906)	- if this type conversion is invalid.
<i>MessageNot-ReadableException</i> (p. 1922)	- if the message is in write-only mode.

Implemented in **activemq::commands::ActiveMQStreamMessage** (p. 367).

6.524.3.10 virtual short cms::StreamMessage::readShort () const [pure virtual]

Reads a 16 bit signed short from the Stream message stream.

Returns

short value from stream

Exceptions

CMSException (p. 826)	- if the CMS provider fails to read the message due to some internal error.
MessageEOF-Exception (p. 1905)	- if unexpected end of message stream has been reached.
MessageFormatException (p. 1906)	- if this type conversion is invalid.
MessageNotReadableException (p. 1922)	- if the message is in write-only mode.

Implemented in **activemq::commands::ActiveMQStreamMessage** (p. 368).

6.524.3.11 virtual std::string cms::StreamMessage::readString () const [pure virtual]

Reads an ASCII String from the Stream message stream.

Returns

String from stream

Exceptions

CMSException (p. 826)	- if the CMS provider fails to read the message due to some internal error.
MessageEOF-Exception (p. 1905)	- if unexpected end of message stream has been reached.
MessageFormatException (p. 1906)	- if this type conversion is invalid.
MessageNotReadableException (p. 1922)	- if the message is in write-only mode.

Implemented in **activemq::commands::ActiveMQStreamMessage** (p. 368).

6.524.3.12 `virtual unsigned short cms::StreamMessage::readUnsignedShort () const`
`[pure virtual]`

Reads a 16 bit unsigned short from the Stream message stream.

Returns

unsigned short value from stream

Exceptions

<i>CMSException</i> (p. 826)	- if the CMS provider fails to read the message due to some internal error.
<i>MessageEOF-Exception</i> (p. 1905)	- if unexpected end of message stream has been reached.
<i>MessageFormat-Exception</i> (p. 1906)	- if this type conversion is invalid.
<i>MessageNot-ReadableException</i> (p. 1922)	- if the message is in write-only mode.

Implemented in **activemq::commands::ActiveMQStreamMessage** (p. 369).

6.524.3.13 `virtual void cms::StreamMessage::writeBoolean (bool value)` `[pure virtual]`

Writes a boolean to the Stream message stream as a 1-byte value.

The value true is written as the value (byte)1; the value false is written as the value (byte)0.

Parameters

<i>value</i>	boolean to write to the stream
--------------	--------------------------------

Exceptions

<i>CMSException</i> (p. 826)	- if the CMS provider fails to write the message due to some internal error.
<i>MessageNot-WriteableException</i> (p. 1923)	- if the message is in read-only mode.

Implemented in **activemq::commands::ActiveMQStreamMessage** (p. 370).

6.524.3.14 virtual void **cms::StreamMessage::writeByte** (unsigned char *value*)
[pure virtual]

Writes a byte to the Stream message stream as a 1-byte value.

Parameters

<i>value</i>	byte to write to the stream
--------------	-----------------------------

Exceptions

<i>CMSException</i> (p. 826)	- if the CMS provider fails to write the message due to some internal error.
<i>MessageNot-WriteableException</i> (p. 1923)	- if the message is in read-only mode.

Implemented in **activemq::commands::ActiveMQStreamMessage** (p. 370).

6.524.3.15 virtual void **cms::StreamMessage::writeBytes** (const std::vector< unsigned char > & *value*) [pure virtual]

Writes a byte array to the Stream message stream using the vector size as the number of bytes to write.

Parameters

<i>value</i>	bytes to write to the stream
--------------	------------------------------

Exceptions

<i>CMSException</i> (p. 826)	- if the CMS provider fails to write the message due to some internal error.
<i>MessageNot-WriteableException</i> (p. 1923)	- if the message is in read-only mode.

Implemented in **activemq::commands::ActiveMQStreamMessage** (p. 370).

6.524.3.16 virtual void **cms::StreamMessage::writeBytes** (const unsigned char * *value*, int *offset*, int *length*) [pure virtual]

Writes a portion of a byte array to the Stream message stream.
size as the number of bytes to write.

Parameters

<i>value</i>	bytes to write to the stream
<i>offset</i>	the initial offset within the byte array
<i>length</i>	the number of bytes to use

Exceptions

<i>CMSException</i> (p. 826)	- if the CMS provider fails to write the message due to some internal error.
<i>MessageNotWriteableException</i> (p. 1923)	- if the message is in read-only mode.

Implemented in **activemq::commands::ActiveMQStreamMessage** (p. 371).

6.524.3.17 `virtual void cms::StreamMessage::writeChar (char value)` [pure virtual]

Writes a char to the Stream message stream as a 1-byte value.

Parameters

<i>value</i>	char to write to the stream
--------------	-----------------------------

Exceptions

<i>CMSException</i> (p. 826)	- if the CMS provider fails to write the message due to some internal error.
<i>MessageNotWriteableException</i> (p. 1923)	- if the message is in read-only mode.

Implemented in **activemq::commands::ActiveMQStreamMessage** (p. 371).

6.524.3.18 `virtual void cms::StreamMessage::writeDouble (double value)` [pure virtual]

Writes a double to the Stream message stream as a 8 byte value.

Parameters

<i>value</i>	double to write to the stream
--------------	-------------------------------

Exceptions

<i>CMSException</i> (p. 826)	- if the CMS provider fails to write the message due to some internal error.
--	--

<i>MessageNotWriteableException</i> (p. 1923)	- if the message is in read-only mode.
---	--

Implemented in **activemq::commands::ActiveMQStreamMessage** (p. 372).

6.524.3.19 virtual void **cms::StreamMessage::writeFloat** (float *value*) [pure virtual]

Writes a float to the Stream message stream as a 4 byte value.

Parameters

<i>value</i>	float to write to the stream
--------------	------------------------------

Exceptions

<i>CMSException</i> (p. 826)	- if the CMS provider fails to write the message due to some internal error.
<i>MessageNotWriteableException</i> (p. 1923)	- if the message is in read-only mode.

Implemented in **activemq::commands::ActiveMQStreamMessage** (p. 372).

6.524.3.20 virtual void **cms::StreamMessage::writeInt** (int *value*) [pure virtual]

Writes a signed int to the Stream message stream as a 4 byte value.

Parameters

<i>value</i>	signed int to write to the stream
--------------	-----------------------------------

Exceptions

<i>CMSException</i> (p. 826)	- if the CMS provider fails to write the message due to some internal error.
<i>MessageNotWriteableException</i> (p. 1923)	- if the message is in read-only mode.

Implemented in **activemq::commands::ActiveMQStreamMessage** (p. 372).

6.524.3.21 `virtual void cms::StreamMessage::writeLong (long long value)` [pure virtual]

Writes a long long to the Stream message stream as a 8 byte value.

Parameters

<i>value</i>	signed long long to write to the stream
--------------	---

Exceptions

<i>CMSException</i> (p. 826)	- if the CMS provider fails to write the message due to some internal error.
<i>MessageNot-WriteableException</i> (p. 1923)	- if the message is in read-only mode.

Implemented in `activemq::commands::ActiveMQStreamMessage` (p. 373).

6.524.3.22 `virtual void cms::StreamMessage::writeShort (short value)` [pure virtual]

Writes a signed short to the Stream message stream as a 2 byte value.

Parameters

<i>value</i>	signed short to write to the stream
--------------	-------------------------------------

Exceptions

<i>CMSException</i> (p. 826)	- if the CMS provider fails to write the message due to some internal error.
<i>MessageNot-WriteableException</i> (p. 1923)	- if the message is in read-only mode.

Implemented in `activemq::commands::ActiveMQStreamMessage` (p. 373).

6.524.3.23 `virtual void cms::StreamMessage::writeString (const std::string & value)` [pure virtual]

Writes an ASCII String to the Stream message stream.

Parameters

<i>value</i>	String to write to the stream
--------------	-------------------------------

Exceptions

<i>CMSException</i> (p. 826)	- if the CMS provider fails to write the message due to some internal error.
<i>MessageNotWriteableException</i> (p. 1923)	- if the message is in read-only mode.

Implemented in **activemq::commands::ActiveMQStreamMessage** (p. 374).

6.524.3.24 virtual void **cms::StreamMessage::writeUnsignedShort** (unsigned short *value*) [pure virtual]

Writes a unsigned short to the Stream message stream as a 2 byte value.

Parameters

<i>value</i>	unsigned short to write to the stream
--------------	---------------------------------------

Exceptions

<i>CMSException</i> (p. 826)	- if the CMS provider fails to write the message due to some internal error.
<i>MessageNotWriteableException</i> (p. 1923)	- if the message is in read-only mode.

Implemented in **activemq::commands::ActiveMQStreamMessage** (p. 374).

The documentation for this class was generated from the following file:

- src/main/cms/**StreamMessage.h**

6.525 decaf::lang::String Class Reference

The **String** (p. 2620) class represents an immutable sequence of chars.

```
#include <src/main/decaf/lang/String.h>
```

Inheritance diagram for decaf::lang::String:

Public Member Functions

- **String** ()
*Creates a new empty **String** (p. 2620) object.*
- **String** (const **String** &source)

Create a new **String** (p. 2620) object that represents the given STL string.

- **String** (const std::string &source)

Create a new **String** (p. 2620) object that represents the given STL string.

- **String** (const char *array, int size)

Create a new **String** (p. 2620) object that represents the given array of characters.

- **String** (const char *array, int size, int offset, int **length**)

Create a new **String** (p. 2620) object that represents the given array of characters.

- virtual ~**String** ()
- **String** & **operator=** (const **String** &)
- **String** & **operator=** (const std::string &)
- bool **isEmpty** () const
- virtual int **length** () const

Returns

the length of the underlying character sequence.

- virtual char **charAt** (int index) const

Returns the Char at the specified index so long as the index is not greater than the length of the sequence.

Parameters

index	The position to return the char at.
-------	-------------------------------------

Returns

the char at the given position.

Exceptions

IndexOutOfBoundsException	if index is > than length() (p. 804) or negative
---------------------------	---

- virtual **CharSequence** * **subSequence** (int start, int end) const

Returns a new **CharSequence** (p. 803) that is a subsequence of this sequence.

The subsequence starts with the char value at the specified index and ends with the char value at index end - 1. The length (in chars) of the returned sequence is end - start, so if start == end then an empty sequence is returned.

Parameters

start	The start index, inclusive.
end	The end index, exclusive.

Returns

a new **CharSequence** (p. 803)

Exceptions

IndexOutOfBoundsException	if start or end > length() (p. 804) or start or end are negative.
---------------------------	--

- virtual std::string **toString** () const

Returns

the string representation of this **CharSequence** (p. 803)

Static Public Member Functions

- static **String valueOf** (bool value)
Returns a **String** (p. 2620) that represents the value of the given boolean value.
- static **String valueOf** (char value)
Returns a **String** (p. 2620) that represents the value of the given char value.
- static **String valueOf** (float value)
Returns a **String** (p. 2620) that represents the value of the given float value.
- static **String valueOf** (double value)
Returns a **String** (p. 2620) that represents the value of the given double value.
- static **String valueOf** (short value)
Returns a **String** (p. 2620) that represents the value of the given short value.
- static **String valueOf** (int value)
Returns a **String** (p. 2620) that represents the value of the given integer value.
- static **String valueOf** (long long value)
Returns a **String** (p. 2620) that represents the value of the given 64bit long value.

6.525.1 Detailed Description

The **String** (p. 2620) class represents an immutable sequence of chars.

Since

1.0

6.525.2 Constructor & Destructor Documentation

6.525.2.1 decaf::lang::String::String ()

Creates a new empty **String** (p. 2620) object.

6.525.2.2 decaf::lang::String::String (const String & source)

Create a new **String** (p. 2620) object that represents the given STL string.

Parameters

source	The string to copy into this new String (p. 2620) object.
---------------	--

6.525.2.3 `decaf::lang::String::String (const std::string & source)`

Create a new **String** (p. 2620) object that represents the given STL string.

Parameters

<i>source</i>	The string to copy into this new String (p. 2620) object.
---------------	--

6.525.2.4 `decaf::lang::String::String (const char * array, int size)`

Create a new **String** (p. 2620) object that represents the given array of characters.

The method takes the size of the array as a parameter to allow for strings that are not N-NULL terminated, the caller can pass `strlen(array)` in the case where the array is properly NULL terminated.

Parameters

<i>array</i>	The character buffer to copy into this new String (p. 2620) object.
<i>size</i>	The size of the string buffer given, in case the string is not NULL terminated.

Exceptions

<i>NullPointerException</i>	if the character array parameter is NULL.
<i>IndexOutOfBoundsException</i>	if the size parameter is negative.

6.525.2.5 `decaf::lang::String::String (const char * array, int size, int offset, int length)`

Create a new **String** (p. 2620) object that represents the given array of characters.

The method takes the size of the array as a parameter to allow for strings that are not N-NULL terminated, the caller can pass `strlen(array)` in the case where the array is properly NULL terminated.

Parameters

<i>array</i>	The character buffer to copy into this new String (p. 2620) object.
<i>size</i>	The size of the string buffer given, in case the string is not NULL terminated.
<i>offset</i>	The position to start copying from in the given buffer.
<i>length</i>	The number of bytes to copy from the given buffer starting from the offset.

Exceptions

<i>NullPointerException</i>	if the character array parameter is NULL.
-----------------------------	---

<i>IndexOutOfBounds-Exception</i>	if the size, offset or length parameter is negative or if the length to copy is greater than the span of size - offset.
-----------------------------------	---

6.525.2.6 virtual **decaf::lang::String::~~String** () [virtual]

6.525.3 Member Function Documentation

6.525.3.1 virtual char **decaf::lang::String::charAt** (int *index*) const [virtual]

Returns the Char at the specified index so long as the index is not greater than the length of the sequence.

Parameters

<i>index</i>	The position to return the char at.
--------------	-------------------------------------

Returns

the char at the given position.

Exceptions

<i>IndexOutOfBounds-Exception</i>	if index is > than length() (p. 804) or negative
-----------------------------------	---

Implements **decaf::lang::CharSequence** (p. 804).

6.525.3.2 bool **decaf::lang::String::isEmpty** () const

Returns

true if the length of this **String** (p. 2620) is zero.

6.525.3.3 virtual int **decaf::lang::String::length** () const [virtual]

Returns

the length of the underlying character sequence.

Implements **decaf::lang::CharSequence** (p. 804).

6.525.3.4 **String&** **decaf::lang::String::operator=** (const **String** &)

6.525.3.5 **String&** **decaf::lang::String::operator=** (const std::string &)

6.525.3.6 `virtual CharSequence* decaf::lang::String::subSequence (int start, int end) const` `[virtual]`

Returns a new **CharSequence** (p. 803) that is a subsequence of this sequence.

The subsequence starts with the char value at the specified index and ends with the char value at index *end* - 1. The length (in chars) of the returned sequence is *end* - *start*, so if *start* == *end* then an empty sequence is returned.

Parameters

<i>start</i>	The start index, inclusive.
<i>end</i>	The end index, exclusive.

Returns

a new **CharSequence** (p. 803)

Exceptions

<i>IndexOutOfBoundsException</i>	if <i>start</i> or <i>end</i> > length() (p. 804) or <i>start</i> or <i>end</i> are negative.
----------------------------------	--

Implements **decaf::lang::CharSequence** (p. 804).

6.525.3.7 `virtual std::string decaf::lang::String::toString () const` `[virtual]`

Returns

the string representation of this **CharSequence** (p. 803)

Implements **decaf::lang::CharSequence** (p. 805).

6.525.3.8 `static String decaf::lang::String::valueOf (bool value)` `[static]`

Returns a **String** (p. 2620) that represents the value of the given boolean value.

Parameters

<i>value</i>	The value whose string representation is to be returned.
--------------	--

Returns

"true" if the boolean is true, "false" otherwise.

6.525.3.9 `static String decaf::lang::String::valueOf (char value)` `[static]`

Returns a **String** (p. 2620) that represents the value of the given char value.

Parameters

<i>value</i>	The value whose string representation is to be returned.
--------------	--

Returns

a **String** (p. 2620) that contains the single character value given.

6.525.3.10 static **String** decaf::lang::String::valueOf (float *value*) [static]

Returns a **String** (p. 2620) that represents the value of the given float value.

Parameters

<i>value</i>	The value whose string representation is to be returned.
--------------	--

Returns

a **String** (p. 2620) that contains the string representation of the float value given.

6.525.3.11 static **String** decaf::lang::String::valueOf (double *value*) [static]

Returns a **String** (p. 2620) that represents the value of the given double value.

Parameters

<i>value</i>	The value whose string representation is to be returned.
--------------	--

Returns

a **String** (p. 2620) that contains the string representation of the double value given.

6.525.3.12 static **String** decaf::lang::String::valueOf (short *value*) [static]

Returns a **String** (p. 2620) that represents the value of the given short value.

Parameters

<i>value</i>	The value whose string representation is to be returned.
--------------	--

Returns

a **String** (p. 2620) that contains the string representation of the short value given.

6.525.3.13 static String decaf::lang::String::valueOf (int *value*) [static]

Returns a **String** (p. 2620) that represents the value of the given integer value.

Parameters

<i>value</i>	The value whose string representation is to be returned.
--------------	--

Returns

a **String** (p. 2620) that contains the string representation of the integer value given.

6.525.3.14 static String decaf::lang::String::valueOf (long long *value*) [static]

Returns a **String** (p. 2620) that represents the value of the given 64bit long value.

Parameters

<i>value</i>	The value whose string representation is to be returned.
--------------	--

Returns

a **String** (p. 2620) that contains the string representation of the 64 bit long value given.

The documentation for this class was generated from the following file:

- src/main/decaf/lang/**String.h**

6.526 decaf::util::StringTokenizer Class Reference

Class that allows for parsing of string based on Tokens.

```
#include <src/main/decaf/util/StringTokenizer.h>
```

Public Member Functions

- **StringTokenizer** (const std::string &str, const std::string &delim=" \t\n\r\f", bool returnDelims=false)

Constructs a string tokenizer for the specified string.

- virtual ~**StringTokenizer** ()
- virtual int **countTokens** () const

Calculates the number of times that this tokenizer's nextToken method can be called before it generates an exception.

- virtual bool **hasMoreTokens** () const

Tests if there are more tokens available from this tokenizer's string.

- virtual std::string **nextToken** ()

Returns the next token from this string tokenizer.

- virtual std::string **nextToken** (const std::string &delim)

Returns the next token in this string tokenizer's string.

- virtual unsigned int **toArray** (std::vector< std::string > &array)

Grab all remaining tokens in the String and return them in the vector that is passed in by reference.

- virtual void **reset** (const std::string &str="", const std::string &delim="", bool returnDelims=false)

Resets the Tokenizer's position in the String to the Beginning calls to countToken and nextToken now start back at the beginning.

6.526.1 Detailed Description

Class that allows for parsing of string based on Tokens.

Since

1.0

6.526.2 Constructor & Destructor Documentation

6.526.2.1 decaf::util::StringTokenizer::StringTokenizer (const std::string & str, const std::string & delim = " \t\n\r\f", bool returnDelims = false)

Constructs a string tokenizer for the specified string.

All characters in the delim argument are the delimiters for separating tokens.

If the returnDelims flag is true, then the delimiter characters are also returned as tokens. Each delimiter is returned as a string of length one. If the flag is false, the delimiter characters are skipped and only serve as separators between tokens.

Note that if delim is "", this constructor does not throw an exception. However, trying to invoke other methods on the resulting **StringTokenizer** (p.2627) may result in an Exception.

Parameters

<i>str</i>	- The string to tokenize
<i>delim</i>	- String containing the delimiters
<i>returnDelims</i>	- boolean indicating if the delimiters are returned as tokens

6.526.2.2 virtual decaf::util::StringTokenizer::~StringTokenizer () [virtual]

6.526.3 Member Function Documentation

6.526.3.1 `virtual int decaf::util::StringTokenizer::countTokens () const`
`[virtual]`

Calculates the number of times that this tokenizer's `nextToken` method can be called before it generates an exception.

The current position is not advanced.

Returns

Count of remaining tokens

6.526.3.2 `virtual bool decaf::util::StringTokenizer::hasMoreTokens () const`
`[virtual]`

Tests if there are more tokens available from this tokenizer's string.

Returns

true if there are more tokens remaining

6.526.3.3 `virtual std::string decaf::util::StringTokenizer::nextToken ()`
`[virtual]`

Returns the next token from this string tokenizer.

Returns

string value of next token

Exceptions

<i>NoSuchElement-Exception</i> (p. 1984)	if there are no more tokens in this string.
--	---

6.526.3.4 `virtual std::string decaf::util::StringTokenizer::nextToken (const std::string & delim)` `[virtual]`

Returns the next token in this string tokenizer's string.

First, the set of characters considered to be delimiters by this **StringTokenizer** (p. 2627) object is changed to be the characters in the string `delim`. Then the next token in the string after the current position is returned. The current position is advanced beyond the recognized token. The new delimiter set remains the default after this call.

Parameters

<i>delim</i>	The string containing the new set of delimiters.
--------------	--

Returns

next string in the token list

Exceptions

NoSuchElementException (p. 1984)	if there are no more tokens in this string.
--	---

6.526.3.5 virtual void decaf::util::StringTokenizer::reset (const std::string & *str* = " ",
const std::string & *delim* = " ", bool *returnDelims* = false) [virtual]

Resets the Tokenizer's position in the String to the Beginning calls to countToken and nextToken now start back at the beginning.

This allows this object to be reused, the caller need not create a new instance every time a String needs tokenizing.

* If set the string param will reset the string that this Tokenizer is working on. If set to "" no change is made. * If set the delim param will reset the string that this Tokenizer is using to tokenize the string. If set to "", no change is made * If set the return Delims will set if this Tokenizer will return delimiters as tokens. Defaults to false.

Parameters

<i>str</i>	New String to tokenize or "", defaults to ""
<i>delim</i>	New Delimiter String to use or "", defaults to ""
<i>returnDelims</i>	Should the Tokenizer return delimiters as Tokens, default false

6.526.3.6 virtual unsigned int decaf::util::StringTokenizer::toArray (std::vector<
std::string > & *array*) [virtual]

Grab all remaining tokens in the String and return them in the vector that is passed in by reference.

Parameters

<i>array</i>	- vector to place token strings in
--------------	------------------------------------

Returns

number of string placed into the vector

The documentation for this class was generated from the following file:

- src/main/decaf/util/**StringTokenizer.h**

6.527 activemq::commands::SubscriptionInfo Class Reference

```
#include <src/main/activemq/commands/SubscriptionInfo.h>
```

Inheritance diagram for activemq::commands::SubscriptionInfo:

Public Member Functions

- **SubscriptionInfo** ()
- virtual **~SubscriptionInfo** ()
- virtual unsigned char **getDataStructureType** () const
*Get the **DataStructure** (p. 1133) Type as defined in CommandTypes.h.*
- virtual **SubscriptionInfo** * **cloneDataStructure** () const
Clone this object and return a new instance that the caller now owns, this will be an exact copy of this one.
- virtual void **copyDataStructure** (const **DataStructure** *src)
Copy the contents of the passed object into this objects members, overwriting any existing data.
- virtual std::string **toString** () const
*Returns a string containing the information for this **DataStructure** (p. 1133) such as its type and value of its elements.*
- virtual bool **equals** (const **DataStructure** *value) const
*Compares the **DataStructure** (p. 1133) passed in to this one, and returns if they are equivalent.*
- virtual const std::string & **getClientId** () const
- virtual std::string & **getClientId** ()
- virtual void **setClientId** (const std::string &clientId)
- virtual const **Pointer** < **ActiveMQDestination** > & **getDestination** () const
- virtual **Pointer** < **ActiveMQDestination** > & **getDestination** ()
- virtual void **setDestination** (const **Pointer**< **ActiveMQDestination** > &destination)
- virtual const std::string & **getSelector** () const
- virtual std::string & **getSelector** ()
- virtual void **setSelector** (const std::string &selector)
- virtual const std::string & **getSubscriptionName** () const
- virtual std::string & **getSubscriptionName** ()
- virtual void **setSubscriptionName** (const std::string &subscriptionName)
- virtual const **Pointer** < **ActiveMQDestination** > & **getSubscribedDestination** () const
- virtual **Pointer** < **ActiveMQDestination** > & **getSubscribedDestination** ()
- virtual void **setSubscribedDestination** (const **Pointer**< **ActiveMQDestination** > &subscribedDestination)

Static Public Attributes

- static const unsigned char **ID_SUBSCRIPTIONINFO** = 55

Protected Attributes

- std::string **clientId**
- **Pointer< ActiveMQDestination > destination**
- std::string **selector**
- std::string **subscriptionName**
- **Pointer< ActiveMQDestination > subscribedDestination**

6.527.1 Constructor & Destructor Documentation

6.527.1.1 **activemq::commands::SubscriptionInfo::SubscriptionInfo ()**

6.527.1.2 **virtual activemq::commands::SubscriptionInfo::~SubscriptionInfo ()**
[virtual]

6.527.2 Member Function Documentation

6.527.2.1 **virtual SubscriptionInfo* activemq::commands::SubscriptionInfo::cloneDataStructure () const**
[virtual]

Clone this object and return a new instance that the caller now owns, this will be an exact copy of this one.

Returns

new copy of this object.

Implements **activemq::commands::DataStructure** (p. 1133).

6.527.2.2 **virtual void activemq::commands::SubscriptionInfo::copyDataStructure (const DataStructure * src)** [virtual]

Copy the contents of the passed object into this objects members, overwriting any existing data.

Returns

src - Source Object

Implements **activemq::commands::DataStructure** (p. 1134).

6.527.2.3 `virtual bool activemq::commands::SubscriptionInfo::equals (const
DataStructure * value) const [virtual]`

Compares the **DataStructure** (p. 1133) passed in to this one, and returns if they are equivalent.

Equivalent here means that they are of the same type, and that each element of the objects are the same.

Returns

true if **DataStructure** (p. 1133)'s are Equal.

Implements **activemq::commands::DataStructure** (p. 1135).

6.527.2.4 `virtual const std::string& activemq::commands::SubscriptionInfo::get-
ClientId () const [virtual]`

6.527.2.5 `virtual std::string& activemq::commands::SubscriptionInfo::getClientId ()
[virtual]`

6.527.2.6 `virtual unsigned char activemq::commands::SubscriptionInfo::getData-
StructureType () const [virtual]`

Get the **DataStructure** (p. 1133) Type as defined in CommandTypes.h.

Returns

The type of the data structure

Implements **activemq::commands::DataStructure** (p. 1137).

6.527.2.7 `virtual const Pointer<ActiveMQDestination>& activemq-
::commands::SubscriptionInfo::getDestination () const
[virtual]`

6.527.2.8 `virtual Pointer<ActiveMQDestination>& activemq-
::commands::SubscriptionInfo::getDestination ()
[virtual]`

6.527.2.9 `virtual const std::string& activemq::commands::SubscriptionInfo::get-
Selector () const [virtual]`

6.527.2.10 `virtual std::string& activemq::commands::SubscriptionInfo::getSelector ()
[virtual]`

6.527.2.11 `virtual const std::string& activemq::commands::-
SubscriptionInfo::getSubscriptionName () const
[virtual]`

- 6.527.2.12 `virtual std::string& activemq::commands::SubscriptionInfo::getSubscriptionName () [virtual]`
- 6.527.2.13 `virtual const Pointer<ActiveMQDestination>& activemq::commands::SubscriptionInfo::getSubscribedDestination () const [virtual]`
- 6.527.2.14 `virtual Pointer<ActiveMQDestination>& activemq::commands::SubscriptionInfo::getSubscribedDestination () [virtual]`
- 6.527.2.15 `virtual void activemq::commands::SubscriptionInfo::setClientId (const std::string & clientId) [virtual]`
- 6.527.2.16 `virtual void activemq::commands::SubscriptionInfo::setDestination (const Pointer< ActiveMQDestination > & destination) [virtual]`
- 6.527.2.17 `virtual void activemq::commands::SubscriptionInfo::setSelector (const std::string & selector) [virtual]`
- 6.527.2.18 `virtual void activemq::commands::SubscriptionInfo::setSubscriptionName (const std::string & subscriptionName) [virtual]`
- 6.527.2.19 `virtual void activemq::commands::SubscriptionInfo::setSubscribedDestination (const Pointer< ActiveMQDestination > & subscribedDestination) [virtual]`
- 6.527.2.20 `virtual std::string activemq::commands::SubscriptionInfo::toString () const [virtual]`

Returns a string containing the information for this **DataStructure** (p. 1133) such as its type and value of its elements.

Returns

formatted string useful for debugging.

Reimplemented from **activemq::commands::BaseDataStructure** (p. 531).

6.527.3 Field Documentation

- 6.527.3.1 `std::string activemq::commands::SubscriptionInfo::clientId [protected]`
- 6.527.3.2 `Pointer<ActiveMQDestination> activemq::commands::SubscriptionInfo::destination [protected]`

6.527.3.3 `const unsigned char activemq::commands::SubscriptionInfo::ID_SUBSCRIPTIONINFO = 55` `[static]`

6.527.3.4 `std::string activemq::commands::SubscriptionInfo::selector` `[protected]`

6.527.3.5 `std::string activemq::commands::SubscriptionInfo::subscriptionName` `[protected]`

6.527.3.6 `Pointer<ActiveMQDestination> activemq::commands::SubscriptionInfo::subscribedDestination` `[protected]`

The documentation for this class was generated from the following file:

- `src/main/activemq/commands/SubscriptionInfo.h`

6.528 `activemq::wireformat::openwire::marshal::generated::SubscriptionInfoMarshaller` Class Reference

Marshaling code for Open Wire Format for **SubscriptionInfoMarshaller** (p. 2635).

```
#include <src/main/activemq/wireformat/openwire/marshal/generated/SubscriptionInfoMarshaller.h>
```

Inheritance diagram for `activemq::wireformat::openwire::marshal::generated::SubscriptionInfoMarshaller`:

Public Member Functions

- **SubscriptionInfoMarshaller** ()
- virtual `~SubscriptionInfoMarshaller` ()
- virtual `commands::DataStructure * createObject` () const
Creates a new instance of the class that this class is a marshaling director for.
- virtual unsigned char `getDataStructureType` () const
Gets the DataStructureType that this class marshals/unmarshals.
- virtual void `tightUnmarshal` (`OpenWireFormat *wireFormat`, `commands::DataStructure *dataStructure`, `decaf::io::DataInputStream *dataIn`, `utils::BooleanStream *bs`)
Tight Un-marhsal to the given stream.
- virtual int `tightMarshal1` (`OpenWireFormat *wireFormat`, `commands::DataStructure *dataStructure`, `utils::BooleanStream *bs`)
Tight Marhsal to the given stream.
- virtual void `tightMarshal2` (`OpenWireFormat *wireFormat`, `commands::DataStructure *dataStructure`, `decaf::io::DataOutputStream *dataOut`, `utils::BooleanStream *bs`)

6.528 activemq::wireformat::openwire::marshal::generated::SubscriptionInfoMarshaller Class

Reference

2643

Tight Marshal to the given stream.

- virtual void **looseUnmarshal** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataInputStream** *dataIn)

Loose Un-marshal to the given stream.

- virtual void **looseMarshal** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataOutputStream** *dataOut)

Tight Marshal to the given stream.

6.528.1 Detailed Description

Marshaling code for Open Wire Format for **SubscriptionInfoMarshaller** (p. 2635).

NOTE!: This file is auto generated - do not modify! if you need to make a change, please see the Java Classes in the activemq-openwire-generator module

6.528.2 Constructor & Destructor Documentation

6.528.2.1 **activemq::wireformat::openwire::marshal::generated::SubscriptionInfoMarshaller::SubscriptionInfoMarshaller** ()
[inline]

6.528.2.2 **virtual activemq::wireformat::openwire::marshal::generated::SubscriptionInfoMarshaller::~~SubscriptionInfoMarshaller** ()
[inline, virtual]

6.528.3 Member Function Documentation

6.528.3.1 **virtual commands::DataStructure* activemq::wireformat::openwire::marshal::generated::SubscriptionInfoMarshaller::createObject** ()
const [virtual]

Creates a new instance of the class that this class is a marshaling director for.

Returns

newly allocated Command

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1120).

6.528.3.2 **virtual unsigned char activemq::wireformat::openwire::marshal::generated::SubscriptionInfoMarshaller::getDataStructureType** () **const**
[virtual]

Gets the DataStructureType that this class marshals/unmarshals.

Returns

byte Id of this classes DataStructureType

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller**
(p. 1122).

6.528.3.3 **virtual void activemq::wireformat::openwire::marshal::generated::SubscriptionInfoMarshaller::looseMarshal (OpenWireFormat * *format*, commands::DataStructure * *command*, decaf::io::DataOutputStream * *ds*)** [virtual]

Tight Marshal to the given stream.

Parameters

<i>format</i>	- The OpenWireFormat properties
<i>command</i>	- the object to Marshal
<i>ds</i>	- DataOutputStream to marshal to

Exceptions

<i>IOException</i>	if an error occurs.
--------------------	---------------------

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller**
(p. 1123).

6.528.3.4 **virtual void activemq::wireformat::openwire::marshal::generated::SubscriptionInfoMarshaller::looseUnmarshal (OpenWireFormat * *format*, commands::DataStructure * *command*, decaf::io::DataInputStream * *dis*)** [virtual]

Loose Un-marshal to the given stream.

Parameters

<i>format</i>	- The OpenWireFormat properties
<i>command</i>	- the object to Un-Marshal
<i>dis</i>	- the DataInputStream to Un-Marshal from

Exceptions

<i>IOException</i>	if an error occurs.
--------------------	---------------------

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller**
(p. 1125).

6.528 activemq::wireformat::openwire::marshal::generated::SubscriptionInfo-Marshaller Class

Reference

2645

6.528.3.5 `virtual int activemq::wireformat::openwire::marshal::generated::SubscriptionInfoMarshaller::tightMarshal1 (OpenWireFormat * format, commands::DataStructure * command, utils::BooleanStream * bs)`
[virtual]

Tight Marhsal to the given stream.

Parameters

<i>format</i>	The OpenwireFormat properties
<i>command</i>	The object to Marshal
<i>bs</i>	The boolean stream to marshal to.

Exceptions

<i>IOException</i>	if an error occurs.
--------------------	---------------------

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1127).

6.528.3.6 `virtual void activemq::wireformat::openwire::marshal::generated::SubscriptionInfoMarshaller::tightMarshal2 (OpenWireFormat * format, commands::DataStructure * command, decaf::io::DataOutputStream * ds, utils::BooleanStream * bs)` [virtual]

Tight Marhsal to the given stream.

Parameters

<i>format</i>	- The OpenwireFormat properties
<i>command</i>	- the object to Marshal
<i>ds</i>	- the DataOutputStream to Marshal to
<i>bs</i>	- boolean stream to marshal to.

Exceptions

<i>IOException</i>	if an error occurs.
--------------------	---------------------

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1129).

6.528.3.7 `virtual void activemq::wireformat::openwire::marshal::generated::SubscriptionInfoMarshaller::tightUnmarshal (OpenWireFormat * format, commands::DataStructure * command, decaf::io::DataInputStream * dis, utils::BooleanStream * bs)` [virtual]

Tight Un-marhsal to the given stream.

Parameters

<i>format</i>	- The OpenwireFormat properties
<i>command</i>	- the object to Un-Marshal
<i>dis</i>	- the DataInputStream to Un-Marshal from
<i>bs</i>	- boolean stream to unmarshal from.

Exceptions

<i>IOException</i>	if an error occurs.
--------------------	---------------------

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1131).

The documentation for this class was generated from the following file:

- src/main/activemq/wireformat/openwire/marshal/generated/**SubscriptionInfo-Marshaller.h**

6.529 decaf::util::concurrent::Synchronizable Class Reference

The interface for all synchronizable objects (that is, objects that can be locked and unlocked).

```
#include <src/main/decaf/util/concurrent/Synchronizable.h>
```

Inheritance diagram for decaf::util::concurrent::Synchronizable:

Public Member Functions

- virtual **~Synchronizable** ()
- virtual void **lock** ()=0
Locks the object.
- virtual bool **tryLock** ()=0
*Attempts to **Lock** (p. 1682) the object, if the lock is already held by another thread than this method returns false.*
- virtual void **unlock** ()=0
Unlocks the object.
- virtual void **wait** ()=0
Waits on a signal from this object, which is generated by a call to Notify.
- virtual void **wait** (long long millisecs)=0
Waits on a signal from this object, which is generated by a call to Notify.
- virtual void **wait** (long long millisecs, int nanos)=0
Waits on a signal from this object, which is generated by a call to Notify.

- virtual void **notify** ()=0
Signals a waiter on this object that it can now wake up and continue.
- virtual void **notifyAll** ()=0
Signals the waiters on this object that it can now wake up and continue.

6.529.1 Detailed Description

The interface for all synchronizable objects (that is, objects that can be locked and unlocked).

Since

1.0

6.529.2 Constructor & Destructor Documentation

6.529.2.1 virtual **decaf::util::concurrent::Synchronizable::~Synchronizable** ()
[inline, virtual]

6.529.3 Member Function Documentation

6.529.3.1 virtual void **decaf::util::concurrent::Synchronizable::lock** () [pure virtual]

Locks the object.

Exceptions

<i>RuntimeException</i>	if an error occurs while locking the object.
-------------------------	--

Implemented in **decaf::util::concurrent::CopyOnWriteArrayList< E >** (p. 1042), **decaf::util::concurrent::CopyOnWriteArrayList< ServiceListener * >** (p. 1042), **decaf::util::concurrent::ConcurrentStlMap< K, V, COMPARATOR >** (p. 913), **decaf::util::concurrent::ConcurrentStlMap< Pointer< Messageld >, Pointer< Message >, Messageld::COMPARATOR >** (p. 913), **decaf::util::concurrent::ConcurrentStlMap< Pointer< ConnectionId >, Pointer< ConnectionState >, ConnectionId::COMPARATOR >** (p. 913), **decaf::util::concurrent::ConcurrentStlMap< Pointer< ConsumerId >, Pointer< ConsumerState >, ConsumerId::COMPARATOR >** (p. 913), **decaf::util::concurrent::ConcurrentStlMap< Pointer< SessionId >, Pointer< SessionState >, SessionId::COMPARATOR >** (p. 913), **decaf::util::concurrent::ConcurrentStlMap< Pointer< LocalTransactionId >, Pointer< TransactionState >, LocalTransactionId::COMPARATOR >** (p. 913), **decaf::util::concurrent::ConcurrentStlMap< Pointer< ProducerId >, Pointer< ProducerState >, ProducerId::COMPARATOR >** (p. 913), **decaf::util::AbstractCollection< E >** (p. 115), **decaf::util::AbstractCollection< Pointer< Transport > >** (p. 115), **decaf::util::AbstractCollection< Pointer< Synchronization > >** (p. 115), **decaf::util::AbstractCollection< Resource * >** (p. 115), **decaf::util::**

AbstractCollection< **cms::MessageConsumer** * > (p. 115), **decaf::util::AbstractCollection**< **CompositeTask** * > (p. 115), **decaf::util::AbstractCollection**< **URI** > (p. 115), **decaf::util::AbstractCollection**< **Pointer**< **MessageDispatch** > > (p. 115), **decaf::util::AbstractCollection**< **Pointer**< **DestinationInfo** > > (p. 115), **decaf::util::AbstractCollection**< **PrimitiveValueNode** > (p. 115), **decaf::util::AbstractCollection**< **Pointer**< **Command** > > (p. 115), **decaf::util::AbstractCollection**< **Pointer**< **BackupTransport** > > (p. 115), **decaf::util::AbstractCollection**< **cms::MessageProducer** * > (p. 115), **decaf::util::AbstractCollection**< **cms::Destination** * > (p. 115), **decaf::util::AbstractCollection**< **cms::Session** * > (p. 115), **decaf::util::AbstractCollection**< **cms::Connection** * > (p. 115), **decaf::io::InputStream** (p. 1467), **decaf::util::StlMap**< **K**, **V**, **COMPARATOR** > (p. 2560), **decaf::util::StlMap**< **cms::Session** *, **SessionResolver** * > (p. 2560), **decaf::util::StlMap**< **std::string**, **WireFormatFactory** * > (p. 2560), **decaf::util::StlMap**< **decaf::lang::Runnable** *, **decaf::util::TimerTask** * > (p. 2560), **decaf::util::StlMap**< **std::string**, **PrimitiveValueNode** > (p. 2560), **decaf::util::StlMap**< **std::string**, **cms::Queue** * > (p. 2560), **decaf::util::StlMap**< **std::string**, **CachedConsumer** * > (p. 2560), **decaf::util::StlMap**< **Pointer**< **commands::ConsumerId** >, **ActiveMQConsumer** *, **commands::ConsumerId::COMPARATOR** > (p. 2560), **decaf::util::StlMap**< **std::string**, **TransportFactory** * > (p. 2560), **decaf::util::StlMap**< **Pointer**< **ConsumerId** >, **Pointer**< **ConsumerInfo** >, **ConsumerId::COMPARATOR** > (p. 2560), **decaf::util::StlMap**< **int**, **Pointer**< **Command** > > (p. 2560), **decaf::util::StlMap**< **std::string**, **CachedProducer** * > (p. 2560), **decaf::util::StlMap**< **std::string**, **cms::Topic** * > (p. 2560), **decaf::util::StlQueue**< **T** > (p. 2569), **decaf::io::OutputStream** (p. 2069), **activemq::core::SimplePriorityMessageDispatchChannel** (p. 2448), **activemq::core::FifoMessageDispatchChannel** (p. 1326), **decaf::util::concurrent::Mutex** (p. 1962), and **decaf::internal::util::concurrent::SynchronizableImpl** (p. 2651).

6.529.3.2 **virtual void decaf::util::concurrent::Synchronizable::notify ()** [pure virtual]

Signals a waiter on this object that it can now wake up and continue.

Must have this object locked before calling.

Exceptions

<i>IllegalMonitorStateException</i>	- if the current thread is not the owner of the the Synchronizable (p. 2639) Object.
<i>RuntimeException</i>	if an error occurs while notifying one of the waiting threads.

Implemented in **decaf::util::concurrent::CopyOnWriteArrayList**< **E** > (p. 1043), **decaf::util::concurrent::CopyOnWriteArrayList**< **ServiceListener** * > (p. 1043), **decaf::util::concurrent::ConcurrentStlMap**< **K**, **V**, **COMPARATOR** > (p. 914), **decaf::util::concurrent::ConcurrentStlMap**< **Pointer**< **MessageId** >, **Pointer**< **Message** >, **MessageId::COMPARATOR** > (p. 914), **decaf::util::concurrent::ConcurrentStlMap**< **Pointer**< **ConnectionId** >, **Pointer**< **ConnectionState** >, **ConnectionId::COMPARATOR** > (p. 914), **decaf::util::concurrent::ConcurrentStlMap**< **Pointer**< **ConsumerId** >, **Pointer**< **ConsumerState** >, **ConsumerId::COMPARATOR** > (p. 914), **decaf::util::concurrent::ConcurrentStlMap**< **Pointer**<

SessionId >, Pointer< SessionState >, SessionId::COMPARATOR > (p. 914), decaf::util::concurrent::ConcurrentStlMap< Pointer< LocalTransactionId >, Pointer< TransactionState >, LocalTransactionId::COMPARATOR > (p. 914), decaf::util::concurrent::ConcurrentStlMap< Pointer< ProducerId >, Pointer< ProducerState >, ProducerId::COMPARATOR > (p. 914), decaf::util::AbstractCollection< E > (p. 115), decaf::util::AbstractCollection< Pointer< Transport > > (p. 115), decaf::util::AbstractCollection< Pointer< Synchronization > > (p. 115), decaf::util::AbstractCollection< Resource * > (p. 115), decaf::util::AbstractCollection< cms::MessageConsumer * > (p. 115), decaf::util::AbstractCollection< CompositeTask * > (p. 115), decaf::util::AbstractCollection< URI > (p. 115), decaf::util::AbstractCollection< Pointer< MessageDispatch > > (p. 115), decaf::util::AbstractCollection< Pointer< DestinationInfo > > (p. 115), decaf::util::AbstractCollection< PrimitiveValueNode > (p. 115), decaf::util::AbstractCollection< Pointer< Command > > (p. 115), decaf::util::AbstractCollection< Pointer< BackupTransport > > (p. 115), decaf::util::AbstractCollection< cms::MessageProducer * > (p. 115), decaf::util::AbstractCollection< cms::Destination * > (p. 115), decaf::util::AbstractCollection< cms::Session * > (p. 115), decaf::util::AbstractCollection< cms::Connection * > (p. 115), decaf::io::InputStream (p. 1468), decaf::util::StlMap< K, V, COMPARATOR > (p. 2560), decaf::util::StlMap< cms::Session *, SessionResolver * > (p. 2560), decaf::util::StlMap< std::string, WireFormatFactory * > (p. 2560), decaf::util::StlMap< decaf::lang::Runnable *, decaf::util::TimerTask * > (p. 2560), decaf::util::StlMap< std::string, PrimitiveValueNode > (p. 2560), decaf::util::StlMap< std::string, cms::Queue * > (p. 2560), decaf::util::StlMap< std::string, CachedConsumer * > (p. 2560), decaf::util::StlMap< Pointer< commands::ConsumerId >, ActiveMQConsumer *, commands::ConsumerId::COMPARATOR > (p. 2560), decaf::util::StlMap< std::string, TransportFactory * > (p. 2560), decaf::util::StlMap< Pointer< ConsumerId >, Pointer< ConsumerInfo >, ConsumerId::COMPARATOR > (p. 2560), decaf::util::StlMap< int, Pointer< Command > > (p. 2560), decaf::util::StlMap< std::string, CachedProducer * > (p. 2560), decaf::util::StlMap< std::string, cms::Topic * > (p. 2560), decaf::util::StlQueue< T > (p. 2570), decaf::io::OutputStream (p. 2069), activemq::core::SimplePriorityMessageDispatchChannel (p. 2448), activemq::core::FifoMessageDispatchChannel (p. 1326), decaf::util::concurrent::Mutex (p. 1962), and decaf::internal::util::concurrent::SynchronizableImpl (p. 2651).

6.529.3.3 virtual void decaf::util::concurrent::Synchronizable::notifyAll() [pure virtual]

Signals the waiters on this object that it can now wake up and continue.

Must have this object locked before calling.

Exceptions

<i>IllegalMonitorStateException</i>	- if the current thread is not the owner of the the Synchronizable (p. 2639) Object.
<i>RuntimeException</i>	if an error occurs while notifying the waiting threads.

Implemented in **decaf::util::concurrent::CopyOnWriteArrayList< E >** (p. 1043),

decaf::util::concurrent::CopyOnWriteArrayList< ServiceListener * > (p. 1043),
 decaf::util::concurrent::ConcurrentStlMap< K, V, COMPARATOR > (p. 914),
 decaf::util::concurrent::ConcurrentStlMap< Pointer< MessageId >, Pointer<
 Message >, MessageId::COMPARATOR > (p. 914), decaf::util::concurrent::-
 ConcurrentStlMap< Pointer< ConnectionId >, Pointer< ConnectionState >,
 ConnectionId::COMPARATOR > (p. 914), decaf::util::concurrent::Concurrent-
 StlMap< Pointer< ConsumerId >, Pointer< ConsumerState >, ConsumerId::C-
 OMPARATOR > (p. 914), decaf::util::concurrent::ConcurrentStlMap< Pointer<
 SessionId >, Pointer< SessionState >, SessionId::COMPARATOR > (p. 914),
 decaf::util::concurrent::ConcurrentStlMap< Pointer< LocalTransactionId >,
 Pointer< TransactionState >, LocalTransactionId::COMPARATOR > (p. 914),
 decaf::util::concurrent::ConcurrentStlMap< Pointer< ProducerId >, Pointer<
 ProducerState >, ProducerId::COMPARATOR > (p. 914), decaf::util::Abstract-
 Collection< E > (p. 116), decaf::util::AbstractCollection< Pointer< Transport
 > > (p. 116), decaf::util::AbstractCollection< Pointer< Synchronization > >
 (p. 116), decaf::util::AbstractCollection< Resource * > (p. 116), decaf::util::-
 AbstractCollection< cms::MessageConsumer * > (p. 116), decaf::util::Abstract-
 Collection< CompositeTask * > (p. 116), decaf::util::AbstractCollection< URI >
 (p. 116), decaf::util::AbstractCollection< Pointer< MessageDispatch > > (p. 116),
 decaf::util::AbstractCollection< Pointer< DestinationInfo > > (p. 116), decaf-
 ::util::AbstractCollection< PrimitiveValueNode > (p. 116), decaf::util::Abstract-
 Collection< Pointer< Command > > (p. 116), decaf::util::AbstractCollection<
 Pointer< BackupTransport > > (p. 116), decaf::util::AbstractCollection< cms::-
 MessageProducer * > (p. 116), decaf::util::AbstractCollection< cms::Destination
 * > (p. 116), decaf::util::AbstractCollection< cms::Session * > (p. 116), decaf-
 ::util::AbstractCollection< cms::Connection * > (p. 116), decaf::io::InputStream
 (p. 1469), decaf::util::StlMap< K, V, COMPARATOR > (p. 2561), decaf::util::-
 StlMap< cms::Session *, SessionResolver * > (p. 2561), decaf::util::StlMap<
 std::string, WireFormatFactory * > (p. 2561), decaf::util::StlMap< decaf::lang::-
 Runnable *, decaf::util::TimerTask * > (p. 2561), decaf::util::StlMap< std::string,
 PrimitiveValueNode > (p. 2561), decaf::util::StlMap< std::string, cms::Queue *
 > (p. 2561), decaf::util::StlMap< std::string, CachedConsumer * > (p. 2561),
 decaf::util::StlMap< Pointer< commands::ConsumerId >, ActiveMQConsumer
 *, commands::ConsumerId::COMPARATOR > (p. 2561), decaf::util::StlMap<
 std::string, TransportFactory * > (p. 2561), decaf::util::StlMap< Pointer<
 ConsumerId >, Pointer< ConsumerInfo >, ConsumerId::COMPARATOR >
 (p. 2561), decaf::util::StlMap< int, Pointer< Command > > (p. 2561), decaf-
 ::util::StlMap< std::string, CachedProducer * > (p. 2561), decaf::util::StlMap<
 std::string, cms::Topic * > (p. 2561), decaf::util::StlQueue< T > (p. 2570), decaf-
 ::io::OutputStream (p. 2070), activemq::core::SimplePriorityMessageDispatch-
 Channel (p. 2448), activemq::core::FifoMessageDispatchChannel (p. 1327),
 decaf::util::concurrent::Mutex (p. 1962), and decaf::internal::util::concurrent::-
 SynchronizableImpl (p. 2652).

6.529.3.4 virtual bool decaf::util::concurrent::Synchronizable::tryLock() [pure
 virtual]

Attempts to **Lock** (p. 1682) the object, if the lock is already held by another thread than
 this method returns false.

Returns

true if the lock was acquired, false if it is already held by another thread.

Exceptions

<i>RuntimeException</i>	if an error occurs while locking the object.
-------------------------	--

Implemented in **decaf::util::concurrent::CopyOnWriteArrayList< E >** (p. 1047), **decaf::util::concurrent::CopyOnWriteArrayList< ServiceListener * >** (p. 1047), **decaf::util::concurrent::ConcurrentStlMap< K, V, COMPARATOR >** (p. 919), **decaf::util::concurrent::ConcurrentStlMap< Pointer< Messageld >, Pointer< Message >, Messageld::COMPARATOR >** (p. 919), **decaf::util::concurrent::ConcurrentStlMap< Pointer< ConnectionId >, Pointer< ConnectionState >, ConnectionId::COMPARATOR >** (p. 919), **decaf::util::concurrent::ConcurrentStlMap< Pointer< ConsumerId >, Pointer< ConsumerState >, ConsumerId::COMPARATOR >** (p. 919), **decaf::util::concurrent::ConcurrentStlMap< Pointer< SessionId >, Pointer< SessionState >, SessionId::COMPARATOR >** (p. 919), **decaf::util::concurrent::ConcurrentStlMap< Pointer< LocalTransactionId >, Pointer< TransactionState >, LocalTransactionId::COMPARATOR >** (p. 919), **decaf::util::concurrent::ConcurrentStlMap< Pointer< ProducerId >, Pointer< ProducerState >, ProducerId::COMPARATOR >** (p. 919), **decaf::util::AbstractCollection< E >** (p. 119), **decaf::util::AbstractCollection< Pointer< Transport > >** (p. 119), **decaf::util::AbstractCollection< Pointer< Synchronization > >** (p. 119), **decaf::util::AbstractCollection< Resource * >** (p. 119), **decaf::util::AbstractCollection< cms::MessageConsumer * >** (p. 119), **decaf::util::AbstractCollection< CompositeTask * >** (p. 119), **decaf::util::AbstractCollection< URI >** (p. 119), **decaf::util::AbstractCollection< Pointer< MessageDispatch > >** (p. 119), **decaf::util::AbstractCollection< Pointer< DestinationInfo > >** (p. 119), **decaf::util::AbstractCollection< PrimitiveValueNode >** (p. 119), **decaf::util::AbstractCollection< Pointer< Command > >** (p. 119), **decaf::util::AbstractCollection< Pointer< BackupTransport > >** (p. 119), **decaf::util::AbstractCollection< cms::MessageProducer * >** (p. 119), **decaf::util::AbstractCollection< cms::Destination * >** (p. 119), **decaf::util::AbstractCollection< cms::Session * >** (p. 119), **decaf::util::AbstractCollection< cms::Connection * >** (p. 119), **decaf::io::InputStream** (p. 1473), **decaf::util::StlMap< K, V, COMPARATOR >** (p. 2563), **decaf::util::StlMap< cms::Session *, SessionResolver * >** (p. 2563), **decaf::util::StlMap< std::string, WireFormatFactory * >** (p. 2563), **decaf::util::StlMap< decaf::lang::Runnable *, decaf::util::TimerTask * >** (p. 2563), **decaf::util::StlMap< std::string, PrimitiveValueNode >** (p. 2563), **decaf::util::StlMap< std::string, cms::Queue * >** (p. 2563), **decaf::util::StlMap< std::string, CachedConsumer * >** (p. 2563), **decaf::util::StlMap< Pointer< commands::ConsumerId >, ActiveMQConsumer *, commands::ConsumerId::COMPARATOR >** (p. 2563), **decaf::util::StlMap< std::string, TransportFactory * >** (p. 2563), **decaf::util::StlMap< Pointer< ConsumerId >, Pointer< ConsumerInfo >, ConsumerId::COMPARATOR >** (p. 2563), **decaf::util::StlMap< int, Pointer< Command > >** (p. 2563), **decaf::util::StlMap< std::string, CachedProducer * >** (p. 2563), **decaf::util::StlMap< std::string, cms::Topic * >** (p. 2563), **decaf::util::StlQueue< T >** (p. 2571), **decaf::io::OutputStream** (p. 2070), **activemq::core::SimplePriorityMessageDispatchChannel** (p. 2450), **activemq::core::FifoMessageDispatchChannel** (p. 1328),

decaf::util::concurrent::Mutex (p. 1963), and **decaf::internal::util::concurrent::SynchronizableImpl** (p. 2652).

6.529.3.5 **virtual void decaf::util::concurrent::Synchronizable::unlock ()** [pure virtual]

Unlocks the object.

Exceptions

<i>RuntimeException</i>	if an error occurs while unlocking the object.
-------------------------	--

Implemented in **decaf::util::concurrent::CopyOnWriteArrayList< E >** (p. 1048), **decaf::util::concurrent::CopyOnWriteArrayList< ServiceListener * >** (p. 1048), **decaf::util::concurrent::ConcurrentStlMap< K, V, COMPARATOR >** (p. 919), **decaf::util::concurrent::ConcurrentStlMap< Pointer< Messageld >, Pointer< Message >, Messageld::COMPARATOR >** (p. 919), **decaf::util::concurrent::ConcurrentStlMap< Pointer< ConnectionId >, Pointer< ConnectionState >, ConnectionId::COMPARATOR >** (p. 919), **decaf::util::concurrent::ConcurrentStlMap< Pointer< ConsumerId >, Pointer< ConsumerState >, ConsumerId::COMPARATOR >** (p. 919), **decaf::util::concurrent::ConcurrentStlMap< Pointer< SessionId >, Pointer< SessionState >, SessionId::COMPARATOR >** (p. 919), **decaf::util::concurrent::ConcurrentStlMap< Pointer< LocalTransactionId >, Pointer< TransactionState >, LocalTransactionId::COMPARATOR >** (p. 919), **decaf::util::concurrent::ConcurrentStlMap< Pointer< ProducerId >, Pointer< ProducerState >, ProducerId::COMPARATOR >** (p. 919), **decaf::util::AbstractCollection< E >** (p. 120), **decaf::util::AbstractCollection< Pointer< Transport > >** (p. 120), **decaf::util::AbstractCollection< Pointer< Synchronization > >** (p. 120), **decaf::util::AbstractCollection< Resource * >** (p. 120), **decaf::util::AbstractCollection< cms::MessageConsumer * >** (p. 120), **decaf::util::AbstractCollection< CompositeTask * >** (p. 120), **decaf::util::AbstractCollection< URI >** (p. 120), **decaf::util::AbstractCollection< Pointer< MessageDispatch > >** (p. 120), **decaf::util::AbstractCollection< Pointer< DestinationInfo > >** (p. 120), **decaf::util::AbstractCollection< PrimitiveValueNode >** (p. 120), **decaf::util::AbstractCollection< Pointer< Command > >** (p. 120), **decaf::util::AbstractCollection< Pointer< BackupTransport > >** (p. 120), **decaf::util::AbstractCollection< cms::MessageProducer * >** (p. 120), **decaf::util::AbstractCollection< cms::Destination * >** (p. 120), **decaf::util::AbstractCollection< cms::Session * >** (p. 120), **decaf::util::AbstractCollection< cms::Connection * >** (p. 120), **decaf::io::InputStream** (p. 1473), **decaf::util::StlMap< K, V, COMPARATOR >** (p. 2563), **decaf::util::StlMap< cms::Session *, SessionResolver * >** (p. 2563), **decaf::util::StlMap< std::string, WireFormatFactory * >** (p. 2563), **decaf::util::StlMap< decaf::lang::Runnable *, decaf::util::TimerTask * >** (p. 2563), **decaf::util::StlMap< std::string, PrimitiveValueNode >** (p. 2563), **decaf::util::StlMap< std::string, cms::Queue * >** (p. 2563), **decaf::util::StlMap< std::string, CachedConsumer * >** (p. 2563), **decaf::util::StlMap< Pointer< commands::ConsumerId >, ActiveMQConsumer *, commands::ConsumerId::COMPARATOR >** (p. 2563), **decaf::util::StlMap< std::string, TransportFactory * >** (p. 2563), **decaf::util::StlMap< Pointer< ConsumerId >, Pointer< ConsumerInfo >, ConsumerId::COMPARATOR >**

(p. 2563), `decaf::util::StlMap< int, Pointer< Command > >` (p. 2563), `decaf::util::StlMap< std::string, CachedProducer * >` (p. 2563), `decaf::util::StlMap< std::string, cms::Topic * >` (p. 2563), `decaf::util::StlQueue< T >` (p. 2572), `decaf::io::OutputStream` (p. 2071), `activemq::core::SimplePriorityMessageDispatchChannel` (p. 2450), `activemq::core::FifoMessageDispatchChannel` (p. 1328), `decaf::util::concurrent::Mutex` (p. 1963), and `decaf::internal::util::concurrent::SynchronizableImpl` (p. 2652).

6.529.3.6 `virtual void decaf::util::concurrent::Synchronizable::wait ()` [pure virtual]

Waits on a signal from this object, which is generated by a call to Notify.

Must have this object locked before calling.

Exceptions

<i>RuntimeException</i>	if an error occurs while waiting on the object.
<i>InterruptedException</i>	if the wait is interrupted before it completes.
<i>IllegalMonitorStateException</i>	- if the current thread is not the owner of the the Synchronizable (p. 2639) Object.

Implemented in `decaf::util::concurrent::CopyOnWriteArrayList< E >` (p. 1048), `decaf::util::concurrent::CopyOnWriteArrayList< ServiceListener * >` (p. 1048), `decaf::util::concurrent::ConcurrentStlMap< K, V, COMPARATOR >` (p. 920), `decaf::util::concurrent::ConcurrentStlMap< Pointer< MessageId >, Pointer< Message >, MessageId::COMPARATOR >` (p. 920), `decaf::util::concurrent::ConcurrentStlMap< Pointer< ConnectionId >, Pointer< ConnectionState >, ConnectionId::COMPARATOR >` (p. 920), `decaf::util::concurrent::ConcurrentStlMap< Pointer< ConsumerId >, Pointer< ConsumerState >, ConsumerId::COMPARATOR >` (p. 920), `decaf::util::concurrent::ConcurrentStlMap< Pointer< SessionId >, Pointer< SessionState >, SessionId::COMPARATOR >` (p. 920), `decaf::util::concurrent::ConcurrentStlMap< Pointer< LocalTransactionId >, Pointer< TransactionState >, LocalTransactionId::COMPARATOR >` (p. 920), `decaf::util::concurrent::ConcurrentStlMap< Pointer< ProducerId >, Pointer< ProducerState >, ProducerId::COMPARATOR >` (p. 920), `decaf::util::AbstractCollection< E >` (p. 120), `decaf::util::AbstractCollection< Pointer< Transport > >` (p. 120), `decaf::util::AbstractCollection< Pointer< Synchronization > >` (p. 120), `decaf::util::AbstractCollection< Resource * >` (p. 120), `decaf::util::AbstractCollection< cms::MessageConsumer * >` (p. 120), `decaf::util::AbstractCollection< CompositeTask * >` (p. 120), `decaf::util::AbstractCollection< URI >` (p. 120), `decaf::util::AbstractCollection< Pointer< MessageDispatch > >` (p. 120), `decaf::util::AbstractCollection< Pointer< DestinationInfo > >` (p. 120), `decaf::util::AbstractCollection< PrimitiveValueNode >` (p. 120), `decaf::util::AbstractCollection< Pointer< Command > >` (p. 120), `decaf::util::AbstractCollection< Pointer< BackupTransport > >` (p. 120), `decaf::util::AbstractCollection< cms::MessageProducer * >` (p. 120), `decaf::util::AbstractCollection< cms::Destination * >` (p. 120), `decaf::util::AbstractCollection< cms::Session * >` (p. 120), `decaf::util::AbstractCollection< cms::Connection * >` (p. 120), `decaf::io::InputStream` (p. 1473), `decaf::util::StlMap< K, V, COMPARATOR >` (p. 2564), `decaf::util::`

StlMap< **cms::Session** *, **SessionResolver** * > (p. 2564), **decaf::util::StlMap**< **std::string**, **WireFormatFactory** * > (p. 2564), **decaf::util::StlMap**< **decaf::lang::Runnable** *, **decaf::util::TimerTask** * > (p. 2564), **decaf::util::StlMap**< **std::string**, **PrimitiveValueNode** > (p. 2564), **decaf::util::StlMap**< **std::string**, **cms::Queue** * > (p. 2564), **decaf::util::StlMap**< **std::string**, **CachedConsumer** * > (p. 2564), **decaf::util::StlMap**< **Pointer**< **commands::ConsumerId** >, **ActiveMQConsumer** *, **commands::ConsumerId::COMPARATOR** > (p. 2564), **decaf::util::StlMap**< **std::string**, **TransportFactory** * > (p. 2564), **decaf::util::StlMap**< **Pointer**< **ConsumerId** >, **Pointer**< **ConsumerInfo** >, **ConsumerId::COMPARATOR** > (p. 2564), **decaf::util::StlMap**< **int**, **Pointer**< **Command** > > (p. 2564), **decaf::util::StlMap**< **std::string**, **CachedProducer** * > (p. 2564), **decaf::util::StlMap**< **std::string**, **cms::Topic** * > (p. 2564), **decaf::util::StlQueue**< **T** > (p. 2572), **decaf::io::OutputStream** (p. 2071), **activemq::core::SimplePriorityMessageDispatchChannel** (p. 2450), **activemq::core::FifoMessageDispatchChannel** (p. 1329), **decaf::util::concurrent::Mutex** (p. 1964), and **decaf::internal::util::concurrent::SynchronizableImpl** (p. 2652).

6.529.3.7 **virtual void decaf::util::concurrent::Synchronizable::wait (long long millisecs)** [pure virtual]

Waits on a signal from this object, which is generated by a call to Notify.

Must have this object locked before calling. This wait will timeout after the specified time interval.

Parameters

<i>millisecs</i>	the time in milliseconds to wait, or WAIT_INFINITE
------------------	--

Exceptions

<i>RuntimeException</i>	if an error occurs while waiting on the object.
<i>InterruptedException</i>	if the wait is interrupted before it completes.
<i>IllegalMonitorStateException</i>	- if the current thread is not the owner of the the Synchronizable (p. 2639) Object.

Implemented in **decaf::util::concurrent::CopyOnWriteArrayList**< **E** > (p. 1048), **decaf::util::concurrent::CopyOnWriteArrayList**< **ServiceListener** * > (p. 1048), **decaf::util::concurrent::ConcurrentStlMap**< **K**, **V**, **COMPARATOR** > (p. 920), **decaf::util::concurrent::ConcurrentStlMap**< **Pointer**< **MessageId** >, **Pointer**< **Message** >, **MessageId::COMPARATOR** > (p. 920), **decaf::util::concurrent::ConcurrentStlMap**< **Pointer**< **ConnectionId** >, **Pointer**< **ConnectionState** >, **ConnectionId::COMPARATOR** > (p. 920), **decaf::util::concurrent::ConcurrentStlMap**< **Pointer**< **ConsumerId** >, **Pointer**< **ConsumerState** >, **ConsumerId::COMPARATOR** > (p. 920), **decaf::util::concurrent::ConcurrentStlMap**< **Pointer**< **SessionId** >, **Pointer**< **SessionState** >, **SessionId::COMPARATOR** > (p. 920), **decaf::util::concurrent::ConcurrentStlMap**< **Pointer**< **LocalTransactionId** >, **Pointer**< **TransactionState** >, **LocalTransactionId::COMPARATOR** > (p. 920), **decaf::util::concurrent::ConcurrentStlMap**< **Pointer**< **ProducerId** >, **Pointer**<

ProducerState >, **ProducerId::COMPARATOR** > (p. 920), **decaf::util::AbstractCollection**< **E** > (p. 120), **decaf::util::AbstractCollection**< **Pointer**< **Transport** > > (p. 120), **decaf::util::AbstractCollection**< **Pointer**< **Synchronization** > > (p. 120), **decaf::util::AbstractCollection**< **Resource** * > (p. 120), **decaf::util::AbstractCollection**< **cms::MessageConsumer** * > (p. 120), **decaf::util::AbstractCollection**< **CompositeTask** * > (p. 120), **decaf::util::AbstractCollection**< **URI** > (p. 120), **decaf::util::AbstractCollection**< **Pointer**< **MessageDispatch** > > (p. 120), **decaf::util::AbstractCollection**< **Pointer**< **DestinationInfo** > > (p. 120), **decaf::util::AbstractCollection**< **PrimitiveValueNode** > (p. 120), **decaf::util::AbstractCollection**< **Pointer**< **Command** > > (p. 120), **decaf::util::AbstractCollection**< **Pointer**< **BackupTransport** > > (p. 120), **decaf::util::AbstractCollection**< **cms::MessageProducer** * > (p. 120), **decaf::util::AbstractCollection**< **cms::Destination** * > (p. 120), **decaf::util::AbstractCollection**< **cms::Session** * > (p. 120), **decaf::util::AbstractCollection**< **cms::Connection** * > (p. 120), **decaf::io::InputStream** (p. 1474), **decaf::util::StlMap**< **K**, **V**, **COMPARATOR** > (p. 2564), **decaf::util::StlMap**< **cms::Session** *, **SessionResolver** * > (p. 2564), **decaf::util::StlMap**< **std::string**, **WireFormatFactory** * > (p. 2564), **decaf::util::StlMap**< **decaf::lang::Runnable** *, **decaf::util::TimerTask** * > (p. 2564), **decaf::util::StlMap**< **std::string**, **PrimitiveValueNode** > (p. 2564), **decaf::util::StlMap**< **std::string**, **cms::Queue** * > (p. 2564), **decaf::util::StlMap**< **std::string**, **CachedConsumer** * > (p. 2564), **decaf::util::StlMap**< **Pointer**< **commands::ConsumerId** >, **ActiveMQConsumer** *, **commands::ConsumerId::COMPARATOR** > (p. 2564), **decaf::util::StlMap**< **std::string**, **TransportFactory** * > (p. 2564), **decaf::util::StlMap**< **Pointer**< **ConsumerId** >, **Pointer**< **ConsumerInfo** >, **ConsumerId::COMPARATOR** > (p. 2564), **decaf::util::StlMap**< **int**, **Pointer**< **Command** > > (p. 2564), **decaf::util::StlMap**< **std::string**, **CachedProducer** * > (p. 2564), **decaf::util::StlMap**< **std::string**, **cms::Topic** * > (p. 2564), **decaf::util::StlQueue**< **T** > (p. 2572), **decaf::io::OutputStream** (p. 2071), **activemq::core::SimplePriorityMessageDispatchChannel** (p. 2451), **activemq::core::FifoMessageDispatchChannel** (p. 1329), **decaf::util::concurrent::Mutex** (p. 1964), and **decaf::internal::util::concurrent::SynchronizableImpl** (p. 2653).

6.529.3.8 `virtual void decaf::util::concurrent::Synchronizable::wait (long long millisecs, int nanos) [pure virtual]`

Waits on a signal from this object, which is generated by a call to `Notify`.

Must have this object locked before calling. This wait will timeout after the specified time interval. This method is similar to the one argument wait function except that it add a finer grained control over the amount of time that it waits by adding in the additional nanosecond argument.

NOTE: The ability to wait accurately at a nanosecond scale depends on the platform and OS that the Decaf API is running on, some systems do not provide an accurate enough clock to provide this level of granularity.

Parameters

<i>millisecs</i>	the time in milliseconds to wait, or <code>WAIT_INFINITE</code>
<i>nanos</i>	additional time in nanoseconds with a range of 0-999999

Exceptions

<i>IllegalArgument-Exception</i>	if an error occurs or the nanos argument is not in the range of [0-999999]
<i>RuntimeException</i>	if an error occurs while waiting on the object.
<i>InterruptedException</i>	if the wait is interrupted before it completes.
<i>IllegalMonitorState-Exception</i>	- if the current thread is not the owner of the the Synchronizable (p. 2639) Object.

Implemented in **decaf::util::concurrent::CopyOnWriteArrayList< E >** (p. 1049), **decaf::util::concurrent::CopyOnWriteArrayList< ServiceListener * >** (p. 1049), **decaf::util::concurrent::ConcurrentStlMap< K, V, COMPARATOR >** (p. 920), **decaf::util::concurrent::ConcurrentStlMap< Pointer< MessageId >, Pointer< Message >, MessageId::COMPARATOR >** (p. 920), **decaf::util::concurrent::ConcurrentStlMap< Pointer< ConnectionId >, Pointer< ConnectionState >, ConnectionId::COMPARATOR >** (p. 920), **decaf::util::concurrent::ConcurrentStlMap< Pointer< ConsumerId >, Pointer< ConsumerState >, ConsumerId::COMPARATOR >** (p. 920), **decaf::util::concurrent::ConcurrentStlMap< Pointer< SessionId >, Pointer< SessionState >, SessionId::COMPARATOR >** (p. 920), **decaf::util::concurrent::ConcurrentStlMap< Pointer< LocalTransactionId >, Pointer< TransactionState >, LocalTransactionId::COMPARATOR >** (p. 920), **decaf::util::concurrent::ConcurrentStlMap< Pointer< ProducerId >, Pointer< ProducerState >, ProducerId::COMPARATOR >** (p. 920), **decaf::util::AbstractCollection< E >** (p. 121), **decaf::util::AbstractCollection< Pointer< Transport > >** (p. 121), **decaf::util::AbstractCollection< Pointer< Synchronization > >** (p. 121), **decaf::util::AbstractCollection< Resource * >** (p. 121), **decaf::util::AbstractCollection< cms::MessageConsumer * >** (p. 121), **decaf::util::AbstractCollection< CompositeTask * >** (p. 121), **decaf::util::AbstractCollection< URI >** (p. 121), **decaf::util::AbstractCollection< Pointer< MessageDispatch > >** (p. 121), **decaf::util::AbstractCollection< Pointer< DestinationInfo > >** (p. 121), **decaf::util::AbstractCollection< PrimitiveValueNode >** (p. 121), **decaf::util::AbstractCollection< Pointer< Command > >** (p. 121), **decaf::util::AbstractCollection< Pointer< BackupTransport > >** (p. 121), **decaf::util::AbstractCollection< cms::MessageProducer * >** (p. 121), **decaf::util::AbstractCollection< cms::Destination * >** (p. 121), **decaf::util::AbstractCollection< cms::Session * >** (p. 121), **decaf::util::AbstractCollection< cms::Connection * >** (p. 121), **decaf::io::InputStream** (p. 1474), **decaf::util::StlMap< K, V, COMPARATOR >** (p. 2565), **decaf::util::StlMap< cms::Session *, SessionResolver * >** (p. 2565), **decaf::util::StlMap< std::string, WireFormatFactory * >** (p. 2565), **decaf::util::StlMap< decaf::lang::Runnable *, decaf::util::TimerTask * >** (p. 2565), **decaf::util::StlMap< std::string, PrimitiveValueNode >** (p. 2565), **decaf::util::StlMap< std::string, cms::Queue * >** (p. 2565), **decaf::util::StlMap< std::string, CachedConsumer * >** (p. 2565), **decaf::util::StlMap< Pointer< commands::ConsumerId >, ActiveMQConsumer *, commands::ConsumerId::COMPARATOR >** (p. 2565), **decaf::util::StlMap< std::string, TransportFactory * >** (p. 2565), **decaf::util::StlMap< Pointer< ConsumerId >, Pointer< ConsumerInfo >, ConsumerId::COMPARATOR >** (p. 2565), **decaf::util::StlMap< int, Pointer< Command > >** (p. 2565), **decaf::util::StlMap< std::string, CachedProducer * >** (p. 2565), **decaf::util::StlMap< std::string, cms::Topic * >** (p. 2565), **decaf::util::StlQueue< T >** (p. 2573), **decaf::io::OutputStream** (p. 2072), **activemq::core::SimplePriorityMessageDispatch-**

6.530 `decaf::internal::util::concurrent::SynchronizableImpl` Class Reference 2657

Channel (p. 2451), **activemq::core::FifoMessageDispatchChannel** (p. 1330), **decaf::util::concurrent::Mutex** (p. 1965), and **decaf::internal::util::concurrent::SynchronizableImpl** (p. 2653).

The documentation for this class was generated from the following file:

- `src/main/decaf/util/concurrent/Synchronizable.h`

6.530 `decaf::internal::util::concurrent::SynchronizableImpl` Class - Reference

A convenience class used by some Decaf classes to implement the Synchronizable interface when there is no issues related to multiple inheritance.

```
#include <src/main/decaf/internal/util/concurrent/Synchronizable-Impl.h>
```

Inheritance diagram for `decaf::internal::util::concurrent::SynchronizableImpl`:

Public Member Functions

- **SynchronizableImpl** ()
- virtual **~SynchronizableImpl** ()
- virtual void **lock** ()
Locks the object.
- virtual bool **tryLock** ()
Attempts to Lock the object, if the lock is already held by another thread than this method returns false.
- virtual void **unlock** ()
Unlocks the object.
- virtual void **wait** ()
Waits on a signal from this object, which is generated by a call to Notify.
- virtual void **wait** (long long millisecs)
Waits on a signal from this object, which is generated by a call to Notify.
- virtual void **wait** (long long millisecs, int nanos)
Waits on a signal from this object, which is generated by a call to Notify.
- virtual void **notify** ()
Signals a waiter on this object that it can now wake up and continue.
- virtual void **notifyAll** ()
Signals the waiters on this object that it can now wake up and continue.

6.530.1 Detailed Description

A convenience class used by some Decaf classes to implement the Synchronizable interface when there is no issues related to multiple inheritance.

Since

1.0

6.530.2 Constructor & Destructor Documentation

6.530.2.1 **decaf::internal::util::concurrent::SynchronizableImpl::SynchronizableImpl ()**

6.530.2.2 **virtual decaf::internal::util::concurrent::SynchronizableImpl::~~SynchronizableImpl ()** [virtual]

6.530.3 Member Function Documentation

6.530.3.1 **virtual void decaf::internal::util::concurrent::SynchronizableImpl::lock ()** [virtual]

Locks the object.

Exceptions

<i>RuntimeException</i>	if an error occurs while locking the object.
-------------------------	--

Implements **decaf::util::concurrent::Synchronizable** (p. 2640).

6.530.3.2 **virtual void decaf::internal::util::concurrent::SynchronizableImpl::notify ()** [virtual]

Signals a waiter on this object that it can now wake up and continue.

Must have this object locked before calling.

Exceptions

<i>IllegalMonitorStateException</i>	- if the current thread is not the owner of the the Synchronizable Object.
<i>RuntimeException</i>	if an error occurs while notifying one of the waiting threads.

Implements **decaf::util::concurrent::Synchronizable** (p. 2641).

6.530 decaf::internal::util::concurrent::SynchronizableImpl Class Reference 2659

6.530.3.3 virtual void **decaf::internal::util::concurrent::SynchronizableImpl::notify-All**() [virtual]

Signals the waiters on this object that it can now wake up and continue.

Must have this object locked before calling.

Exceptions

<i>IllegalMonitorStateException</i>	- if the current thread is not the owner of the the Synchronizable Object.
<i>RuntimeException</i>	if an error occurs while notifying the waiting threads.

Implements **decaf::util::concurrent::Synchronizable** (p. 2642).

6.530.3.4 virtual bool **decaf::internal::util::concurrent::SynchronizableImpl::tryLock**() [virtual]

Attempts to Lock the object, if the lock is already held by another thread than this method returns false.

Returns

true if the lock was acquired, false if it is already held by another thread.

Exceptions

<i>RuntimeException</i>	if an error occurs while locking the object.
-------------------------	--

Implements **decaf::util::concurrent::Synchronizable** (p. 2643).

6.530.3.5 virtual void **decaf::internal::util::concurrent::SynchronizableImpl::unlock**() [virtual]

Unlocks the object.

Exceptions

<i>RuntimeException</i>	if an error occurs while unlocking the object.
-------------------------	--

Implements **decaf::util::concurrent::Synchronizable** (p. 2645).

6.530.3.6 virtual void **decaf::internal::util::concurrent::SynchronizableImpl::wait**() [virtual]

Waits on a signal from this object, which is generated by a call to Notify.

Must have this object locked before calling.

Exceptions

<i>RuntimeException</i>	if an error occurs while waiting on the object.
<i>InterruptedException</i>	if the wait is interrupted before it completes.
<i>IllegalMonitorStateException</i>	- if the current thread is not the owner of the the Synchronizable Object.

Implements **decaf::util::concurrent::Synchronizable** (p. 2646).

6.530.3.7 virtual void **decaf::internal::util::concurrent::SynchronizableImpl::wait** (
long long *millisecs*) [virtual]

Waits on a signal from this object, which is generated by a call to Notify.

Must have this object locked before calling. This wait will timeout after the specified time interval.

Parameters

<i>millisecs</i>	the time in milliseconds to wait, or WAIT_INFINITE
------------------	--

Exceptions

<i>RuntimeException</i>	if an error occurs while waiting on the object.
<i>InterruptedException</i>	if the wait is interrupted before it completes.
<i>IllegalMonitorStateException</i>	- if the current thread is not the owner of the the Synchronizable Object.

Implements **decaf::util::concurrent::Synchronizable** (p. 2647).

6.530.3.8 virtual void **decaf::internal::util::concurrent::SynchronizableImpl::wait** (
long long *millisecs*, int *nanos*) [virtual]

Waits on a signal from this object, which is generated by a call to Notify.

Must have this object locked before calling. This wait will timeout after the specified time interval. This method is similar to the one argument wait function except that it add a finer grained control over the amount of time that it waits by adding in the additional nanosecond argument.

NOTE: The ability to wait accurately at a nanosecond scale depends on the platform and OS that the Decaf API is running on, some systems do not provide an accurate enough clock to provide this level of granularity.

Parameters

<i>millisecs</i>	the time in milliseconds to wait, or WAIT_INFINITE
<i>nanos</i>	additional time in nanoseconds with a range of 0-999999

Exceptions

<i>IllegalArgumentException</i>	if an error occurs or the nanos argument is not in the range of [0-999999]
<i>RuntimeException</i>	if an error occurs while waiting on the object.
<i>InterruptedException</i>	if the wait is interrupted before it completes.
<i>IllegalMonitorStateException</i>	- if the current thread is not the owner of the the Synchronizable Object.

Implements **decaf::util::concurrent::Synchronizable** (p. 2648).

The documentation for this class was generated from the following file:

- src/main/decaf/internal/util/concurrent/**SynchronizableImpl.h**

6.531 activemq::core::Synchronization Class Reference

Transacted Object **Synchronization** (p. 2654), used to sync the events of a Transaction with the items in the Transaction.

```
#include <src/main/activemq/core/Synchronization.h>
```

Public Member Functions

- virtual **~Synchronization** ()
- virtual void **beforeEnd** ()=0
- virtual void **afterCommit** ()=0
- virtual void **afterRollback** ()=0

6.531.1 Detailed Description

Transacted Object **Synchronization** (p. 2654), used to sync the events of a Transaction with the items in the Transaction.

6.531.2 Constructor & Destructor Documentation

6.531.2.1 virtual **activemq::core::Synchronization::~~Synchronization** ()
[inline, virtual]

6.531.3 Member Function Documentation

6.531.3.1 virtual void **activemq::core::Synchronization::afterCommit** () [pure virtual]

6.531.3.2 virtual void **activemq::core::Synchronization::afterRollback** () [pure virtual]

6.531.3.3 `virtual void activemq::core::Synchronization::beforeEnd ()` [pure virtual]

The documentation for this class was generated from the following file:

- `src/main/activemq/core/Synchronization.h`

6.532 `decaf::util::concurrent::SynchronousQueue< E >` Class - Template Reference

A **blocking queue** (p. 538) in which each insert operation must wait for a corresponding remove operation by another thread, and vice versa.

```
#include <src/main/decaf/util/concurrent/Synchronous-Queue.h>
```

Inheritance diagram for `decaf::util::concurrent::SynchronousQueue< E >`:

Data Structures

- class **EmptyIterator**

Public Member Functions

- **SynchronousQueue** ()
- virtual **~SynchronousQueue** ()
- virtual void **put** (const E &value)
Adds the specified element to this queue, waiting if necessary for another thread to receive it.
- virtual bool **offer** (const E &e, long long timeout, const **TimeUnit** &unit)
Inserts the specified element into this queue, waiting if necessary up to the specified wait time for another thread to receive it.
- virtual bool **offer** (const E &value)
Inserts the specified element into this queue, if another thread is waiting to receive it.
- virtual E **take** ()
Retrieves and removes the head of this queue, waiting if necessary for another thread to insert it.
- virtual bool **poll** (E &result, long long timeout, const **TimeUnit** &unit)
Retrieves and removes the head of this queue, waiting if necessary up to the specified wait time, for another thread to insert it.
- virtual bool **poll** (E &result)
Retrieves and removes the head of this queue, if another thread is currently making an element available.

- virtual bool **equals** (const **Collection**< E > &value) const
*Answers true if this **Collection** (p. 851) and the one given are the same size and if each element contained in the **Collection** (p. 851) given is equal to an element contained in this collection.*
- virtual **decaf::util::Iterator** < E > * **iterator** ()
- virtual **decaf::util::Iterator** < E > * **iterator** () const
- virtual bool **isEmpty** () const
Returns true if this collection contains no elements.
- virtual int **size** () const
Returns the number of elements in this collection.
- virtual int **remainingCapacity** () const
Returns the number of additional elements that this queue can ideally (in the absence of memory or resource constraints) accept without blocking, or `Integer::MAX_VALUE` if there is no intrinsic limit.
- virtual void **clear** ()
*Removes all of the elements from this collection (optional operation).
The collection will be empty after this method returns.
This implementation iterates over this collection, removing each element using the **Iterator.remove** (p. 1560) operation. Most implementations will probably choose to override this method for efficiency.
Note that this implementation will throw an `UnsupportedOperationException` if the iterator returned by this collection's iterator method does not implement the remove method and this collection is non-empty.*

Exceptions

UnsupportedOperationException	if the clear operation is not supported by this collection
-------------------------------	--

This implementation repeatedly invokes poll until it returns false.

- virtual bool **contains** (const E &value **DECAF_UNUSED**) const
- virtual bool **containsAll** (const **Collection**< E > &collection) const
Returns true if this collection contains all of the elements in the specified collection.

Parameters

collection	The Collection (p. 851) to compare to this one.
------------	--

Exceptions

NullPointerException	if the Collection (p. 851) contains pointers and the Collection (p. 851) does not allow for NULL elements (optional check).
----------------------	---

This implementation iterates over the specified collection, checking each element returned by the iterator in turn to see if it's contained in this collection. If all elements are so contained true is returned, otherwise false.

- virtual bool **remove** (const E &value **DECAF_UNUSED**)
- virtual bool **removeAll** (const **Collection**< E > &collection **DECAF_UNUSED**)
- virtual bool **retainAll** (const **Collection**< E > &collection **DECAF_UNUSED**)
- virtual bool **peek** (E &result **DECAF_UNUSED**) const
- virtual std::vector< E > **toArray** () const

*Answers an STL vector containing copies of all elements contained in this **Collection** (p. 851).*

- virtual int **drainTo** (**Collection**< E > &c)

Removes all available elements from this queue and adds them to the given collection.

- virtual int **drainTo** (**Collection**< E > &c, int maxElements)

Removes at most the given number of available elements from this queue and adds them to the given collection.

6.532.1 Detailed Description

```
template<typename E>class decaf::util::concurrent::SynchronousQueue< E >
```

A **blocking queue** (p. 538) in which each insert operation must wait for a corresponding remove operation by another thread, and vice versa.

A synchronous queue does not have any internal capacity, not even a capacity of one. You cannot `peek` at a synchronous queue because an element is only present when you try to remove it; you cannot insert an element (using any method) unless another thread is trying to remove it; you cannot iterate as there is nothing to iterate. The *head* of the queue is the element that the first queued inserting thread is trying to add to the queue; if there is no such queued thread then no element is available for removal and `poll()` (p. 2662) will return `null`. For purposes of other **Collection** (p. 851) methods (for example `contains`), a **SynchronousQueue** (p. 2655) acts as an empty collection. This queue does not permit `null` elements.

Synchronous queues are similar to rendezvous channels used in CSP and Ada. They are well suited for handoff designs, in which an object running in one thread must sync up with an object running in another thread in order to hand it some information, event, or task.

This class supports an optional fairness policy for ordering waiting producer and consumer threads. By default, this ordering is not guaranteed. However, a queue constructed with fairness set to `true` grants threads access in FIFO order.

This class and its iterator implement all of the *optional* methods of the **Collection** (p. 851) and **Iterator** (p. 1559) interfaces.

Since

1.0

6.532.2 Constructor & Destructor Documentation

6.532.2.1 `template<typename E > decaf::util::concurrent::SynchronousQueue< E >::SynchronousQueue () [inline]`

6.532.2.2 `template<typename E > virtual decaf::util::concurrent::SynchronousQueue< E >::~~SynchronousQueue () [inline, virtual]`

6.532.3 Member Function Documentation

6.532.3.1 `template<typename E > virtual void decaf::util::concurrent-
::SynchronousQueue< E >::clear () [inline,
virtual]`

Removes all of the elements from this collection (optional operation).

The collection will be empty after this method returns.

This implementation iterates over this collection, removing each element using the - **Iterator.remove** (p. 1560) operation. Most implementations will probably choose to override this method for efficiency.

Note that this implementation will throw an UnsupportedOperationException if the iterator returned by this collection's iterator method does not implement the remove method and this collection is non-empty.

Exceptions

<i>Unsupported- OperationException</i>	if the clear operation is not supported by this collection
--	--

This implementation repeatedly invokes poll until it returns false.

This implementation repeatedly invokes poll until it returns false.

Reimplemented from **decaf::util::AbstractQueue< E >** (p. 141).

6.532.3.2 `template<typename E > virtual bool decaf::util::concurrent::Synchronous-
Queue< E >::contains (const E &value DECAF_UNUSED) const [inline,
virtual]`

6.532.3.3 `template<typename E > virtual bool decaf::util::concurrent::Synchronous-
Queue< E >::containsAll (const Collection< E > & collection) const
[inline, virtual]`

Returns true if this collection contains all of the elements in the specified collection.

Parameters

<i>collection</i>	The Collection (p. 851) to compare to this one.
-------------------	--

Exceptions

<i>NullPointerException</i>	if the Collection (p. 851) contains pointers and the Collection (p. 851) does not allow for NULL elements (optional check).
-----------------------------	---

This implementation iterates over the specified collection, checking each element returned by the iterator in turn to see if it's contained in this collection. If all elements are so contained true is returned, otherwise false.

This implementation iterates over the specified collection, checking each element returned by the iterator in turn to see if it's contained in this collection. If all elements are

so contained true is returned, otherwise false.

Reimplemented from **decaf::util::AbstractCollection**< **E** > (p. 113).

References decaf::util::Collection< **E** >::isEmpty().

6.532.3.4 `template<typename E> virtual int decaf::util::concurrent::Synchronous-
Queue< E >::drainTo (Collection< E > & c) [inline,
virtual]`

Removes all available elements from this queue and adds them to the given collection.

This operation may be more efficient than repeatedly polling this queue. A failure encountered while attempting to add elements to collection *c* may result in elements being in neither, either or both collections when the associated exception is thrown. Attempts to drain a queue to itself result in `IllegalArgumentException`. Further, the behavior of this operation is undefined if the specified collection is modified while the operation is in progress.

Parameters

<i>c</i>	the collection to transfer elements into
----------	--

Returns

the number of elements transferred

Exceptions

<i>Unsupported-OperationException</i>	if addition of elements is not supported by the specified collection
<i>IllegalArgumentException</i>	if the specified collection is this queue, or some property of an element of this queue prevents it from being added to the specified collection

Implements **decaf::util::concurrent::BlockingQueue**< **E** > (p. 541).

References decaf::util::Collection< **E** >::add(), decaf::util::AbstractQueue< **E** >::element(), and decaf::util::concurrent::SynchronousQueue< **E** >::poll().

6.532.3.5 `template<typename E> virtual int decaf::util::concurrent::Synchronous-
Queue< E >::drainTo (Collection< E > & c, int maxElements)
[inline, virtual]`

Removes at most the given number of available elements from this queue and adds them to the given collection.

A failure encountered while attempting to add elements to collection *c* may result in elements being in neither, either or both collections when the associated exception is thrown. Attempts to drain a queue to itself result in `IllegalArgumentException`.

6.532 decaf::util::concurrent::SynchronousQueue< E > Class Template Reference

2667

Further, the behavior of this operation is undefined if the specified collection is modified while the operation is in progress.

Parameters

<i>c</i>	the collection to transfer elements into
<i>max-Elements</i>	the maximum number of elements to transfer

Returns

the number of elements transferred

Exceptions

<i>UnsupportedOperationException</i>	if addition of elements is not supported by the specified collection
<i>IllegalArgumentException</i>	if the specified collection is this queue, or some property of an element of this queue prevents it from being added to the specified collection

Implements **decaf::util::concurrent::BlockingQueue< E >** (p. 542).

References **decaf::util::Collection< E >::add()**, **decaf::util::AbstractQueue< E >::element()**, and **decaf::util::concurrent::SynchronousQueue< E >::poll()**.

6.532.3.6 `template<typename E> virtual bool decaf::util::concurrent::SynchronousQueue< E >::equals (const Collection< E > & collection) const [inline, virtual]`

Answers true if this **Collection** (p. 851) and the one given are the same size and if each element contained in the **Collection** (p. 851) given is equal to an element contained in this collection.

Parameters

<i>collection</i>	- The Collection (p. 851) to be compared to this one.
-------------------	--

Returns

true if this **Collection** (p. 851) is equal to the one given.

Reimplemented from **decaf::util::AbstractCollection< E >** (p. 114).

6.532.3.7 `template<typename E> virtual bool decaf::util::concurrent::SynchronousQueue< E >::isEmpty () const [inline, virtual]`

Returns true if this collection contains no elements.

This implementation returns **size()** (p. 2664) == 0.

Returns

true if the size method return 0.

Reimplemented from **decaf::util::AbstractCollection< E >** (p. 114).

```
6.532.3.8  template<typename E > virtual decaf::util::Iterator<E>*
           decaf::util::concurrent::SynchronousQueue< E >::iterator ( )
           [inline, virtual]
```

Returns

an iterator over a set of elements of type T.

Implements **decaf::lang::Iterable< E >** (p. 1557).

```
6.532.3.9  template<typename E > virtual decaf::util::Iterator<E>*
           decaf::util::concurrent::SynchronousQueue< E >::iterator ( ) const
           [inline, virtual]
```

Implements **decaf::lang::Iterable< E >** (p. 1558).

```
6.532.3.10 template<typename E > virtual bool decaf::util::concurrent::Synchronous-
           Queue< E >::offer ( const E & e, long long timeout, const TimeUnit & unit )
           [inline, virtual]
```

Inserts the specified element into this queue, waiting if necessary up to the specified wait time for another thread to receive it.

Returns

true if successful, or false if the specified waiting time elapses before a consumer appears.

Exceptions

<i>InterruptedException</i>	Inserts the specified element into this queue, waiting up to the specified wait time if necessary for space to become available.
<i>NullPointerException</i>	Inserts the specified element into this queue, waiting up to the specified wait time if necessary for space to become available.
<i>IllegalArgumentException</i>	Inserts the specified element into this queue, waiting up to the specified wait time if necessary for space to become available.

Implements **decaf::util::concurrent::BlockingQueue< E >** (p. 543).

6.532.3.11 `template<typename E > virtual bool decaf::util::concurrent::SynchronousQueue< E >::offer (const E & value) [inline, virtual]`

Inserts the specified element into this queue, if another thread is waiting to receive it.

Parameters

<i>value</i>	the element to add to the Queue (p. 2222)
--------------	--

Returns

`true` if the element was added to this queue, else `false`

Exceptions

<i>NullPointerException</i>	if the Queue (p. 2222) implementation does not allow Null values to be inserted into the Queue (p. 2222).
<i>IllegalArgumentException</i>	if some property of the specified element prevents it from being added to this queue

Implements **decaf::util::Queue< E >** (p. 2224).

6.532.3.12 `template<typename E > virtual bool decaf::util::concurrent::SynchronousQueue< E >::peek (E &result DECAF_UNUSED) const [inline, virtual]`

6.532.3.13 `template<typename E > virtual bool decaf::util::concurrent::SynchronousQueue< E >::poll (E &result, long long timeout, const TimeUnit & unit) [inline, virtual]`

Retrieves and removes the head of this queue, waiting if necessary up to the specified wait time, for another thread to insert it.

Parameters

<i>result</i>	a reference to the value where the head of the Queue (p. 2222) should be copied to.
<i>timeout</i>	the time that the method should block if there is no element available to return.
<i>unit</i>	the Time Units that the timeout value represents.

Returns

`true` if the head of the **Queue** (p. 2222) was copied to the result param or `false` if no value could be returned.

Implements **decaf::util::concurrent::BlockingQueue< E >** (p. 543).

Referenced by `decaf::util::concurrent::SynchronousQueue< E >::drainTo()`.

6.532.3.14 `template<typename E> virtual bool decaf::util::concurrent::-`
`SynchronousQueue< E >::poll (E & result)` `[inline,`
`virtual]`

Retrieves and removes the head of this queue, if another thread is currently making an element available.

Parameters

<i>result</i>	a reference to the value where the head of the Queue (p. 2222) should be copied to.
---------------	--

Returns

true if the head of the **Queue** (p. 2222) was copied to the result param or false if no value could be returned.

Implements **`decaf::util::Queue< E >`** (p. 2225).

6.532.3.15 `template<typename E> virtual void decaf::util::concurrent::-`
`SynchronousQueue< E >::put (const E & value)` `[inline,`
`virtual]`

Adds the specified element to this queue, waiting if necessary for another thread to receive it.

Parameters

<i>value</i>	the element to add to the Queue (p. 2222).
--------------	---

Exceptions

<i>InterruptedException</i>	Inserts the specified element into this queue, waiting if necessary for space to become available.
<i>NullPointerException</i>	Inserts the specified element into this queue, waiting if necessary for space to become available.
<i>IllegalArgumentException</i>	Inserts the specified element into this queue, waiting if necessary for space to become available.

Implements **`decaf::util::concurrent::BlockingQueue< E >`** (p. 544).

6.532.3.16 `template<typename E > virtual int decaf::util::concurrent::SynchronousQueue< E >::remainingCapacity () const [inline, virtual]`

Returns the number of additional elements that this queue can ideally (in the absence of memory or resource constraints) accept without blocking, or `Integer::MAX_VALUE` if there is no intrinsic limit.

Note that you *cannot* always tell if an attempt to insert an element will succeed by inspecting `remainingCapacity` because it may be the case that another thread is about to insert or remove an element.

Returns

the remaining capacity

Implements **decaf::util::concurrent::BlockingQueue< E >** (p. 544).

6.532.3.17 `template<typename E > virtual bool decaf::util::concurrent::SynchronousQueue< E >::remove (const E &value DECAF_UNUSED) [inline, virtual]`

6.532.3.18 `template<typename E > virtual bool decaf::util::concurrent::SynchronousQueue< E >::removeAll (const Collection< E > &collection DECAF_UNUSED) [inline, virtual]`

6.532.3.19 `template<typename E > virtual bool decaf::util::concurrent::SynchronousQueue< E >::retainAll (const Collection< E > &collection DECAF_UNUSED) [inline, virtual]`

6.532.3.20 `template<typename E > virtual int decaf::util::concurrent::SynchronousQueue< E >::size () const [inline, virtual]`

Returns the number of elements in this collection.

If this collection contains more than `Integer::MAX_VALUE` elements, returns `Integer::MAX_VALUE`.

Returns

the number of elements in this collection

Implements **decaf::util::Collection< E >** (p. 864).

6.532.3.21 `template<typename E > virtual E decaf::util::concurrent::SynchronousQueue< E >::take () [inline, virtual]`

Retrieves and removes the head of this queue, waiting if necessary for another thread to insert it.

Returns

the head of this queue

Exceptions

<i>InterruptedException</i>	Retrieves and removes the head of this queue, waiting if necessary until an element becomes available.
-----------------------------	--

Implements **decaf::util::concurrent::BlockingQueue< E >** (p. 545).

6.532.3.22 `template<typename E> virtual std::vector<E> decaf::util::concurrent-
::SynchronousQueue< E >::toArray () const [inline,
virtual]`

Answers an STL vector containing copies of all elements contained in this **Collection** (p. 851).

All the elements in the array will not be referenced by the collection. The elements in the returned array will be sorted to the same order as those returned by the iterator of this collection itself if the collection guarantees the order.

Returns

an vector of copies of all the elements from this **Collection** (p. 851)

Reimplemented from **decaf::util::AbstractCollection< E >** (p. 119).

The documentation for this class was generated from the following file:

- `src/main/decaf/util/concurrent/SynchronousQueue.h`

6.533 decaf::lang::System Class Reference

The **System** (p. 2665) class provides static methods for accessing system level resources and performing some system dependent tasks such as looking up environment values and copying memory and arrays.

```
#include <src/main/decaf/lang/System.h>
```

Public Member Functions

- `virtual ~System ()`

Static Public Member Functions

- `static void arraycopy (const char *src, std::size_t srcPos, char *dest, std::size_t destPos, std::size_t length)`

Copies the number of elements specified by length from the source array starting at the given source offset specified by srcPos to the dest array starting at the given destination offset given by destPos.

- static void **arraycopy** (const unsigned char *src, std::size_t srcPos, unsigned char *dest, std::size_t destPos, std::size_t length)

Copies the number of elements specified by length from the source array starting at the given source offset specified by srcPos to the dest array starting at the given destination offset given by destPos.

- static void **arraycopy** (const short *src, std::size_t srcPos, short *dest, std::size_t destPos, std::size_t length)

Copies the number of elements specified by length from the source array starting at the given source offset specified by srcPos to the dest array starting at the given destination offset given by destPos.

- static void **arraycopy** (const int *src, std::size_t srcPos, int *dest, std::size_t destPos, std::size_t length)

Copies the number of elements specified by length from the source array starting at the given source offset specified by srcPos to the dest array starting at the given destination offset given by destPos.

- static void **arraycopy** (const long long *src, std::size_t srcPos, long long *dest, std::size_t destPos, std::size_t length)

Copies the number of elements specified by length from the source array starting at the given source offset specified by srcPos to the dest array starting at the given destination offset given by destPos.

- static void **arraycopy** (const float *src, std::size_t srcPos, float *dest, std::size_t destPos, std::size_t length)

Copies the number of elements specified by length from the source array starting at the given source offset specified by srcPos to the dest array starting at the given destination offset given by destPos.

- static void **arraycopy** (const double *src, std::size_t srcPos, double *dest, std::size_t destPos, std::size_t length)

Copies the number of elements specified by length from the source array starting at the given source offset specified by srcPos to the dest array starting at the given destination offset given by destPos.

- template<typename E >
static void **arraycopy** (const E *src, std::size_t srcPos, E *dest, std::size_t destPos, std::size_t length)

Copies the number of elements specified by length from the source array starting at the given source offset specified by srcPos to the dest array starting at the given destination offset given by destPos.

- static const **util::Map** < std::string, std::string > & **getenv** ()

Enumerates the system environment and returns a map of env variable names to the string values they hold.

- static std::string **getenv** (const std::string &name)

Reads an environment value from the system and returns it as a string object.

- static void **unsetenv** (const std::string &name)

Clears a set environment value if one is set.

- static void **setenv** (const std::string &name, const std::string &value)

Sets the specified system property to the value given.

- static long long **currentTimeMillis** ()
Returns the current time in milliseconds.
- static long long **nanoTime** ()
Returns the current value of the most precise available system timer, in nanoseconds.
- static int **availableProcessors** ()
Returns the number of processors available for execution of Decaf Threads.
- static **decaf::util::Properties** & **getProperties** ()
Gets the Properties object that holds the Properties accessed from calls to getProperty and setProperty.
- static std::string **getProperty** (const std::string &key)
*Gets the specified **System** (p. 2665) property if set, otherwise returns an empty string.*
- static std::string **getProperty** (const std::string &key, const std::string &default-Value)
*Gets the specified **System** (p. 2665) property if set, otherwise returns the specified default value.*
- static std::string **setProperty** (const std::string &key, const std::string &value)
*Sets the **System** (p. 2665) Property to the specified value.*
- static std::string **clearProperty** (const std::string &key)
Clear any value associated with the system property specified.

Protected Member Functions

- **System** ()

Friends

- class **decaf::lang::Runtime**

6.533.1 Detailed Description

The **System** (p. 2665) class provides static methods for accessing system level resources and performing some system dependent tasks such as looking up environment values and copying memory and arrays.

Since

1.0

6.533.2 Constructor & Destructor Documentation

6.533.2.1 **decaf::lang::System::System** () [protected]

6.533.2.2 **virtual decaf::lang::System::~~System** () [inline, virtual]

6.533.3 Member Function Documentation

6.533.3.1 static void decaf::lang::System::arraycopy (const char * *src*, std::size_t *srcPos*, char * *dest*, std::size_t *destPos*, std::size_t *length*) [static]

Copies the number of elements specified by length from the source array starting at the given source offset specified by srcPos to the dest array starting at the given destination offset given by destPos.

Parameters

<i>src</i>	The source array to copy from.
<i>srcPos</i>	The position in the array to start copying from.
<i>dest</i>	The destination array to copy to.
<i>destPos</i>	The position in the destination array to start writing at.
<i>length</i>	The number of elements to copy from src to dest.

Exceptions

<i>NullPointerException</i>	if src or dest are NULL.
-----------------------------	--------------------------

Referenced by decaf::util::concurrent::CopyOnWriteArrayList< ServiceListener * >::add(), decaf::util::concurrent::CopyOnWriteArrayList< ServiceListener * >::addAll(), decaf::util::concurrent::CopyOnWriteArrayList< ServiceListener * >::addAllAbsent(), decaf::util::concurrent::CopyOnWriteArrayList< ServiceListener * >::addIfAbsent(), decaf::lang::ArrayPointer< decaf::util::LinkedList< Pointer< MessageDispatch > >::clone(), decaf::util::ArrayList< E >::ensureCapacity(), decaf::util::concurrent::CopyOnWriteArrayList< ServiceListener * >::removeAll(), decaf::util::ArrayList< E >::removeAt(), decaf::util::concurrent::CopyOnWriteArrayList< ServiceListener * >::removeAt(), decaf::util::concurrent::CopyOnWriteArrayList< ServiceListener * >::retainAll(), decaf::util::concurrent::CopyOnWriteArrayList< ServiceListener * >::set(), and decaf::util::ArrayList< E >::trimToSize().

6.533.3.2 static void decaf::lang::System::arraycopy (const unsigned char * *src*, std::size_t *srcPos*, unsigned char * *dest*, std::size_t *destPos*, std::size_t *length*) [static]

Copies the number of elements specified by length from the source array starting at the given source offset specified by srcPos to the dest array starting at the given destination offset given by destPos.

Parameters

<i>src</i>	The source array to copy from.
<i>srcPos</i>	The position in the array to start copying from.
<i>dest</i>	The destination array to copy to.
<i>destPos</i>	The position in the destination array to start writing at.
<i>length</i>	The number of elements to copy from src to dest.

Exceptions

<i>NullPointerException</i>	if <i>src</i> or <i>dest</i> are NULL.
-----------------------------	--

6.533.3.3 `static void decaf::lang::System::arraycopy (const short * src, std::size_t srcPos, short * dest, std::size_t destPos, std::size_t length) [static]`

Copies the number of elements specified by *length* from the source array starting at the given source offset specified by *srcPos* to the dest array starting at the given destination offset given by *destPos*.

Parameters

<i>src</i>	The source array to copy from.
<i>srcPos</i>	The position in the array to start copying from.
<i>dest</i>	The destination array to copy to.
<i>destPos</i>	The position in the destination array to start writing at.
<i>length</i>	The number of elements to copy from <i>src</i> to <i>dest</i> .

Exceptions

<i>NullPointerException</i>	if <i>src</i> or <i>dest</i> are NULL.
-----------------------------	--

6.533.3.4 `static void decaf::lang::System::arraycopy (const int * src, std::size_t srcPos, int * dest, std::size_t destPos, std::size_t length) [static]`

Copies the number of elements specified by *length* from the source array starting at the given source offset specified by *srcPos* to the dest array starting at the given destination offset given by *destPos*.

Parameters

<i>src</i>	The source array to copy from.
<i>srcPos</i>	The position in the array to start copying from.
<i>dest</i>	The destination array to copy to.
<i>destPos</i>	The position in the destination array to start writing at.
<i>length</i>	The number of elements to copy from <i>src</i> to <i>dest</i> .

Exceptions

<i>NullPointerException</i>	if <i>src</i> or <i>dest</i> are NULL.
-----------------------------	--

6.533.3.5 static void decaf::lang::System::arraycopy (const long long * *src*, std::size_t *srcPos*, long long * *dest*, std::size_t *destPos*, std::size_t *length*) [static]

Copies the number of elements specified by length from the source array starting at the given source offset specified by srcPos to the dest array starting at the given destination offset given by destPos.

Parameters

<i>src</i>	The source array to copy from.
<i>srcPos</i>	The position in the array to start copying from.
<i>dest</i>	The destination array to copy to.
<i>destPos</i>	The position in the destination array to start writing at.
<i>length</i>	The number of elements to copy from src to dest.

Exceptions

<i>NullPointerException</i>	if src or dest are NULL.
-----------------------------	--------------------------

6.533.3.6 static void decaf::lang::System::arraycopy (const float * *src*, std::size_t *srcPos*, float * *dest*, std::size_t *destPos*, std::size_t *length*) [static]

Copies the number of elements specified by length from the source array starting at the given source offset specified by srcPos to the dest array starting at the given destination offset given by destPos.

Parameters

<i>src</i>	The source array to copy from.
<i>srcPos</i>	The position in the array to start copying from.
<i>dest</i>	The destination array to copy to.
<i>destPos</i>	The position in the destination array to start writing at.
<i>length</i>	The number of elements to copy from src to dest.

Exceptions

<i>NullPointerException</i>	if src or dest are NULL.
-----------------------------	--------------------------

6.533.3.7 static void decaf::lang::System::arraycopy (const double * *src*, std::size_t *srcPos*, double * *dest*, std::size_t *destPos*, std::size_t *length*) [static]

Copies the number of elements specified by length from the source array starting at the given source offset specified by srcPos to the dest array starting at the given destination offset given by destPos.

Parameters

<i>src</i>	The source array to copy from.
<i>srcPos</i>	The position in the array to start copying from.
<i>dest</i>	The destination array to copy to.
<i>destPos</i>	The position in the destination array to start writing at.
<i>length</i>	The number of elements to copy from src to dest.

Exceptions

<i>NullPointerException</i>	if src or dest are NULL.
-----------------------------	--------------------------

```
6.533.3.8  template<typename E > static void decaf::lang::System::arraycopy ( const E
           * src, std::size_t srcPos, E * dest, std::size_t destPos, std::size_t length )
           [inline, static]
```

Copies the number of elements specified by length from the source array starting at the given source offset specified by srcPos to the dest array starting at the given destination offset given by destPos.

Parameters

<i>src</i>	The source array to copy from.
<i>srcPos</i>	The position in the array to start copying from.
<i>dest</i>	The destination array to copy to.
<i>destPos</i>	The position in the destination array to start writing at.
<i>length</i>	The number of elements to copy from src to dest.

Exceptions

<i>NullPointerException</i>	if src or dest are NULL.
-----------------------------	--------------------------

```
6.533.3.9  static int decaf::lang::System::availableProcessors ( ) [static]
```

Returns the number of processors available for execution of Decaf Threads.

This value may change during a particular execution of a Decaf based application. - Applications that are sensitive to the number of available processors should therefore occasionally poll this property and adjust their resource usage appropriately.

Returns

the number of available processors.

6.533.3.10 `static std::string decaf::lang::System::clearProperty (const std::string & key) [static]`

Clear any value associated with the system property specified.

Parameters

<i>key</i>	The key name of the system property to clear.
------------	---

Returns

the previous value of the property named by key if there was one, otherwise returns an empty string.

Exceptions

<i>IllegalArgument-Exception</i>	if key is an empty string.
----------------------------------	----------------------------

6.533.3.11 `static long long decaf::lang::System::currentTimeMillis () [static]`

Returns the current time in milliseconds.

Note that while the unit of time of the return value is a millisecond, the granularity of the value depends on the underlying operating system and may be larger. For example, many operating systems measure time in units of tens of milliseconds.

See the description of the class Date for a discussion of slight discrepancies that may arise between "computer time" and coordinated universal time (UTC).

Returns

the difference, measured in milliseconds, between the current time and midnight, January 1, 1970 UTC.

6.533.3.12 `static const util::Map<std::string, std::string>& decaf::lang::System::getenv () [static]`

Enumerates the system environment and returns a map of env variable names to the string values they hold.

Returns

A Map of all environment variables.

Exceptions

Exception (p. 1279)	if an error occurs while getting the Environment Map.
----------------------------	---

6.533.3.13 `static std::string decaf::lang::System::getenv (const std::string & name)`
[static]

Reads an environment value from the system and returns it as a string object.

Parameters

<i>name</i>	The environment variable to read.
-------------	-----------------------------------

Returns

a string with the value from the variables or ""

Exceptions

<i>an</i>	Exception (p. 1279) if an error occurs while reading the Env.
-----------	--

6.533.3.14 `static decaf::util::Properties& decaf::lang::System::getProperties ()`
[static]

Gets the Properties object that holds the Properties accessed from calls to getProperty and setProperty.

If the Properties has not yet been created or are not yet initialized then they will be on the first call to a Properties accessor.

Returns

a reference to the static system Properties object.

6.533.3.15 `static std::string decaf::lang::System::getProperty (const std::string & key)`
[static]

Gets the specified **System** (p. 2665) property if set, otherwise returns an empty string.

If the Properties has not yet been created or are not yet initialized then they will be on the first call to a Properties accessor.

Parameters

<i>key</i>	The key name of the desired system property to retrieve.
------------	--

Returns

an empty string if the named property is not set, otherwise returns the value.

Exceptions

<i>IllegalArgument-Exception</i>	if key is an empty string.
----------------------------------	----------------------------

6.533.3.16 static std::string decaf::lang::System::getProperty (const std::string & key, const std::string & defaultValue) [static]

Gets the specified **System** (p.2665) property if set, otherwise returns the specified default value.

If the Properties has not yet been created or are not yet initialized then they will be on the first call to a Properties accessor.

Parameters

<i>key</i>	The key name of the desired system property to retrieve.
<i>defaultValue</i>	The default value to return if the key is not set in the System (p.2665) properties.

Returns

the value of the named system property or the defaultValue if the property isn't set..

Exceptions

<i>IllegalArgument-Exception</i>	if key is an empty string.
----------------------------------	----------------------------

6.533.3.17 static long long decaf::lang::System::nanoTime () [static]

Returns the current value of the most precise available system timer, in nanoseconds.

This method can only be used to measure elapsed time and is not related to any other notion of system or wall-clock time. The value returned represents nanoseconds since some fixed but arbitrary time (perhaps in the future, so values may be negative). This method provides nanosecond precision, but not necessarily nanosecond accuracy. No guarantees are made about how frequently values change. Differences in successive calls that span greater than approximately 292 years (263 nanoseconds) will not accurately compute elapsed time due to numerical overflow.

For example, to measure how long some code takes to execute:

```
long long startTime = System::nanoTime() (p.2674); // ... the code being measured ...
long long estimatedTime = System::nanoTime() (p.2674) - startTime;
```

Returns

The current value of the system timer, in nanoseconds.

6.533.3.18 `static void decaf::lang::System::setenv (const std::string & name, const std::string & value) [static]`

Sets the specified system property to the value given.

Parameters

<i>name</i>	The name of the environment variables to set.
<i>value</i>	The value to assign to name.

Exceptions

<i>an</i>	Exception (p. 1279) if an error occurs when setting the environment variable.
-----------	--

6.533.3.19 `static std::string decaf::lang::System::setProperty (const std::string & key, const std::string & value) [static]`

Sets the **System** (p. 2665) Property to the specified value.

Parameters

<i>key</i>	The key name of the system property to set to the given value.
<i>value</i>	The value to assign to the key.

Returns

the previous value of the property named by key if there was one, otherwise returns an empty string.

Exceptions

<i>IllegalArgument-Exception</i>	if key is an empty string.
----------------------------------	----------------------------

6.533.3.20 `static void decaf::lang::System::unsetenv (const std::string & name) [static]`

Clears a set environment value if one is set.

Parameters

<i>name</i>	The environment variables to clear.
-------------	-------------------------------------

Exceptions

<i>an</i>	Exception (p. 1279) if an error occurs while reading the environment.
-----------	--

6.533.4 Friends And Related Function Documentation

6.533.4.1 friend class decaf::lang::Runtime [friend]

The documentation for this class was generated from the following file:

- src/main/decaf/lang/**System.h**

6.534 activemq::threads::Task Class Reference

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

```
#include <src/main/activemq/threads/Task.h>
```

Inheritance diagram for activemq::threads::Task:

Public Member Functions

- virtual **~Task** ()
- virtual bool **iterate** ()=0

Perform one iteration of work, returns true if the task needs to run again to complete or false to indicate that the task is now complete.

6.534.1 Detailed Description

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

Since

3.0

6.534.2 Constructor & Destructor Documentation

6.534.2.1 virtual activemq::threads::Task::~Task () [inline, virtual]

6.534.3 Member Function Documentation

6.534.3.1 virtual bool `activemq::threads::Task::iterate ()` [pure virtual]

Perform one iteration of work, returns true if the task needs to run again to complete or false to indicate that the task is now complete.

Returns

true if the task should be run again or false if the task has completed and the runner should wait for a wakeup call.

Implemented in `activemq::transport::failover::FailoverTransport` (p. 1312), `activemq::core::ActiveMQSessionExecutor` (p. 358), `activemq::threads::CompositeTaskRunner` (p. 895), `activemq::transport::failover::BackupTransportPool` (p. 491), and `activemq::transport::failover::CloseTransportsTask` (p. 819).

The documentation for this class was generated from the following file:

- `src/main/activemq/threads/Task.h`

6.535 `activemq::threads::TaskRunner` Class Reference

```
#include <src/main/activemq/threads/TaskRunner.h>
```

Inheritance diagram for `activemq::threads::TaskRunner`:

Public Member Functions

- virtual `~TaskRunner ()`
- virtual void `shutdown (unsigned int timeout)=0`
Shutdown after a timeout, does not guarantee that the task's iterate method has completed and the thread halted.
- virtual void `shutdown ()=0`
*Shutdown once the task has finished and the **TaskRunner** (p. 2677)'s thread has exited.*
- virtual void `wakeup ()=0`
*Signal the **TaskRunner** (p. 2677) to wakeup and execute another iteration cycle on the task, the **Task** (p. 2676) instance will be run until its iterate method has returned false indicating it is done.*

6.535.1 Constructor & Destructor Documentation

6.535.1.1 virtual `activemq::threads::TaskRunner::~TaskRunner ()` [inline, virtual]

6.535.2 Member Function Documentation

6.535.2.1 `virtual void activemq::threads::TaskRunner::shutdown (unsigned int timeout) [pure virtual]`

Shutdown after a timeout, does not guarantee that the task's iterate method has completed and the thread halted.

Parameters

<i>timeout</i>	- Time in Milliseconds to wait for the task to stop.
----------------	--

Implemented in **activemq::threads::CompositeTaskRunner** (p. 895), and **activemq::threads::DedicatedTaskRunner** (p. 1145).

6.535.2.2 `virtual void activemq::threads::TaskRunner::shutdown () [pure virtual]`

Shutdown once the task has finished and the **TaskRunner** (p. 2677)'s thread has exited.

Implemented in **activemq::threads::CompositeTaskRunner** (p. 896), and **activemq::threads::DedicatedTaskRunner** (p. 1145).

6.535.2.3 `virtual void activemq::threads::TaskRunner::wakeup () [pure virtual]`

Signal the **TaskRunner** (p. 2677) to wakeup and execute another iteration cycle on the task, the **Task** (p. 2676) instance will be run until its iterate method has returned false indicating it is done.

Implemented in **activemq::threads::CompositeTaskRunner** (p. 896), and **activemq::threads::DedicatedTaskRunner** (p. 1145).

The documentation for this class was generated from the following file:

- `src/main/activemq/threads/TaskRunner.h`

6.536 decaf::internal::net::tcp::TcpSocket Class Reference

Platform-independent implementation of the socket interface.

```
#include <src/main/decaf/internal/net/tcp/TcpSocket.h>
```

Inheritance diagram for decaf::internal::net::tcp::TcpSocket:

Public Member Functions

• **TcpSocket** ()

Construct a non-connected socket.

• virtual **~TcpSocket** ()

Releases the socket handle but not gracefully shut down the connection.

• SocketHandle **getSocketHandle** ()

Gets the handle for the socket.

• bool **isConnected** () const• bool **isClosed** () const• virtual std::string **getLocalAddress** () const

*Gets the value of the local **Inet** address the **Socket** (p. 2452) is bound to if bound, otherwise return the **InetAddress** (p. 1437) ANY value "0.0.0.0".*

Returns

the local address bound to, or ANY.

• virtual void **create** ()

*Creates the underlying platform **Socket** (p. 2452) data structures which allows for - **Socket** (p. 2452) options to be applied.*

The created socket is in an unconnected state.

Exceptions

IOException	<i>if an I/O error occurs while attempting this operation.</i>
-------------	--

• virtual void **accept** (SocketImpl *socket)• virtual void **bind** (const std::string &ipaddress, int port)

*Binds this **Socket** (p. 2452) instance to the local ip address and port number given.*

Parameters

ipaddress	<i>The address of local ip to bind to.</i>
port	<i>The port number on the host to bind to.</i>

Exceptions

IOException	<i>if an I/O error occurs while attempting this operation.</i>
-------------	--

• virtual void **connect** (const std::string &hostname, int port, int timeout)

Connects this socket to the given host and port.

Parameters

hostname	<i>The name of the host to connect to, or IP address.</i>
port	<i>The port number on the host to connect to.</i>
timeout	<i>Time in milliseconds to wait for a connection, 0 indicates forever.</i>

Exceptions

IOException	<i>if an I/O error occurs while attempting this operation.</i>
SocketTimeout-Exception (p. 2493)	<i>if the connect call times out due to timeout being set.</i>
IllegalArgument-Exception	<i>if a parameter has an illegal value.</i>

- virtual void **listen** (int backlog)

Sets the maximum queue length for incoming connection indications (a request to connect) to the count argument.

If a connection indication arrives when the queue is full, the connection is refused.

Parameters

backlog	The maximum length of the connection queue.
---------	---

Exceptions

IOException	if an I/O error occurs while attempting this operation.
-------------	---

- virtual **decaf::io::InputStream * getInputStream** ()

*Gets the InputStream linked to this **Socket** (p. 2452).*

Returns

*an InputStream pointer owned by the **Socket** (p. 2452) object.*

Exceptions

IOException	if an I/O error occurs while attempting this operation.
-------------	---

- virtual **decaf::io::OutputStream * getOutputStream** ()

*Gets the OutputStream linked to this **Socket** (p. 2452).*

Returns

*an OutputStream pointer owned by the **Socket** (p. 2452) object.*

Exceptions

IOException	if an I/O error occurs while attempting this operation.
-------------	---

- virtual int **available** ()

*Gets the number of bytes that can be read from the **Socket** (p. 2452) without blocking.*

Returns

*the number of bytes that can be read from the **Socket** (p. 2452) without blocking.*

Exceptions

IOException	if an I/O error occurs while attempting this operation.
-------------	---

- virtual void **close** ()

Closes the socket, terminating any blocked reads or writes.

Exceptions

IOException	if an I/O error occurs while attempting this operation.
-------------	---

- virtual void **shutdownInput** ()

Places the input stream for this socket at "end of stream".

*Any data sent to this socket is acknowledged and then silently discarded. If you read from a socket input stream after invoking **shutdownInput()** (p. 2486) on the socket, the stream will return EOF.*

Exceptions

IOException	if an I/O error occurs while attempting this operation.
-------------	---

- virtual void **shutdownOutput** ()

Disables the output stream for this socket.

For a TCP socket, any previously written data will be sent followed by TCP's normal connection termination sequence. If you write to a socket output stream after invoking **shutdownOutput()** (p. 2486) on the socket, the stream will throw an *IOException*.

Exceptions

<i>IOException</i>	if an I/O error occurs while attempting this operation.
--------------------	---

- virtual int **getOption** (int option) const

*Gets the specified **Socket** (p. 2452) option.*

Parameters

option	The Socket (p. 2452) options whose value is to be retrieved.
--------	---

Returns

the value of the given socket option.

Exceptions

<i>IOException</i>	if an I/O error occurs while performing this operation.
--------------------	---

- virtual void **setOption** (int option, int value)

*Sets the specified option on the **Socket** (p. 2452) if supported.*

Parameters

option	The Socket (p. 2452) option to set.
--------	--

value	The value of the socket option to apply to the socket.
-------	--

Exceptions

<i>IOException</i>	if an I/O error occurs while performing this operation.
--------------------	---

- int **read** (unsigned char *buffer, int size, int offset, int length)

Reads the requested data from the Socket and write it into the passed in buffer.

- void **write** (const unsigned char *buffer, int size, int offset, int length)

Writes the specified data in the passed in buffer to the Socket.

Protected Member Functions

- void **checkResult** (apr_status_t value) const

6.536.1 Detailed Description

Platform-independent implementation of the socket interface.

6.536.2 Constructor & Destructor Documentation

6.536.2.1 decaf::internal::net::tcp::TcpSocket::TcpSocket ()

Construct a non-connected socket.

Exceptions

<i>SocketException</i>	thrown if an error occurs while creating the Socket.
------------------------	--

6.536.2.2 virtual **decaf::internal::net::tcp::TcpSocket::~~TcpSocket** ()
[virtual]

Releases the socket handle but not gracefully shut down the connection.

6.536.3 Member Function Documentation

6.536.3.1 virtual void **decaf::internal::net::tcp::TcpSocket::accept** (SocketImpl *
socket) [virtual]

6.536.3.2 virtual int **decaf::internal::net::tcp::TcpSocket::available** ()
[virtual]

Gets the number of bytes that can be read from the **Socket** (p. 2452) without blocking.

Returns

the number of bytes that can be read from the **Socket** (p. 2452) without blocking.

Exceptions

<i>IOException</i>	if an I/O error occurs while attempting this operation.
--------------------	---

Implements **decaf::net::SocketImpl** (p. 2481).

6.536.3.3 virtual void **decaf::internal::net::tcp::TcpSocket::bind** (const std::string &
ipaddress, int *port*) [virtual]

Binds this **Socket** (p. 2452) instance to the local ip address and port number given.

Parameters

<i>ipaddress</i>	The address of local ip to bind to.
<i>port</i>	The port number on the host to bind to.

Exceptions

<i>IOException</i>	if an I/O error occurs while attempting this operation.
--------------------	---

Implements **decaf::net::SocketImpl** (p. 2482).

6.536.3.4 `void decaf::internal::net::tcp::TcpSocket::checkResult (apr_status_t value) const` [protected]

6.536.3.5 `virtual void decaf::internal::net::tcp::TcpSocket::close ()` [virtual]

Closes the socket, terminating any blocked reads or writes.

Exceptions

<i>IOException</i>	if an I/O error occurs while attempting this operation.
--------------------	---

Implements **decaf::net::SocketImpl** (p. 2482).

6.536.3.6 `virtual void decaf::internal::net::tcp::TcpSocket::connect (const std::string & hostname, int port, int timeout)` [virtual]

Connects this socket to the given host and port.

Parameters

<i>hostname</i>	The name of the host to connect to, or IP address.
<i>port</i>	The port number on the host to connect to.
<i>timeout</i>	Time in milliseconds to wait for a connection, 0 indicates forever.

Exceptions

<i>IOException</i>	if an I/O error occurs while attempting this operation.
SocketTimeout-Exception (p. 2493)	if the connect call times out due to timeout being set.
<i>IllegalArgument-Exception</i>	if a parameter has an illegal value.

Implements **decaf::net::SocketImpl** (p. 2482).

6.536.3.7 `virtual void decaf::internal::net::tcp::TcpSocket::create ()` [virtual]

Creates the underlying platform **Socket** (p. 2452) data structures which allows for **Socket** (p. 2452) options to be applied.

The created socket is in an unconnected state.

Exceptions

<i>IOException</i>	if an I/O error occurs while attempting this operation.
--------------------	---

Implements **decaf::net::SocketImpl** (p. 2483).

6.536.3.8 virtual **decaf::io::InputStream*** **decaf::internal::net::tcp::TcpSocket::get-InputStream ()** [virtual]

Gets the InputStream linked to this **Socket** (p. 2452).

Returns

an InputStream pointer owned by the **Socket** (p. 2452) object.

Exceptions

<i>IOException</i>	if an I/O error occurs while attempting this operation.
--------------------	---

Implements **decaf::net::SocketImpl** (p. 2483).

6.536.3.9 virtual std::string **decaf::internal::net::tcp::TcpSocket::getLocalAddress ()** const [virtual]

Gets the value of the local Inet address the **Socket** (p. 2452) is bound to if bound, otherwise return the **InetAddress** (p. 1437) ANY value "0.0.0.0".

Returns

the local address bound to, or ANY.

Implements **decaf::net::SocketImpl** (p. 2484).

6.536.3.10 virtual int **decaf::internal::net::tcp::TcpSocket::getOption (int option)** const [virtual]

Gets the specified **Socket** (p. 2452) option.

Parameters

<i>option</i>	The Socket (p. 2452) options whose value is to be retrieved.
---------------	---

Returns

the value of the given socket option.

Exceptions

<i>IOException</i>	if an I/O error occurs while performing this operation.
--------------------	---

Implements **decaf::net::SocketImpl** (p. 2484).

6.536.3.11 **virtual** **decaf::io::OutputStream*** **decaf::internal::net::tcp::TcpSocket::getOutputStream ()** *[virtual]*

Gets the OutputStream linked to this **Socket** (p. 2452).

Returns

an OutputStream pointer owned by the **Socket** (p. 2452) object.

Exceptions

<i>IOException</i>	if an I/O error occurs while attempting this operation.
--------------------	---

Implements **decaf::net::SocketImpl** (p. 2484).

6.536.3.12 **SocketHandle** **decaf::internal::net::tcp::TcpSocket::getSocketHandle ()** *[inline]*

Gets the handle for the socket.

Returns

SocketHabler for this Socket, can be NULL

6.536.3.13 **bool** **decaf::internal::net::tcp::TcpSocket::isClosed ()** **const** *[inline]*

Returns

true if the close method has been called on this Socket.

6.536.3.14 **bool** **decaf::internal::net::tcp::TcpSocket::isConnected ()** **const** *[inline]*

Returns

true if the socketHandle is not in a disconnected state.

6.536.3.15 **virtual void** **decaf::internal::net::tcp::TcpSocket::listen (int backlog)** *[virtual]*

Sets the maximum queue length for incoming connection indications (a request to connect) to the count argument.

If a connection indication arrives when the queue is full, the connection is refused.

Parameters

<i>backlog</i>	The maximum length of the connection queue.
----------------	---

Exceptions

<i>IOException</i>	if an I/O error occurs while attempting this operation.
--------------------	---

Implements **decaf::net::SocketImpl** (p. 2485).

6.536.3.16 `int decaf::internal::net::tcp::TcpSocket::read (unsigned char * buffer, int size, int offset, int length)`

Reads the requested data from the Socket and write it into the passed in buffer.

Parameters

<i>buffer</i>	The buffer to read into
<i>size</i>	The size of the specified buffer
<i>offset</i>	The offset into the buffer where reading should start filling.
<i>length</i>	The number of bytes past offset to fill with data.

Returns

the actual number of bytes read or -1 if at EOF.

Exceptions

<i>IOException</i>	if an I/O error occurs during the read.
<i>NullPointerException</i>	if buffer is Null.
<i>IndexOutOfBoundsException</i>	if offset + length is greater than buffer size.

6.536.3.17 `virtual void decaf::internal::net::tcp::TcpSocket::setOption (int option, int value)` [virtual]

Sets the specified option on the **Socket** (p. 2452) if supported.

Parameters

<i>option</i>	The Socket (p. 2452) option to set.
<i>value</i>	The value of the socket option to apply to the socket.

Exceptions

<i>IOException</i>	if an I/O error occurs while performing this operation.
--------------------	---

Implements **decaf::net::SocketImpl** (p. 2486).

6.536.3.18 virtual void **decaf::internal::net::tcp::TcpSocket::shutdownInput** ()
[virtual]

Places the input stream for this socket at "end of stream".

Any data sent to this socket is acknowledged and then silently discarded. If you read from a socket input stream after invoking **shutdownInput()** (p. 2486) on the socket, the stream will return EOF.

Exceptions

<i>IOException</i>	if an I/O error occurs while attempting this operation.
--------------------	---

Implements **decaf::net::SocketImpl** (p. 2486).

6.536.3.19 virtual void **decaf::internal::net::tcp::TcpSocket::shutdownOutput** ()
[virtual]

Disables the output stream for this socket.

For a TCP socket, any previously written data will be sent followed by TCP's normal connection termination sequence. If you write to a socket output stream after invoking **shutdownOutput()** (p. 2486) on the socket, the stream will throw an *IOException*.

Exceptions

<i>IOException</i>	if an I/O error occurs while attempting this operation.
--------------------	---

Implements **decaf::net::SocketImpl** (p. 2486).

6.536.3.20 void **decaf::internal::net::tcp::TcpSocket::write** (const unsigned char *
buffer, int *size*, int *offset*, int *length*)

Writes the specified data in the passed in buffer to the Socket.

Parameters

<i>buffer</i>	The buffer to write to the socket.
<i>size</i>	The size of the specified buffer.
<i>offset</i>	The offset into the buffer where the data to write starts at.
<i>length</i>	The number of bytes past offset to write.

Exceptions

<i>IOException</i>	if an I/O error occurs during the write.
<i>NullPointerException</i>	if buffer is Null.

<i>IndexOutOfBounds-Exception</i>	if offset + length is greater than buffer size.
-----------------------------------	---

The documentation for this class was generated from the following file:

- src/main/decaf/internal/net/tcp/TcpSocket.h

6.537 decaf::internal::net::tcp::TcpSocketInputStream Class - Reference

Input stream for performing reads on a socket.

```
#include <src/main/decaf/internal/net/tcp/TcpSocketInputStream.h>
```

Inheritance diagram for decaf::internal::net::tcp::TcpSocketInputStream:

Public Member Functions

- **TcpSocketInputStream** (**TcpSocket** *socket)
Create a new InputStream to use for reading from the TCP/IP socket.
- virtual ~**TcpSocketInputStream** ()
- virtual int **available** () const
*Indicates the number of bytes available.
The default implementation of this methods returns 0. Classes that override this method may return the total number of bytes that are currently available for reading and others may simply return a value of one indicating that there is some data available. The caller should view the result of this method as an absolute.
The default implementation of this method returns zero.*
Returns
the number of bytes available on this input stream.

Exceptions

IOException (p. 1545)	if an I/O error occurs.
------------------------------	-------------------------

- virtual void **close** ()
Close - does nothing.
- virtual long long **skip** (long long num)
Not supported.

Protected Member Functions

- virtual int **doReadByte** ()

- virtual int **doReadArrayBounded** (unsigned char *buffer, int size, int offset, int length)

6.537.1 Detailed Description

Input stream for performing reads on a socket.

This class will only work properly for blocking sockets.

Since

1.0

6.537.2 Constructor & Destructor Documentation

6.537.2.1 decaf::internal::net::tcp::TcpSocketInputStream::TcpSocketInputStream (TcpSocket * *socket*)

Create a new InputStream to use for reading from the TCP/IP socket.

Parameters

<i>socket</i>	The parent SocketImpl for this stream.
---------------	--

6.537.2.2 virtual decaf::internal::net::tcp::TcpSocketInputStream::~~TcpSocket- InputStream () [virtual]

6.537.3 Member Function Documentation

6.537.3.1 virtual int decaf::internal::net::tcp::TcpSocketInputStream::available () const [virtual]

Indicates the number of bytes available.

The default implementation of this methods returns 0. Classes that override this method may return the total number of bytes that are currently available for reading and others may simply return a value of one indicating that there is some data available. The caller should view the result of this method as an absolute.

The default implementation of this method returns zero.

Returns

the number of bytes available on this input stream.

Exceptions

<i>IOException</i> (p. 1545)	if an I/O error occurs.
--	-------------------------

Reimplemented from **decaf::io::InputStream** (p. 1466).

6.537.3.2 `virtual void decaf::internal::net::tcp::TcpSocketInputStream::close ()`
[virtual]

Close - does nothing.

It is the responsibility of the owner of the socket object to close it.

Closes the **InputStream** (p. 1464) freeing any resources that might have been acquired during the lifetime of this stream.

The default implementation of this method does nothing.

Exceptions

<i>IOException</i> (p. 1545)	if an I/O error occurs while closing the InputStream (p. 1464).
--	--

Reimplemented from **decaf::io::InputStream** (p. 1466).

6.537.3.3 `virtual int decaf::internal::net::tcp::TcpSocketInputStream::doRead-
ArrayBounded (unsigned char * buffer, int size, int offset, int length)`
[protected, virtual]

Reimplemented from **decaf::io::InputStream** (p. 1467).

6.537.3.4 `virtual int decaf::internal::net::tcp::TcpSocketInputStream::doReadByte ()`
[protected, virtual]

Implements **decaf::io::InputStream** (p. 1467).

6.537.3.5 `virtual long long decaf::internal::net::tcp::TcpSocketInputStream::skip (long long num)`
[virtual]

Not supported.

Skips over and discards n bytes of data from this input stream.

The skip method may, for a variety of reasons, end up skipping over some smaller number of bytes, possibly 0. This may result from any of a number of conditions; reaching end of file before n bytes have been skipped is only one possibility. The actual number of bytes skipped is returned.

The skip method of **InputStream** (p. 1464) creates a byte array and then repeatedly reads into it until num bytes have been read or the end of the stream has been reached. Subclasses are encouraged to provide a more efficient implementation of this method.

Parameters

<i>num</i>	The number of bytes to skip.
------------	------------------------------

Returns

total bytes skipped

Exceptions

<i>IOException</i> (p. 1545)	if an I/O error occurs.
<i>UnsupportedOperationException</i>	if the concrete stream class does not support skipping bytes.

Reimplemented from **decaf::io::InputStream** (p. 1472).

The documentation for this class was generated from the following file:

- src/main/decaf/internal/net/tcp/**TcpSocketInputStream.h**

6.538 decaf::internal::net::tcp::TcpSocketOutputStream Class - Reference

Output stream for performing write operations on a socket.

```
#include <src/main/decaf/internal/net/tcp/TcpSocketOutputStream.h>
```

Inheritance diagram for decaf::internal::net::tcp::TcpSocketOutputStream:

Public Member Functions

- **TcpSocketOutputStream** (**TcpSocket** *socket)
Create a new instance of a Socket OutputStream class.
- virtual **~TcpSocketOutputStream** ()
- virtual void **close** ()
*Closes this object and deallocates the appropriate resources.
The object is generally no longer usable after calling close.*

Exceptions

<i>IOException</i> (p. 1545)	if an error occurs while closing.
-------------------------------------	-----------------------------------

The default implementation of this method does nothing.

Protected Member Functions

- virtual void **doWriteByte** (unsigned char c)
- virtual void **doWriteArrayBounded** (const unsigned char *buffer, int size, int offset, int length)

6.538.1 Detailed Description

Output stream for performing write operations on a socket.

Since

1.0

6.538.2 Constructor & Destructor Documentation

6.538.2.1 **decaf::internal::net::tcp::TcpSocketOutputStream::TcpSocketOutputStream (TcpSocket * socket)**

Create a new instance of a Socket OutputStream class.

Parameters

<i>socket</i>	The socket to use to write out the data.
---------------	--

6.538.2.2 **virtual decaf::internal::net::tcp::TcpSocketOutputStream::~~TcpSocketOutputStream ()** [virtual]

6.538.3 Member Function Documentation

6.538.3.1 **virtual void decaf::internal::net::tcp::TcpSocketOutputStream::close ()** [virtual]

Closes this object and deallocates the appropriate resources.

The object is generally no longer usable after calling close.

Exceptions

<i>IOException</i> (p. 1545)	if an error occurs while closing.
--	-----------------------------------

The default implementation of this method does nothing.

The default implementation of this method does nothing.

Reimplemented from **decaf::io::OutputStream** (p. 2068).

6.538.3.2 virtual void **decaf::internal::net::tcp::TcpSocketOutputStream::doWriteArrayBounded** (const unsigned char * *buffer*, int *size*, int *offset*, int *length*)
[protected, virtual]

Reimplemented from **decaf::io::OutputStream** (p. 2068).

6.538.3.3 virtual void **decaf::internal::net::tcp::TcpSocketOutputStream::doWriteByte** (unsigned char *c*) [protected, virtual]

Implements **decaf::io::OutputStream** (p. 2069).

The documentation for this class was generated from the following file:

- src/main/decaf/internal/net/tcp/TcpSocketOutputStream.h

6.539 activemq::transport::tcp::TcpTransport Class Reference

Implements a TCP/IP based transport filter, this transport is meant to wrap an instance of an **IOTransport** (p. 1548).

```
#include <src/main/activemq/transport/tcp/TcpTransport.h>
```

Inheritance diagram for **activemq::transport::tcp::TcpTransport**:

Public Member Functions

- **TcpTransport** (const **Pointer**< **Transport** > &next)
*Creates a new instance of a **TcpTransport** (p. 2693), the transport is left unconnected and is in a unusable state until the connect method is called.*
- virtual ~**TcpTransport** ()
- void **connect** (const **decaf::net::URI** &uri, const **decaf::util::Properties** &properties)
Creates a Socket and configures it before attempting to connect to the location specified by the URI passed in.
- virtual void **close** ()
Closes this object and deallocates the appropriate resources.
- virtual bool **isFaultTolerant** () const
*Is this **Transport** (p. 2790) fault tolerant, meaning that it will reconnect to a broker on disconnect.*
- virtual bool **isConnected** () const
*Is the **Transport** (p. 2790) Connected to its Broker.*
- virtual bool **isClosed** () const
*Has the **Transport** (p. 2790) been shutdown and no longer usable.*

Protected Member Functions

- virtual **decaf::net::Socket** * **createSocket** ()
Create an unconnected Socket instance to be used by the transport to communicate with the broker.
- virtual void **configureSocket** (**decaf::net::Socket** *socket, const **decaf::util::Properties** &properties)
Using options from configuration URI, configure the socket options before the Socket instance is connected to the Server.

6.539.1 Detailed Description

Implements a TCP/IP based transport filter, this transport is meant to wrap an instance of an **IOTransport** (p. 1548).

The lower level transport should take care of managing stream reads and writes.

6.539.2 Constructor & Destructor Documentation

6.539.2.1 **activemq::transport::tcp::TcpTransport::TcpTransport** (const **Pointer**< **Transport** > &next)

Creates a new instance of a **TcpTransport** (p. 2693), the transport is left unconnected and is in a unusable state until the connect method is called.

Parameters

<i>next</i>	The next transport in the chain
-------------	---------------------------------

6.539.2.2 **virtual activemq::transport::tcp::TcpTransport::~~TcpTransport** ()
 [virtual]

6.539.3 Member Function Documentation

6.539.3.1 **virtual void activemq::transport::tcp::TcpTransport::close** ()
 [virtual]

Closes this object and deallocates the appropriate resources.

The object is generally no longer usable after calling close.

Exceptions

<i>IOException</i>	if an error occurs while closing.
--------------------	-----------------------------------

Reimplemented from **activemq::transport::TransportFilter** (p. 2802).

```
6.539.3.2 virtual void activemq::transport::tcp::TcpTransport::configureSocket (
    decaf::net::Socket * socket, const decaf::util::Properties & properties )
    [protected, virtual]
```

Using options from configuration URI, configure the socket options before the Socket instance is connected to the Server.

Subclasses can override this option to set more configuration options, they should called the base class version to allow the default set of Socket options to also be configured.

Parameters

<i>socket</i>	The Socket instance to configure using options from the given - Properties.
---------------	---

Exceptions

<i>NullPointerException</i>	if the Socket instance is null.
<i>IllegalArgumentException</i>	if the socket instance is not handled by the class.
<i>SocketException</i>	if there is an error while setting one of the Socket options.

```
6.539.3.3 void activemq::transport::tcp::TcpTransport::connect ( const
    decaf::net::URI & uri, const decaf::util::Properties & properties )
```

Creates a Socket and configures it before attempting to connect to the location specified by the URI passed in.

The Socket is configured using parameters in the properties that are passed to this method.

Parameters

<i>uri</i>	The URI that the Transport (p. 2790) is to connect to once initialized.
<i>properties</i>	The Properties that have been parsed from the URI or from configuration files.

```
6.539.3.4 virtual decaf::net::Socket* activemq::transport::tcp-
    ::TcpTransport::createSocket ( ) [protected,
    virtual]
```

Create an unconnected Socket instance to be used by the transport to communicate with the broker.

Returns

a newly created unconnected Socket instance.

6.540 `activemq::transport::tcp::TcpTransportFactory` Class Reference 2703

Exceptions

<code>IOException</code>	if there is an error while creating the unconnected Socket.
--------------------------	---

Reimplemented in `activemq::transport::tcp::SslTransport` (p. 2526).

6.539.3.5 `virtual bool activemq::transport::tcp::TcpTransport::isClosed () const`
`[inline, virtual]`

Has the **Transport** (p. 2790) been shutdown and no longer usable.

Returns

true if the **Transport** (p. 2790)

Reimplemented from `activemq::transport::TransportFilter` (p. 2804).

6.539.3.6 `virtual bool activemq::transport::tcp::TcpTransport::isConnected ()`
`const [inline, virtual]`

Is the **Transport** (p. 2790) Connected to its Broker.

Returns

true if a connection has been made.

Reimplemented from `activemq::transport::TransportFilter` (p. 2804).

6.539.3.7 `virtual bool activemq::transport::tcp::TcpTransport::isFaultTolerant ()`
`const [inline, virtual]`

Is this **Transport** (p. 2790) fault tolerant, meaning that it will reconnect to a broker on disconnect.

Returns

true if the **Transport** (p. 2790) is fault tolerant.

Reimplemented from `activemq::transport::TransportFilter` (p. 2804).

The documentation for this class was generated from the following file:

- `src/main/activemq/transport/tcp/TcpTransport.h`

6.540 `activemq::transport::tcp::TcpTransportFactory` Class - Reference

Factory Responsible for creating the **TcpTransport** (p. 2693).

```
#include <src/main/activemq/transport/tcp/TcpTransport-Factory.h>
```

Inheritance diagram for `activemq::transport::tcp::TcpTransportFactory`:

Public Member Functions

- virtual `~TcpTransportFactory()`
- virtual `Pointer< Transport > create` (const `decaf::net::URI` &location)
*Creates a fully configured **Transport** (p. 2790) instance which could be a chain of filters and transports.*
- virtual `Pointer< Transport > createComposite` (const `decaf::net::URI` &location)
*Creates a slimed down **Transport** (p. 2790) instance which can be used in composite transport instances.*

Protected Member Functions

- virtual `Pointer< Transport > doCreateComposite` (const `decaf::net::URI` &location, const `Pointer< wireformat::WireFormat >` &wireFormat, const `decaf::util::Properties` &properties)

6.540.1 Detailed Description

Factory Responsible for creating the **TcpTransport** (p. 2693).

6.540.2 Constructor & Destructor Documentation

6.540.2.1 `virtual activemq::transport::tcp::TcpTransportFactory::~TcpTransportFactory()` [`inline`, `virtual`]

6.540.3 Member Function Documentation

6.540.3.1 `virtual Pointer<Transport> activemq::transport::tcp::TcpTransportFactory::create` (const `decaf::net::URI` & *location*) [`virtual`]

Creates a fully configured **Transport** (p. 2790) instance which could be a chain of filters and transports.

Parameters

<i>location</i>	- URI location to connect to plus any properties to assign.
-----------------	---

Exceptions

<i>ActiveMQException</i>	if an error occurs
--------------------------	--------------------

Implements **activemq::transport::TransportFactory** (p. 2799).

6.540.3.2 **virtual Pointer<Transport> activemq::transport::tcp::TcpTransportFactory::createComposite (const decaf::net::URI & location)**
[virtual]

Creates a slimed down **Transport** (p. 2790) instance which can be used in composite transport instances.

Parameters

<i>location</i>	- URI location to connect to plus any properties to assign.
-----------------	---

Exceptions

<i>ActiveMQException</i>	if an error occurs
--------------------------	--------------------

Implements **activemq::transport::TransportFactory** (p. 2800).

6.540.3.3 **virtual Pointer<Transport> activemq::transport::tcp::TcpTransportFactory::doCreateComposite (const decaf::net::URI & location, const Pointer< wireformat::WireFormat > & wireFormat, const decaf::util::Properties & properties)** [protected, virtual]

Reimplemented in **activemq::transport::tcp::SslTransportFactory** (p. 2527).

The documentation for this class was generated from the following file:

- src/main/activemq/transport/tcp/**TcpTransportFactory.h**

6.541 cms::TemporaryQueue Class Reference

Defines a Temporary **Queue** (p. 2221) based **Destination** (p. 1210).

```
#include <src/main/cms/TemporaryQueue.h>
```

Inheritance diagram for cms::TemporaryQueue:

Public Member Functions

- virtual **~TemporaryQueue** () throw ()

- virtual std::string **getQueueName** () const =0
Gets the name of this queue.
- virtual void **destroy** ()=0
*Destroy's the Temporary **Destination** (p. 1210) at the Provider.*

6.541.1 Detailed Description

Defines a Temporary **Queue** (p. 2221) based **Destination** (p. 1210).

A **TemporaryQueue** (p. 2698) is a special type of **Queue** (p. 2221) **Destination** (p. 1210) that can only be consumed from the **Connection** (p. 933) which created it. TemporaryQueues are most commonly used as the reply to address for **Message** (p. 1839)'s that implement the request response pattern.

A **TemporaryQueue** (p. 2698) is guaranteed to exist at the Provider only for the lifetime of the **Connection** (p. 933) that created it.

Since

1.0

6.541.2 Constructor & Destructor Documentation

6.541.2.1 virtual cms::TemporaryQueue::~TemporaryQueue () throw ()
[virtual]

6.541.3 Member Function Documentation

6.541.3.1 virtual void cms::TemporaryQueue::destroy () [pure virtual]

Destroy's the Temporary **Destination** (p. 1210) at the Provider.

Exceptions

CMSException (p. 826)	- if an internal error occurs.
---------------------------------	--------------------------------

Implemented in **activemq::commands::ActiveMQTempQueue** (p. 389).

6.541.3.2 virtual std::string cms::TemporaryQueue::getQueueName () const
[pure virtual]

Gets the name of this queue.

Returns

The queue name.

Exceptions

CMSException (p. 826)	- if an internal error occurs.
---------------------------------	--------------------------------

Implemented in **activemq::commands::ActiveMQTempQueue** (p. 391).

The documentation for this class was generated from the following file:

- src/main/cms/**TemporaryQueue.h**

6.542 cms::TemporaryTopic Class Reference

Defines a Temporary **Topic** (p. 2764) based **Destination** (p. 1210).

```
#include <src/main/cms/TemporaryTopic.h>
```

Inheritance diagram for cms::TemporaryTopic:

Public Member Functions

- virtual **~TemporaryTopic** () throw ()
- virtual std::string **getTopicName** () const =0
Gets the name of this topic.
- virtual void **destroy** ()=0
*Destroy's the Temporary **Destination** (p. 1210) at the Provider.*

6.542.1 Detailed Description

Defines a Temporary **Topic** (p. 2764) based **Destination** (p. 1210).

A **TemporaryTopic** (p. 2700) is a special type of **Topic** (p. 2764) **Destination** (p. 1210) that can only be consumed from the **Connection** (p. 933) which created it. Temporary-Topics are most commonly used as the reply to address for **Message** (p. 1839)'s that implement the request response pattern.

A **TemporaryTopic** (p. 2700) is guaranteed to exist at the Provider only for the lifetime of the **Connection** (p. 933) that created it.

Since

1.0

6.542.2 Constructor & Destructor Documentation

6.542.2.1 `virtual cms::TemporaryTopic::~~TemporaryTopic () throw ()`
`[virtual]`

6.542.3 Member Function Documentation

6.542.3.1 `virtual void cms::TemporaryTopic::destroy () [pure virtual]`

Destroy's the Temporary **Destination** (p. 1210) at the Provider.

Exceptions

<i>CMSException</i> (p. 826)
--

Implemented in **activemq::commands::ActiveMQTempTopic** (p. 398).

6.542.3.2 `virtual std::string cms::TemporaryTopic::getTopicName () const` `[pure virtual]`

Gets the name of this topic.

Returns

The topic name.

Exceptions

<i>CMSException</i> (p. 826)	- if an internal error occurs.
--	--------------------------------

Implemented in **activemq::commands::ActiveMQTempTopic** (p. 400).

The documentation for this class was generated from the following file:

- `src/main/cms/TemporaryTopic.h`

6.543 cms::TextMessage Class Reference

Interface for a text message.

```
#include <src/main/cms/TextMessage.h>
```

Inheritance diagram for `cms::TextMessage`:

Public Member Functions

- virtual **~TextMessage** () throw ()
- virtual std::string **getText** () const =0
Gets the message character buffer.
- virtual void **setText** (const char *msg)=0
Sets the message contents, does not take ownership of the passed char, but copies it instead.*
- virtual void **setText** (const std::string &msg)=0
Sets the message contents.

6.543.1 Detailed Description

Interface for a text message.

A **TextMessage** (p.2701) can contain any Text based pay load such as an XML - Document or other Text based document.

Like all Messages, a **TextMessage** (p.2701) is received in Read-Only mode, any attempt to write to the **Message** (p.1839) will result in a MessageNotWritableException being thrown until the `clearBody` method is called which will erase the contents and place the message back in a read / write mode.

Since

1.0

6.543.2 Constructor & Destructor Documentation

6.543.2.1 virtual cms::TextMessage::~TextMessage () throw () [virtual]

6.543.3 Member Function Documentation

6.543.3.1 virtual std::string cms::TextMessage::getText () const [pure virtual]

Gets the message character buffer.

Returns

The message character buffer.

Exceptions

CMSException (p. 826)	- if an internal error occurs.
---------------------------------	--------------------------------

Implemented in **activemq::commands::ActiveMQTextMessage** (p. 408).

6.543.3.2 `virtual void cms::TextMessage::setText (const char * msg)` [pure virtual]

Sets the message contents, does not take ownership of the passed char*, but copies it instead.

Parameters

<i>msg</i>	The message buffer.
------------	---------------------

Exceptions

<i>CMSException</i> (p. 826)	- if an internal error occurs.
<i>MessageNot-WriteableException</i> (p. 1923)	- if the message is in read-only mode..

Implemented in `activemq::commands::ActiveMQTextMessage` (p. 409).

6.543.3.3 `virtual void cms::TextMessage::setText (const std::string & msg)` [pure virtual]

Sets the message contents.

Parameters

<i>msg</i>	The message buffer.
------------	---------------------

Exceptions

<i>CMSException</i> (p. 826)	- if an internal error occurs.
<i>MessageNot-WriteableException</i> (p. 1923)	- if the message is in read-only mode..

Implemented in `activemq::commands::ActiveMQTextMessage` (p. 409).

The documentation for this class was generated from the following file:

- `src/main/cms/TextMessage.h`

6.544 decaf::lang::Thread Class Reference

A **Thread** (p. 2703) is a concurrent unit of execution.

```
#include <src/main/decaf/lang/Thread.h>
```

Inheritance diagram for decaf::lang::Thread:

Data Structures

- class **UncaughtExceptionHandler**

*Interface for handlers invoked when a **Thread** (p. 2703) abruptly terminates due to an uncaught exception.*

Public Types

- enum **State** { **NEW** = 0, **RUNNABLE** = 1, **BLOCKED** = 2, **WAITING** = 3, **TIME-D_WAITING** = 4, **SLEEPING** = 5, **TERMINATED** = 6 }

*Represents the various states that the **Thread** (p. 2703) can be in during its lifetime.*

Public Member Functions

- **Thread** ()
*Constructs a new **Thread** (p. 2703).*
- **Thread** (**Runnable** *task)
*Constructs a new **Thread** (p. 2703) with the given target **Runnable** (p. 2312) task.*
- **Thread** (const std::string &name)
*Constructs a new **Thread** (p. 2703) with the given name.*
- **Thread** (**Runnable** *task, const std::string &name)
*Constructs a new **Thread** (p. 2703) with the given target **Runnable** (p. 2312) task and name.*
- virtual ~**Thread** ()
- virtual void **start** ()
Creates a system thread and starts it in a joinable mode.
- virtual void **join** ()
*Forces the Current **Thread** (p. 2703) to wait until the thread exits.*
- virtual void **join** (long long millisecs)
*Forces the Current **Thread** (p. 2703) to wait until the thread exits.*
- virtual void **join** (long long millisecs, unsigned int nanos)
*Forces the Current **Thread** (p. 2703) to wait until the thread exits.*
- virtual void **run** ()
Default implementation of the run method - does nothing.
- std::string **getName** () const
*Returns the **Thread** (p. 2703)'s assigned name.*
- void **setName** (const std::string &name)
*Sets the name of the **Thread** (p. 2703) to the new Name given by the argument name*
- int **getPriority** () const

*Gets the currently set priority for this **Thread** (p. 2703).*

- void **setPriority** (int value)

*Sets the current **Thread** (p. 2703)'s priority to the newly specified value.*

- void **setDaemon** (bool value)

*Sets if the given **Thread** (p. 2703) is a Daemon **Thread** (p. 2703) or not.*

- bool **isDaemon** () const

Returns whether this thread is a daemon thread or not, if true this thread cannot be joined.

- const **UncaughtExceptionHandler** * **getUncaughtExceptionHandler** () const

Set the handler invoked when this thread abruptly terminates due to an uncaught exception.

- void **setUncaughtExceptionHandler** (**UncaughtExceptionHandler** *handler)

Set the handler invoked when this thread abruptly terminates due to an uncaught exception.

- std::string **toString** () const

*Returns a string that describes the **Thread** (p. 2703).*

- bool **isAlive** () const

*Returns true if the **Thread** (p. 2703) is alive, meaning it has been started and has not yet died.*

- **Thread::State** **getState** () const

*Returns the currently set State of this **Thread** (p. 2703).*

Static Public Member Functions

- static void **sleep** (long long millisecs)

Causes the currently executing thread to halt execution for the specified number of milliseconds, subject to the precision and accuracy of system timers and schedulers.

- static void **sleep** (long long millisecs, unsigned int nanos)

Causes the currently executing thread to halt execution for the specified number of milliseconds plus any additionally specified nanoseconds given, subject to the precision and accuracy of system timers and schedulers.

- static void **yield** ()

Causes the currently executing thread object to temporarily pause and allow other threads to execute.

- static long long **getId** ()

*Obtains the **Thread** (p. 2703) Id of the current thread.*

- static **Thread** * **currentThread** ()

Returns a pointer to the currently executing thread object.

Static Public Attributes

- static const int **MIN_PRIORITY** = 1
The minimum priority that a thread can have.
- static const int **NORM_PRIORITY** = 5
The default priority that a thread is given at create time.
- static const int **MAX_PRIORITY** = 10
The maximum priority that a thread can have.

Friends

- class **decaf::util::concurrent::locks::LockSupport**
- class **decaf::lang::Runtime**

6.544.1 Detailed Description

A **Thread** (p. 2703) is a concurrent unit of execution.

It has its own call stack for methods being invoked, their arguments and local variables. Each process has at least one main **Thread** (p. 2703) running when it is started; typically, there are several others for housekeeping. The application might decide to launch additional Threads for specific purposes.

Threads in the same process interact and synchronize by the use of shared objects and monitors associated with these objects.

There are basically two main ways of having a **Thread** (p. 2703) execute application code. One is providing a new class that extends **Thread** (p. 2703) and overriding its **run()** (p. 2710) method. The other is providing a new **Thread** (p. 2703) instance with a **Runnable** (p. 2312) object during its creation. In both cases, the **start()** (p. 2712) method must be called to actually execute the new **Thread** (p. 2703).

Each **Thread** (p. 2703) has an integer priority that basically determines the amount of CPU time the **Thread** (p. 2703) gets. It can be set using the **setPriority(int)** (p. 2711) method. A **Thread** (p. 2703) can also be made a daemon, which makes it run in the background. The latter also affects VM termination behavior: the VM does not terminate automatically as long as there are non-daemon threads running.

See also

decaf.lang.ThreadGroup (p. 2715)

Since

1.0

6.544.2 Member Enumeration Documentation

6.544.2.1 enum `decaf::lang::Thread::State`

Represents the various states that the **Thread** (p. 2703) can be in during its lifetime.

Enumerator:

NEW Before a **Thread** (p. 2703) is started it exists in this State.

RUNNABLE While a **Thread** (p. 2703) is running and is not blocked it is in this State.

BLOCKED A **Thread** (p. 2703) that is waiting to acquire a lock is in this state.

WAITING A **Thread** (p. 2703) that is waiting for another **Thread** (p. 2703) to perform an action is in this state.

TIMED_WAITING A **Thread** (p. 2703) that is waiting for another **Thread** (p. 2703) to perform an action up to a specified time interval is in this state.

SLEEPING A **Thread** (p. 2703) that is blocked in a Sleep call is in this state.

TERMINATED A **Thread** (p. 2703) whose run method has exited is in this state.

6.544.3 Constructor & Destructor Documentation

6.544.3.1 `decaf::lang::Thread::Thread ()`

Constructs a new **Thread** (p. 2703).

This constructor has the same effect as `Thread(NULL, NULL, GIVEN_NAME)`, where `GIVEN_NAME` is a newly generated name. When no name is given the name is automatically generated and are of the form "Thread-"+n, where n is an integer.

6.544.3.2 `decaf::lang::Thread::Thread (Runnable * task)`

Constructs a new **Thread** (p. 2703) with the given target **Runnable** (p. 2312) task.

This constructor has the same effect as `Thread(NULL, task, GIVEN_NAME)`, where `GIVEN_NAME` is a newly generated name. When no name is given the name is automatically generated and are of the form "Thread-"+n, where n is an integer.

Parameters

<i>task</i>	the Runnable (p. 2312) that this thread manages, if the task is NULL the Thread (p. 2703)'s run method is used instead.
-------------	---

6.544.3.3 `decaf::lang::Thread::Thread (const std::string & name)`

Constructs a new **Thread** (p. 2703) with the given name.

This constructor has the same effect as `Thread(NULL, NULL, GIVEN_NAME)`, where

GIVEN_NAME is a newly generated name. When no name is given the name is automatically generated and are of the form "Thread-"+n, where n is an integer.

Parameters

<i>name</i>	the name to assign to this Thread (p. 2703).
-------------	---

6.544.3.4 decaf::lang::Thread::Thread (Runnable * task, const std::string & name)

Constructs a new **Thread** (p. 2703) with the given target **Runnable** (p. 2312) task and name.

This constructor has the same effect as Thread(NULL, task, GIVEN_NAME), where GIVEN_NAME is a newly generated name. When no name is given the name is automatically generated and are of the form "Thread-"+n, where n is an integer.

Parameters

<i>task</i>	the Runnable (p. 2312) that this thread manages, if the task is NULL the Thread (p. 2703)'s run method is used instead.
<i>name</i>	the name to assign to this Thread (p. 2703).

6.544.3.5 virtual decaf::lang::Thread::~~Thread () [virtual]

6.544.4 Member Function Documentation

6.544.4.1 static Thread* decaf::lang::Thread::currentThread () [static]

Returns a pointer to the currently executing thread object.

Returns

Pointer (p. 2083) to the **Thread** (p. 2703) object representing the currently running **Thread** (p. 2703).

6.544.4.2 static long long decaf::lang::Thread::getId () [static]

Obtains the **Thread** (p. 2703) Id of the current thread.

Returns

Thread (p. 2703) Id

6.544.4.3 std::string decaf::lang::Thread::getName () const

Returns the **Thread** (p. 2703)'s assigned name.

Returns

the Name of the **Thread** (p. 2703).

6.544.4.4 int decaf::lang::Thread::getPriority () const

Gets the currently set priority for this **Thread** (p. 2703).

Returns

an int value representing the **Thread** (p. 2703)'s current priority.

6.544.4.5 Thread::State decaf::lang::Thread::getState () const

Returns the currently set State of this **Thread** (p. 2703).

Returns

the **Thread** (p. 2703)'s current state.

6.544.4.6 const UncaughtExceptionHandler* decaf::lang::Thread::getUncaughtExceptionHandler () const

Set the handler invoked when this thread abruptly terminates due to an uncaught exception.

Returns

a pointer to the set **UncaughtExceptionHandler** (p. 2815).

6.544.4.7 bool decaf::lang::Thread::isAlive () const

Returns true if the **Thread** (p. 2703) is alive, meaning it has been started and has not yet died.

Returns

true if the thread is alive.

6.544.4.8 bool decaf::lang::Thread::isDaemon () const

Returns whether this thread is a daemon thread or not, if true this thread cannot be joined.

Returns

true if the thread is a daemon thread.

6.544.4.9 virtual void **decaf::lang::Thread::join** () [virtual]

Forces the Current **Thread** (p. 2703) to wait until the thread exits.

Exceptions

<i>InterruptedException</i>	if any thread has interrupted the current thread. The interrupted status of the current thread is cleared when this exception is thrown.
-----------------------------	--

6.544.4.10 virtual void **decaf::lang::Thread::join** (long long *millisecs*) [virtual]

Forces the Current **Thread** (p. 2703) to wait until the thread exits.

Parameters

<i>millisecs</i>	the time in Milliseconds before the thread resumes
------------------	--

Exceptions

<i>IllegalArgumentException</i>	if the milliseconds parameter is negative.
<i>InterruptedException</i>	if any thread has interrupted the current thread. The interrupted status of the current thread is cleared when this exception is thrown.

6.544.4.11 virtual void **decaf::lang::Thread::join** (long long *millisecs*, unsigned int *nanos*) [virtual]

Forces the Current **Thread** (p. 2703) to wait until the thread exits.

Parameters

<i>millisecs</i>	the time in Milliseconds before the thread resumes
<i>nanos</i>	0-999999 extra nanoseconds to sleep.

Exceptions

<i>IllegalArgumentException</i>	if the nanoseconds parameter is out of range or the milliseconds paramter is negative.
<i>InterruptedException</i>	if any thread has interrupted the current thread. The interrupted status of the current thread is cleared when this exception is thrown.

6.544.4.12 virtual void **decaf::lang::Thread::run** () [virtual]

Default implementation of the run method - does nothing.

Implements **decaf::lang::Runnable** (p. 2312).

Reimplemented in **activemq::transport::mock::InternalCommandListener** (p. 1528).

6.544.4.13 void decaf::lang::Thread::setDaemon (bool *value*)

Sets if the given **Thread** (p. 2703) is a Daemon **Thread** (p. 2703) or not.

Daemon threads cannot be joined and its resource are automatically reclaimed when it terminates.

Parameters

<i>value</i>	Boolean (p. 545) indicating if this thread should be a daemon thread or not.
--------------	---

Exceptions

<i>IllegalThreadStateException</i>	if the thread is already active.
------------------------------------	----------------------------------

6.544.4.14 void decaf::lang::Thread::setName (const std::string & *name*)

Sets the name of the **Thread** (p. 2703) to the new Name given by the argument *name* name the new name of the **Thread** (p. 2703).

6.544.4.15 void decaf::lang::Thread::setPriority (int *value*)

Sets the current **Thread** (p. 2703)'s priority to the newly specified value.

The given value must be within the range **Thread::MIN_PRIORITY** (p. 2713) and **Thread::MAX_PRIORITY** (p. 2713).

Parameters

<i>value</i>	the new priority value to assign to this Thread (p. 2703).
--------------	---

Exceptions

<i>IllegalArgumentException</i>	if the value is out of range.
---------------------------------	-------------------------------

6.544.4.16 void decaf::lang::Thread::setUncaughtExceptionHandler (**UncaughtExceptionHandler** * *handler*)

Set the handler invoked when this thread abruptly terminates due to an uncaught exception.

Parameters

<i>handler</i>	the <code>UncaughtExceptionHandler</code> to invoke when the Thread (p. 2703) terminates due to an uncaught exception.
----------------	---

6.544.4.17 static void decaf::lang::Thread::sleep (long long *millisecs*) [static]

Causes the currently executing thread to halt execution for the specified number of milliseconds, subject to the precision and accuracy of system timers and schedulers.

Note that this method is a static method that applies to the calling thread and not to the thread object.

Parameters

<i>millisecs</i>	time in milliseconds to halt execution.
------------------	---

Exceptions

<i>IllegalArgumentException</i>	if the milliseconds parameter is negative.
<i>InterruptedException</i>	if the Thread (p. 2703) was interrupted while sleeping.

6.544.4.18 static void decaf::lang::Thread::sleep (long long *millisecs*, unsigned int *nanos*) [static]

Causes the currently executing thread to halt execution for the specified number of milliseconds plus any additionally specified nanoseconds given, subject to the precision and accuracy of system timers and schedulers.

Note that this method is a static method that applies to the calling thread and not to the thread object.

Parameters

<i>millisecs</i>	time in milliseconds to halt execution.
<i>nanos</i>	0-999999 extra nanoseconds to sleep.

Exceptions

<i>IllegalArgumentException</i>	if the nanoseconds parameter is out of range or the milliseconds parameter is negative.
<i>InterruptedException</i>	if the Thread (p. 2703) was interrupted while sleeping.

6.544.4.19 virtual void decaf::lang::Thread::start () [virtual]

Creates a system thread and starts it in a joinable mode.

Upon creation, the **run()** (p. 2710) method of either this object or the provided **Runnable** (p. 2312) object will be invoked in the context of this thread.

Exceptions

<i>IllegalThreadStateException</i>	if the thread has already been started.
<i>RuntimeException</i>	if the Thread (p. 2703) cannot be created for some reason.

6.544.4.20 `std::string decaf::lang::Thread::toString () const`

Returns a string that describes the **Thread** (p. 2703).

Returns

string describing the **Thread** (p. 2703).

6.544.4.21 `static void decaf::lang::Thread::yield () [static]`

Causes the currently executing thread object to temporarily pause and allow other threads to execute.

6.544.5 Friends And Related Function Documentation

6.544.5.1 `friend class decaf::lang::Runtime [friend]`

6.544.5.2 `friend class decaf::util::concurrent::locks::LockSupport [friend]`

6.544.6 Field Documentation

6.544.6.1 `const int decaf::lang::Thread::MAX_PRIORITY = 10 [static]`

The maximum priority that a thread can have.

6.544.6.2 `const int decaf::lang::Thread::MIN_PRIORITY = 1 [static]`

The minimum priority that a thread can have.

6.544.6.3 `const int decaf::lang::Thread::NORM_PRIORITY = 5 [static]`

The default priority that a thread is given at create time.

The documentation for this class was generated from the following file:

- `src/main/decaf/lang/Thread.h`

6.545 decaf::util::concurrent::ThreadFactory Class Reference

public interface **ThreadFactory** (p. 2714)

```
#include <src/main/decaf/util/concurrent/ThreadFactory.h>
```

Public Member Functions

- virtual **~ThreadFactory** ()
- virtual **decaf::lang::Thread * newThread** (**decaf::lang::Runnable *r**)=0
Constructs a new Thread.

6.545.1 Detailed Description

public interface **ThreadFactory** (p. 2714)

An object that creates new threads on demand. Using thread factories removes hard-wiring of calls to new Thread, enabling applications to use special thread subclasses, priorities, etc.

The simplest implementation of this interface is just:

```
class SimpleThreadFactory : public ThreadFactory (p. 2714) { public: Thread* newThread( Runnable* r ) { return new Thread(r); } }
```

The Executors.defaultThreadFactory() method provides a more useful simple implementation, that sets the created thread context to known values before returning it.

Since

1.0

6.545.2 Constructor & Destructor Documentation

6.545.2.1 virtual **decaf::util::concurrent::ThreadFactory::~~ThreadFactory** ()
[inline, virtual]

6.545.3 Member Function Documentation

6.545.3.1 virtual **decaf::lang::Thread* decaf::util::concurrent::ThreadFactory::newThread** (**decaf::lang::Runnable * r**) [pure virtual]

Constructs a new Thread.

Implementations may also initialize priority, name, daemon status, ThreadGroup, etc. The pointer passed is still owned by the caller and is not deleted by the Thread object. The caller owns the returned Thread object and must delete it when finished.

Parameters

<i>r</i>	A pointer to a Runnable instance to be executed by new Thread instance returned.
----------	--

Returns

constructed thread, or NULL if the request to create a thread is rejected the caller owns the returned pointer.

The documentation for this class was generated from the following file:

- `src/main/decaf/util/concurrent/ThreadFactory.h`

6.546 decaf::lang::ThreadGroup Class Reference

```
#include <src/main/decaf/lang/ThreadGroup.h>
```

Public Member Functions

- **ThreadGroup** ()
- virtual **~ThreadGroup** ()

6.546.1 Detailed Description

Since

1.0

6.546.2 Constructor & Destructor Documentation

6.546.2.1 **decaf::lang::ThreadGroup::ThreadGroup** ()

6.546.2.2 **virtual decaf::lang::ThreadGroup::~~ThreadGroup** () `[virtual]`

The documentation for this class was generated from the following file:

- `src/main/decaf/lang/ThreadGroup.h`

6.547 decaf::util::concurrent::ThreadPoolExecutor Class Reference

Defines a Thread Pool object that implements the functionality of pooling threads to perform user tasks.


```
#include <src/main/decaf/util/concurrent/ThreadPoolExecutor.h>
```

Inheritance diagram for decaf::util::concurrent::ThreadPoolExecutor:

Data Structures

- class **AbortPolicy**
*Handler policy for tasks that are rejected upon a call to **ThreadPoolExecutor::execute** (p. 2724) this class always throws a **RejectedExecutionException** (p. 2263).*
- class **CallerRunsPolicy**
*Handler policy for tasks that are rejected upon a call to **ThreadPoolExecutor::execute** (p. 2724) this class will attempt to run the task in the Thread that called the execute method unless the executor is shutdown in which case the task is not run and is destroyed.*

Public Member Functions

- **ThreadPoolExecutor** (int corePoolSize, int maxPoolSize, long long keepAliveTime, const **TimeUnit** &unit, **BlockingQueue**< **decaf::lang::Runnable** * > *workQueue)
*Creates a new instance of a **ThreadPoolExecutor** (p. 2715).*
- **ThreadPoolExecutor** (int corePoolSize, int maxPoolSize, long long keepAliveTime, const **TimeUnit** &unit, **BlockingQueue**< **decaf::lang::Runnable** * > *workQueue, **RejectedExecutionHandler** *handler)
*Creates a new instance of a **ThreadPoolExecutor** (p. 2715).*
- **ThreadPoolExecutor** (int corePoolSize, int maxPoolSize, long long keepAliveTime, const **TimeUnit** &unit, **BlockingQueue**< **decaf::lang::Runnable** * > *workQueue, **ThreadFactory** *threadFactory)
*Creates a new instance of a **ThreadPoolExecutor** (p. 2715).*
- **ThreadPoolExecutor** (int corePoolSize, int maxPoolSize, long long keepAliveTime, const **TimeUnit** &unit, **BlockingQueue**< **decaf::lang::Runnable** * > *workQueue, **ThreadFactory** *threadFactory, **RejectedExecutionHandler** *handler)
*Creates a new instance of a **ThreadPoolExecutor** (p. 2715).*
- virtual ~**ThreadPoolExecutor** ()
- virtual void **execute** (**decaf::lang::Runnable** *task)
Executes the given command at some time in the future.
- virtual void **shutdown** ()
*Performs an orderly shutdown of this **Executor** (p. 1297).*
- virtual **ArrayList** < **decaf::lang::Runnable** * > **shutdownNow** ()
*Attempts to stop all currently executing tasks and returns an **ArrayList** (p. 445) containing the **Runnables** that did not get executed, these object become the property of the caller and are not deleted by this class, they are removed from the work queue and forgotten about.*

- virtual bool **awaitTermination** (long long timeout, const **decaf::util::concurrent-
::TimeUnit** &unit)

The caller will block until the executor has completed termination meaning all tasks that were scheduled before shutdown have now completed and the executor is ready for deletion.
- virtual bool **isShutdown** () const

Returns whether this executor has been shutdown or not.
- virtual bool **isTerminated** () const

Returns whether all tasks have completed after this executor was shut down.
- virtual int **getPoolSize** () const

*Returns the number of threads that currently exists for this **Executor** (p. 1297).*
- virtual int **getCorePoolSize** () const

*Returns the configured number of core threads for this **Executor** (p. 1297).*
- virtual void **setCorePoolSize** (int poolSize)

Set (p. 2397) the number of threads that this executor treats as its core threads, this value will override the value set in the constructor.
- virtual int **getMaximumPoolSize** () const

*Returns the configured maximum number of threads for this **Executor** (p. 1297).*
- virtual void **setMaximumPoolSize** (int maxSize)

*Sets the maximum number of workers this **Executor** (p. 1297) is allowed to have at any given time above the core pool size.*
- virtual long long **getTaskCount** () const

Returns the current number of pending tasks in the work queue.
- virtual int **getActiveCount** () const

Returns an approximation of the number of threads that are currently running tasks for this executor.
- virtual long long **getCompletedTaskCount** () const

*Returns the approximate number of Tasks that have been completed by this **Executor** (p. 1297), this value never decreases.*
- virtual int **getLargestPoolSize** () const

*Returns the most Threads that have ever been active at one time within this **Executors** (p. 1299) Thread pool.*
- virtual **BlockingQueue** < **decaf::lang::Runnable** * > * **getQueue** ()

*Provides access to the Task **Queue** (p. 2222) used by this **Executor** (p. 1297).*
- virtual bool **isTerminating** () const

Returns true if the executor has begin the process of terminating but has not yet completed the process of shutting down all worker threads.
- virtual void **allowCoreThreadTimeout** (bool value)

When true this setting allows the threads in the core pool to terminate if they sit idle longer than the set keep alive time.
- virtual bool **allowsCoreThreadTimeout** () const

Returns whether this executor has been configured to allow core threads to terminate if they sit idle longer than the configured keep alive time.
- virtual long long **getKeepAliveTime** (const **TimeUnit** &unit) const

Returns the currently set value for the maximum amount of time a worker Thread that is not part of the core threads is allowed to sit idle before it terminates.

- virtual void **setKeepAliveTime** (long long timeout, const **TimeUnit** &unit)
Configures the amount of time a non core Thread will remain alive after it has completed its assigned task.
- virtual void **setThreadFactory** (**ThreadFactory** *factory)
*Sets the **ThreadFactory** (p. 2714) instance used to create new Threads for this - **Executor** (p. 1297).*
- virtual **ThreadFactory** * **getThreadFactory** () const
*Gets the currently configured **ThreadFactory** (p. 2714).*
- virtual **RejectedExecutionHandler** * **getRejectedExecutionHandler** () const
*Gets the currently configured **RejectedExecutionHandler** (p. 2266) for this **Executor** (p. 1297).*
- virtual void **setRejectedExecutionHandler** (**RejectedExecutionHandler** *handler)
*Sets the new **RejectedExecutionHandler** (p. 2266) that this executor should use to process any rejected Runnable tasks.*
- virtual bool **prestartCoreThread** ()
By default a Core thread is only created once the first task is queued, this method forces the creation of core thread that waits in an idle mode for new work to be enqueued.
- virtual int **prestartAllCoreThreads** ()
This method will create and start new core threads running in an idle state waiting for new tasks up to the set core thread limit.
- bool **remove** (decaf::lang::Runnable *task)
Attempts to remove the Runnable from the work queue, if successful then the caller now owns the Runnable and is responsible for deleting it.
- virtual void **purge** ()
*Attempts to remove any **Future** (p. 1390) derived tasks from the pending work queue if they have been canceled.*

Protected Member Functions

- virtual void **beforeExecute** (decaf::lang::Thread *thread, decaf::lang::Runnable *task)
Method called before a task is executed by the given thread.
- virtual void **afterExecute** (decaf::lang::Runnable *task, decaf::lang::Throwable *error)
Called upon completion of execution of a given task.
- virtual void **terminated** ()
*Method invoked when the **Executor** (p. 1297) has terminated, by default this method does nothing.*

Friends

- class **ExecutorKernel**

6.547.1 Detailed Description

Defines a Thread Pool object that implements the functionality of pooling threads to perform user tasks.

The Thread Poll has max size that it will grow to. The thread pool allocates threads in blocks. When there are no waiting worker threads and a task is queued then a new batch is allocated. The user can specify the size of the blocks, otherwise a default value is used.

When the user queues a task they must also queue a listener to be notified when the task has completed, this provides the user with a mechanism to know when a task object can be freed.

To have the Thread Pool perform a task, the user enqueue's an object that implements the `Runnable` interface and one of the worker threads will executing it in its thread context.

6.547.2 Constructor & Destructor Documentation

6.547.2.1 `decaf::util::concurrent::ThreadPoolExecutor::ThreadPoolExecutor (int corePoolSize, int maxPoolSize, long long keepAliveTime, const TimeUnit & unit, BlockingQueue< decaf::lang::Runnable * > * workQueue)`

Creates a new instance of a **ThreadPoolExecutor** (p. 2715).

The executor instance is configured with the passed in parameters and a default thread Factory is used along with a default rejected execution handler.

Parameters

<i>corePool-Size</i>	The number of threads to pool regardless of their idle state.
<i>maxPool-Size</i>	The maximum number of threads that will ever exist at one time in the pool.
<i>keepAlive-Time</i>	The maximum time to keep a thread in the pool for if the number of current threads exceeds to core pool size.
<i>unit</i>	The units that the keepAliveTime is specified in.
<i>workQueue</i>	A BlockingQueue (p. 538) implementation that will be used to hold - Runnable tasks that are awaiting execution within this executor. The Executor (p. 1297) takes ownership of the BlockingQueue (p. 538) instance passed once this method returns.

Exceptions

<i>IllegalArgument-Exception</i>	if the corePoolSize or keepAliveTime are negative or the or if maximumPoolSize is less than or equal to zero, or if corePoolSize is greater than maximumPoolSize.
<i>NullPointerException</i>	if the workQueue pointer is NULL.

6.547.2.2 `decaf::util::concurrent::ThreadPoolExecutor::ThreadPoolExecutor (int corePoolSize, int maxPoolSize, long long keepAliveTime, const TimeUnit & unit, BlockingQueue< decaf::lang::Runnable * > * workQueue, RejectedExecutionHandler * handler)`

Creates a new instance of a **ThreadPoolExecutor** (p. 2715).

The executor instance is configured with the passed in parameters and a default thread Factory is used along with a default rejected execution handler.

Parameters

<i>corePool-Size</i>	The number of threads to pool regardless of their idle state.
<i>maxPool-Size</i>	The maximum number of threads that will ever exist at one time in the pool.
<i>keepAlive-Time</i>	The maximum time to keep a thread in the pool for if the number of current threads exceeds to core pool size.
<i>unit</i>	The units that the keepAliveTime is specified in.
<i>workQueue</i>	A BlockingQueue (p. 538) implementation that will be used to hold - Runnable tasks that are awaiting execution within this executor. The Executor (p. 1297) takes ownership of the BlockingQueue (p. 538) instance passed once this method returns.
<i>handler</i>	A RejectedExecutionHandler (p. 2266) implementation that will be used to handle any rejected tasks when they are submitted to this executor. The Executor (p. 1297) takes ownership of the Rejected-ExecutionHandler (p. 2266) instance passed once this method returns.

Exceptions

<i>IllegalArgument-Exception</i>	if the corePoolSize or keepAliveTime are negative or the or if maximumPoolSize is less than or equal to zero, or if corePoolSize is greater than maximumPoolSize.
<i>NullPointerException</i>	if the workQueue pointer is NULL.

6.547.2.3 `decaf::util::concurrent::ThreadPoolExecutor::ThreadPoolExecutor (int corePoolSize, int maxPoolSize, long long keepAliveTime, const TimeUnit & unit, BlockingQueue< decaf::lang::Runnable * > * workQueue, ThreadFactory * threadFactory)`

Creates a new instance of a **ThreadPoolExecutor** (p. 2715).

The executor instance is configured with the passed in parameters and a default thread Factory is used along with a default rejected execution handler.

Parameters

<i>corePool-Size</i>	The number of threads to pool regardless of their idle state.
----------------------	---

<i>maxPool-Size</i>	The maximum number of threads that will ever exist at one time in the pool.
<i>keepAlive-Time</i>	The maximum time to keep a thread in the pool for if the number of current threads exceeds to core pool size.
<i>unit</i>	The units that the keepAliveTime is specified in.
<i>workQueue</i>	A BlockingQueue (p. 538) implementation that will be used to hold - Runnable tasks that are awaiting execution within this executor. The Executor (p. 1297) takes ownership of the BlockingQueue (p. 538) instance passed once this method returns.
<i>thread-Factory</i>	A ThreadFactory (p. 2714) implementation that will be used to create worker threads that are used by this executor to run the submitted tasks. The Executor (p. 1297) takes ownership of the ThreadFactory (p. 2714) instance passed once this method returns.

Exceptions

<i>IllegalArgument-Exception</i>	if the corePoolSize or keepAliveTime are negative or the or if maximumPoolSize is less than or equal to zero, or if corePoolSize is greater than maximumPoolSize.
<i>NullPointerException</i>	if the workQueue pointer is NULL.

6.547.2.4 `decaf::util::concurrent::ThreadPoolExecutor::ThreadPoolExecutor (int corePoolSize, int maxPoolSize, long long keepAliveTime, const TimeUnit & unit, BlockingQueue< decaf::lang::Runnable * > * workQueue, ThreadFactory * threadFactory, RejectedExecutionHandler * handler)`

Creates a new instance of a **ThreadPoolExecutor** (p. 2715).

The executor instance is configured with the passed in parameters and a default thread Factory is used along with a default rejected execution handler.

Parameters

<i>corePool-Size</i>	The number of threads to pool regardless of their idle state.
<i>maxPool-Size</i>	The maximum number of threads that will ever exist at one time in the pool.
<i>keepAlive-Time</i>	The maximum time to keep a thread in the pool for if the number of current threads exceeds to core pool size.
<i>unit</i>	The units that the keepAliveTime is specified in.
<i>workQueue</i>	A BlockingQueue (p. 538) implementation that will be used to hold - Runnable tasks that are awaiting execution within this executor. The Executor (p. 1297) takes ownership of the BlockingQueue (p. 538) instance passed once this method returns.
<i>thread-Factory</i>	A ThreadFactory (p. 2714) implementation that will be used to create worker threads that are used by this executor to run the submitted tasks. The Executor (p. 1297) takes ownership of the ThreadFactory (p. 2714) instance passed once this method returns.

<i>handler</i>	A RejectedExecutionHandler (p. 2266) implementation that will be used to handle any rejected tasks when they are submitted to this executor. The Executor (p. 1297) takes ownership of the BlockingQueue (p. 538) instance passed once this method returns.
----------------	--

Exceptions

<i>IllegalArgumentException</i>	if the <code>corePoolSize</code> or <code>keepAliveTime</code> are negative or the or if <code>maximumPoolSize</code> is less than or equal to zero, or if <code>corePoolSize</code> is greater than <code>maximumPoolSize</code> .
<i>NullPointerException</i>	if the <code>workQueue</code> pointer is NULL.

6.547.2.5 virtual `decaf::util::concurrent::ThreadPoolExecutor::~~ThreadPool-Executor ()` [virtual]

6.547.3 Member Function Documentation

6.547.3.1 virtual void `decaf::util::concurrent::ThreadPoolExecutor::afterExecute (decaf::lang::Runnable * task, decaf::lang::Throwable * error)`
[protected, virtual]

Called upon completion of execution of a given task.

This method is called from the Thread that executed the given Runnable. If the `Throwable` pointer is not NULL then its value is the Exception that caused the task to terminate.

The base class implementation does nothing, a derived class should call this method on its base class to ensure that all subclasses have a chance to process the `afterExecute` event.

Parameters

<i>task</i>	The Runnable instance that was executed by the calling Thread.
<i>error</i>	The exception that was thrown from the given Runnable.

6.547.3.2 virtual void `decaf::util::concurrent::ThreadPool-Executor::allowCoreThreadTimeout (bool value)`
[virtual]

When true this setting allows the threads in the core pool to terminate if they sit idle longer than the set keep alive time.

Core threads that terminate are replaced as needed by new ones on demand. This settings requires that the set keep alive time be greater than zero and will throw an `IllegalArgumentException` if that is not the case.

Parameters

<i>value</i>	Boolean value indicating if core threads are allowed to time out when idle.
--------------	---

Exceptions

<i>IllegalArgumentException</i>	if the keep alive time is set to zero.
---------------------------------	--

6.547.3.3 virtual bool decaf::util::concurrent::ThreadPoolExecutor::allowsCore-ThreadTimeout () const [virtual]

Returns whether this executor has been configured to allow core threads to terminate if they sit idle longer than the configured keep alive time.

Threads that are not core threads continue to time out using the set keep alive value regardless of whether this option is enabled.

Returns

true if core threads can timeout when idle.

6.547.3.4 virtual bool decaf::util::concurrent::ThreadPoolExecutor::await-Termination (long long timeout, const decaf::util::concurrent::TimeUnit & unit) [virtual]

The caller will block until the executor has completed termination meaning all tasks that where scheduled before shutdown have now completed and the executor is ready for deletion.

If the timeout period elapses before the executor reaches the terminated state then this method return false to indicate it has not terminated.

Parameters

<i>timeout</i>	The amount of time to wait before abandoning the wait for termination.
<i>unit</i>	The unit of time that the timeout value represents.

Returns

true if the executor terminated or false if the timeout expired.

Exceptions

<i>InterruptedException</i>	if this call is interrupted while awaiting termination.
-----------------------------	---

Implements **decaf::util::concurrent::ExecutorService** (p. 1303).

6.547.3.5 virtual void decaf::util::concurrent::ThreadPoolExecutor::beforeExecute
 (decaf::lang::Thread * *thread*, decaf::lang::Runnable * *task*)
 [protected, virtual]

Method called before a task is executed by the given thread.

The default implementation of this method does nothing, however a subclass can override this method to add some new functionality.

It is recommended that a subclass call this method on its base class to ensure that all base classes have a chance to process this event.

Parameters

<i>thread</i>	The thread that will be executing the given task.
<i>task</i>	The task that will be executed by the given thread.

6.547.3.6 virtual void decaf::util::concurrent::ThreadPoolExecutor::execute (
 decaf::lang::Runnable * *command*) [virtual]

Executes the given command at some time in the future.

The command may execute in a new thread, in a pooled thread, or in the calling thread, at the discretion of the **Executor** (p. 1297) implementation.

Parameters

<i>command</i>	the runnable task
----------------	-------------------

Exceptions

RejectedExecution- Exception (p. 2263)	if this task cannot be accepted for execution.
<i>NullPointerException</i>	if command is null

Implements **decaf::util::concurrent::Executor** (p. 1299).

Referenced by decaf::util::concurrent::ThreadPoolExecutor::CallerRunsPolicy::Discard-
 OldestPolicy::rejectedExecution().

6.547.3.7 virtual int decaf::util::concurrent::ThreadPoolExecutor::getActiveCount (
) const [virtual]

Returns an approximation of the number of threads that are currently running tasks for this executor.

This value can change rapidly.

Returns

the number of currently active threads.

6.547.3.8 `virtual long long decaf::util::concurrent::ThreadPoolExecutor::getCompletedTaskCount () const`
`[virtual]`

Returns the approximate number of Tasks that have been completed by this **Executor** (p. 1297), this value never decreases.

Returns

the number of completed tasks since creation of the **Executor** (p. 1297).

6.547.3.9 `virtual int decaf::util::concurrent::ThreadPoolExecutor::getCorePoolSize () const` `[virtual]`

Returns the configured number of core threads for this **Executor** (p. 1297).

Returns

the configured number of core Threads.

6.547.3.10 `virtual long long decaf::util::concurrent::ThreadPoolExecutor::getKeepAliveTime (const TimeUnit & unit) const`
`[virtual]`

Returns the currently set value for the maximum amount of time a worker Thread that is not part of the core threads is allowed to sit idle before it terminates.

Parameters

<i>unit</i>	The unit of time to return the results in.
-------------	--

Returns

the configure keep alive time in the requested time units.

6.547.3.11 `virtual int decaf::util::concurrent::ThreadPoolExecutor::getLargestPoolSize () const` `[virtual]`

Returns the most Threads that have ever been active at one time within this **Executors** (p. 1299) Thread pool.

Returns

the largest number of threads ever to coexist in this executor.

6.547.3.12 `virtual int decaf::util::concurrent::ThreadPoolExecutor::getMaximumPoolSize () const [virtual]`

Returns the configured maximum number of threads for this **Executor** (p. 1297).

Returns

the configured maximum number of Threads.

6.547.3.13 `virtual int decaf::util::concurrent::ThreadPoolExecutor::getPoolSize () const [virtual]`

Returns the number of threads that currently exists for this **Executor** (p. 1297).

Returns

the configured number of Threads in the Pool.

6.547.3.14 `virtual BlockingQueue<decaf::lang::Runnable*>* decaf::util::concurrent::ThreadPoolExecutor::getQueue () [virtual]`

Provides access to the Task **Queue** (p. 2222) used by this **Executor** (p. 1297).

This method is meant mainly for debugging and monitoring, care should be taken when using this method. The executor continues to execute tasks from the **Queue** (p. 2222).

Returns

a pointer to the blocking queue that this executor stores future tasks in.

Referenced by decaf::util::concurrent::ThreadPoolExecutor::CallerRunsPolicy::DiscardOldestPolicy::rejectedExecution().

6.547.3.15 `virtual RejectedExecutionHandler* decaf::util::concurrent::ThreadPoolExecutor::getRejectedExecutionHandler () const [virtual]`

Gets the currently configured **RejectedExecutionHandler** (p. 2266) for this **Executor** (p. 1297).

Returns

a pointer to the current **RejectedExecutionHandler** (p. 2266).

6.547.3.16 `virtual long long decaf::util::concurrent::ThreadPoolExecutor::getTaskCount () const [virtual]`

Returns the current number of pending tasks in the work queue.

This is an approximation as the number of pending tasks can quickly changes as tasks complete and new tasks are started.

Returns

number of outstanding tasks, approximate.

6.547.3.17 `virtual ThreadFactory* decaf::util::concurrent::ThreadPoolExecutor::getThreadFactory () const [virtual]`

Gets the currently configured **ThreadFactory** (p. 2714).

It is considered a programming error to delete the pointer returned by this method.

Returns

the currently configured **ThreadFactory** (p. 2714) instance used by this object.

6.547.3.18 `virtual bool decaf::util::concurrent::ThreadPoolExecutor::isShutdown () const [virtual]`

Returns whether this executor has been shutdown or not.

Returns

true if this executor has been shutdown.

Implements **decaf::util::concurrent::ExecutorService** (p. 1304).

Referenced by `decaf::util::concurrent::ThreadPoolExecutor::CallerRunsPolicy::DiscardOldestPolicy::rejectedExecution()`.

6.547.3.19 `virtual bool decaf::util::concurrent::ThreadPoolExecutor::isTerminated () const [virtual]`

Returns whether all tasks have completed after this executor was shut down.

Returns

true if all tasks have completed after a request to shut down was made.

Implements **decaf::util::concurrent::ExecutorService** (p. 1304).

6.547.3.20 `virtual bool decaf::util::concurrent::ThreadPoolExecutor::isTerminating () const [virtual]`

Returns true if the executor has begin the process of terminating but has not yet completed the process of shutting down all worker threads.

If the **Executor** (p. 1297) does not transition from this state to terminated after some time its generally an indication that one of the submitted tasks will not complete and the executor is locked in a terminating state.

Returns

true if all tasks have completed after a request to shut down was made.

6.547.3.21 `virtual int decaf::util::concurrent::ThreadPoolExecutor::prestartAllCoreThreads () [virtual]`

This method will create and start new core threads running in an idle state waiting for new tasks up to the set core thread limit.

When the limit is reached this method returns zero to indicate no more core threads can be created.

Returns

the number of core threads created, or zero if the limit has already been met.

6.547.3.22 `virtual bool decaf::util::concurrent::ThreadPoolExecutor::prestartCoreThread () [virtual]`

By default a Core thread is only created once the first task is queued, this method forces the creation of core thread that waits in an idle mode for new work to be enqueued.

If the limit on core threads has already been reached then this method returns false.

Returns

true if a new core thread was added, false otherwise.

6.547.3.23 `virtual void decaf::util::concurrent::ThreadPoolExecutor::purge () [virtual]`

Attempts to remove any **Future** (p. 1390) derived tasks from the pending work queue if they have been canceled.

This method can be used to more quickly remove and reclaim space as canceled tasks are not run but must await a worker thread to be removed normally. Since there are multiple threads in operation its possible for this method to not remove all canceled tasks from the work queue.

6.547.3.24 **bool** `decaf::util::concurrent::ThreadPoolExecutor::remove (decaf::lang::Runnable * task)`

Attempts to remove the Runnable from the work queue, if successful then the caller now owns the Runnable and is responsible for deleting it.

Parameters

<i>task</i>	The task that is to be removed from the work queue.
-------------	---

Returns

true if the task was removed from the **Queue** (p. 2222).

6.547.3.25 **virtual void** `decaf::util::concurrent::ThreadPoolExecutor::setCorePoolSize (int poolSize)` [virtual]

Set (p. 2397) the number of threads that this executor treats as its core threads, this value will override the value set in the constructor.

If the value given is less than the current value then the core threads will shrink to the new value over time. If the value is larger than the current value then new threads may be started to process currently pending tasks, otherwise they will be started as needed when new tasks arrive.

Parameters

<i>poolSize</i>	The new core pool size for this executor.
-----------------	---

Exceptions

<i>IllegalArgumentException</i>	if the pool size value is less than zero.
---------------------------------	---

6.547.3.26 **virtual void** `decaf::util::concurrent::ThreadPoolExecutor::setKeepAliveTime (long long timeout, const TimeUnit & unit)` [virtual]

Configures the amount of time a non core Thread will remain alive after it has completed its assigned task.

This value can also be applied to core threads if the allowCoreThreadsTimeout option is enabled.

Parameters

<i>timeout</i>	The amount of time an idle worker will live before terminating.
<i>unit</i>	The units that the timeout is given in.

Exceptions

<i>IllegalArgumentException</i>	if allowCoreThreadsTimeout is enabled and the the timeout value given is zero, or the timeout given is negative.
---------------------------------	--

6.547.3.27 virtual void decaf::util::concurrent::ThreadPool-
Executor::setMaximumPoolSize (int *maxSize*)
[virtual]

Sets the maximum number of workers this **Executor** (p. 1297) is allowed to have at any given time above the core pool size.

This new value overrides any set in the constructor and if smaller than the current value worker threads will terminate as they complete their current tasks and become idle.

Parameters

<i>maxSize</i>	The new maximum allowed worker pool size.
----------------	---

Exceptions

<i>IllegalArgumentException</i>	if maxSize is negative or less than core pool size.
---------------------------------	---

6.547.3.28 virtual void decaf::util::concurrent::ThreadPoolExecutor::set-
RejectedExecutionHandler (RejectedExecutionHandler * *handler*)
[virtual]

Sets the new **RejectedExecutionHandler** (p. 2266) that this executor should use to process any rejected Runnable tasks.

This executor takes ownership of the supplied pointer and will desotroy it upon termination, any previous handler is destroyed by this call.

Parameters

<i>handler</i>	The new RejectedExecutionHandler (p. 2266) instance to use.
----------------	--

Exceptions

<i>NullPointerException</i>	if the handler is NULL.
-----------------------------	-------------------------

6.547.3.29 `virtual void decaf::util::concurrent::ThreadPoolExecutor::setThreadFactory (ThreadFactory * factory)`
[virtual]

Sets the **ThreadFactory** (p.2714) instance used to create new Threads for this - **Executor** (p. 1297).

This class takes ownership of the given **ThreadFactory** (p.2714) and will destroy it upon termination or when a new **ThreadFactory** (p.2714) is set using this method.

Parameters

<i>factory</i>	A ThreadFactory (p.2714) instance used by this Executor (p. 1297) to create new Threads.
----------------	--

Exceptions

<i>NullPointerException</i>	if the given factory pointer is NULL.
-----------------------------	---------------------------------------

6.547.3.30 `virtual void decaf::util::concurrent::ThreadPoolExecutor::shutdown ()`
[virtual]

Performs an orderly shutdown of this **Executor** (p. 1297).

Previously queued tasks are allowed to complete but no new tasks are accepted for execution. Calling this method more than once has no affect on this executor.

Implements **decaf::util::concurrent::ExecutorService** (p. 1304).

6.547.3.31 `virtual ArrayList<decaf::lang::Runnable*> decaf::util::concurrent::ThreadPoolExecutor::shutdownNow ()`
[virtual]

Attempts to stop all currently executing tasks and returns an **ArrayList** (p. 445) containing the Runnables that did not get executed, these object become the property of the caller and are not deleted by this class, they are removed from the work queue and forgotten about.

There is no guarantee that this method will halt execution of currently executing tasks.

Returns

an **ArrayList** (p. 445) containing all Runnable instance that were still waiting to be executed by this class, call now owns those pointers.

Implements **decaf::util::concurrent::ExecutorService** (p. 1305).

6.547.3.32 virtual void **decaf::util::concurrent::ThreadPoolExecutor::terminated** ()
[protected, virtual]

Method invoked when the **Executor** (p. 1297) has terminated, by default this method does nothing.

When overridden the subclass should call superclass::terminated to ensure that all subclasses have their terminated method invoked.

6.547.4 Friends And Related Function Documentation

6.547.4.1 friend class **ExecutorKernel** [friend]

The documentation for this class was generated from the following file:

- src/main/decaf/util/concurrent/**ThreadPoolExecutor.h**

6.548 decaf::lang::Throwable Class Reference

This class represents an error that has occurred.

```
#include <src/main/decaf/lang/Throwable.h>
```

Inheritance diagram for decaf::lang::Throwable:

Public Member Functions

- **Throwable** () throw ()
- virtual ~**Throwable** () throw ()
- virtual std::string **getMessage** () const =0
Gets the cause of the error, if no message was provided to the instance of this interface but a cause was then the value cause.getMessage is then returned.
- virtual const std::exception * **getCause** () const =0
Gets the exception that caused this one to be thrown, this allows for chaining of exceptions in the case of a method that throws only a particular exception but wishes to allow for the real causal exception to be passed only in case the caller knows about that type of exception and wishes to respond to it.
- virtual void **initCause** (const std::exception *cause)=0
Initializes the contained cause exception with the one given.
- virtual void **setMark** (const char *file, const int lineNumber)=0
Adds a file/line number to the stack trace.
- virtual **Throwable** * **clone** () const =0
Clones this exception.
- virtual std::vector< std::pair < std::string, int > > **getStackTrace** () const =0

Provides the stack trace for every point where this exception was caught, marked, and rethrown.

- virtual void **printStackTrace** () const =0

Prints the stack trace to std::err.

- virtual void **printStackTrace** (std::ostream &stream) const =0

Prints the stack trace to the given output stream.

- virtual std::string **getStackTraceString** () const =0

Gets the stack trace as one contiguous string.

6.548.1 Detailed Description

This class represents an error that has occurred.

All Exceptions in the Decaf library should extend from this or from the **Exception** (p. 1279) class in order to ensure that all Decaf Exceptions are interchangeable with the std::exception class.

Throwable (p. 2732) can wrap another **Throwable** (p. 2732) as the cause if the error being thrown. The user can inspect the cause by calling `getCause`, the pointer returned is the property of the **Throwable** (p. 2732) instance and will be deleted when it is deleted or goes out of scope.

Since

1.0

6.548.2 Constructor & Destructor Documentation

6.548.2.1 **decaf::lang::Throwable::Throwable** () throw () [inline]

6.548.2.2 **virtual decaf::lang::Throwable::~~Throwable** () throw () [inline, virtual]

6.548.3 Member Function Documentation

6.548.3.1 **virtual Throwable*** **decaf::lang::Throwable::clone** () const [pure virtual]

Clones this exception.

This is useful for cases where you need to preserve the type of the original exception as well as the message. All subclasses should override.

Returns

Copy of this **Exception** (p. 1279) object

Implemented in **decaf::net::URISyntaxException** (p. 2859), **decaf::lang::Exception** (p. 1282), **decaf::internal::net::ssl::openssl::OpenSSLSocketException** (p. 2032), **decaf::lang::exceptions::InvalidStateException** (p. 1545), **decaf::lang::exceptions::IllegalMonitorStateException** (p. 1418), **decaf::lang::exceptions::InterruptedException** (p. 1531), **decaf::security::GeneralSecurityException** (p. 1397), **decaf::security::NoSuchProviderException** (p. 1989), **decaf::security::NoSuchAlgorithmException** (p. 1983), **decaf::security::SignatureException** (p. 2440), **decaf::lang::exceptions::IllegalStateException** (p. 1422), **decaf::lang::exceptions::IllegalThreadStateException** (p. 1425), **decaf::lang::exceptions::NullPointerException** (p. 1991), **decaf::security::InvalidKeyException** (p. 1538), **decaf::lang::exceptions::IllegalArgumentException** (p. 1416), **decaf::util::concurrent::BrokenBarrierException** (p. 558), **decaf::lang::exceptions::RuntimeException** (p. 2317), **decaf::security::KeyException** (p. 1603), **decaf::security::KeyManagementException** (p. 1605), **decaf::util::concurrent::ExecutionException** (p. 1297), **decaf::util::concurrent::TimeoutException** (p. 2739), **decaf::util::NoSuchElementException** (p. 1986), **decaf::lang::exceptions::ClassCastException** (p. 815), **decaf::lang::exceptions::IndexOutOfBoundsException** (p. 1431), **decaf::lang::exceptions::NumberFormatException** (p. 1997), **decaf::util::concurrent::CancellationException** (p. 750), **decaf::util::concurrent::RejectedExecutionException** (p. 2266), **decaf::net::BindException** (p. 534), **decaf::lang::exceptions::UnsupportedOperationException** (p. 2827), **decaf::net::ConnectException** (p. 933), **decaf::nio::InvalidMarkException** (p. 1541), **decaf::io::UTFDataFormatException** (p. 2877), **decaf::io::UnsupportedEncodingException** (p. 2824), **decaf::net::HttpRetryException** (p. 1411), **decaf::net::MalformedURLException** (p. 1768), **decaf::net::NoRouteToHostException** (p. 1981), **decaf::net::PortUnreachableException** (p. 2109), **decaf::net::ProtocolException** (p. 2213), **decaf::net::SocketTimeoutException** (p. 2495), **decaf::net::UnknownHostException** (p. 2818), **decaf::net::UnknownServiceException** (p. 2821), **decaf::util::zip::ZipException** (p. 2955), **decaf::io::EOFException** (p. 1277), **decaf::io::InterruptedIOException** (p. 1533), **decaf::nio::ReadOnlyBufferException** (p. 2246), **decaf::util::ConcurrentModificationException** (p. 904), **decaf::util::zip::DataFormatException** (p. 1078), **decaf::io::IOException** (p. 1547), **decaf::nio::BufferOverflowException** (p. 611), **decaf::nio::BufferUnderflowException** (p. 614), **decaf::net::SocketException** (p. 2473), **decaf::security::cert::CertificateExpiredException** (p. 760), **decaf::security::cert::CertificateNotYetValidException** (p. 762), **decaf::security::cert::CertificateParsingException** (p. 764), **decaf::security::cert::CertificateEncodingException** (p. 756), **decaf::security::cert::CertificateException** (p. 758), **activemq::exceptions::ActiveMQException** (p. 263), **activemq::exceptions::BrokerException** (p. 564), and **activemq::exceptions::ConnectionFailedException** (p. 959).

```
6.548.3.2 virtual const std::exception* decaf::lang::Throwable::getCause ( ) const
    [pure virtual]
```

Gets the exception that caused this one to be thrown, this allows for chaining of exceptions in the case of a method that throws only a particular exception but wishes to allow

for the real causal exception to be passed only in case the caller knows about that type of exception and wishes to respond to it.

Returns

a const pointer reference to the causal exception, if there was no cause associated with this exception then NULL is returned.

Implemented in **decaf::lang::Exception** (p. 1283).

6.548.3.3 `virtual std::string decaf::lang::Throwable::getMessage () const [pure virtual]`

Gets the cause of the error, if no message was provided to the instance of this interface but a cause was then the value `cause.getMessage` is then returned.

Returns

string errors message

Implemented in **decaf::lang::Exception** (p. 1284).

6.548.3.4 `virtual std::vector< std::pair< std::string, int> > decaf::lang::Throwable::getStackTrace () const [pure virtual]`

Provides the stack trace for every point where this exception was caught, marked, and rethrown.

Returns

vector containing stack trace strings

Implemented in **decaf::lang::Exception** (p. 1284).

6.548.3.5 `virtual std::string decaf::lang::Throwable::getStackTraceString () const [pure virtual]`

Gets the stack trace as one contiguous string.

Returns

string with formatted stack trace data

Implemented in **decaf::lang::Exception** (p. 1284).

6.548.3.6 virtual void **decaf::lang::Throwable::initCause** (const std::exception * *cause*)
[pure virtual]

Initializes the contained cause exception with the one given.

A copy is made to avoid ownership issues.

Parameters

<i>cause</i>	The exception that was the cause of this one.
--------------	---

Implemented in **decaf::lang::Exception** (p. 1284).

6.548.3.7 virtual void **decaf::lang::Throwable::printStackTrace** () const [pure virtual]

Prints the stack trace to std::err.

Implemented in **decaf::lang::Exception** (p. 1285).

6.548.3.8 virtual void **decaf::lang::Throwable::printStackTrace** (std::ostream & *stream*) const [pure virtual]

Prints the stack trace to the given output stream.

Parameters

<i>stream</i>	the target output stream.
---------------	---------------------------

Implemented in **decaf::lang::Exception** (p. 1285).

6.548.3.9 virtual void **decaf::lang::Throwable::setMark** (const char * *file*, const int *lineNumber*) [pure virtual]

Adds a file/line number to the stack trace.

Parameters

<i>file</i>	The name of the file calling this method (use __FILE__).
<i>lineNumber</i>	The line number in the calling file (use __LINE__).

Implemented in **decaf::lang::Exception** (p. 1285).

The documentation for this class was generated from the following file:

- src/main/decaf/lang/**Throwable.h**

6.549 decaf::util::concurrent::TimeoutException Class Reference

```
#include <src/main/decaf/util/concurrent/TimeoutException.h>
```

Inheritance diagram for decaf::util::concurrent::TimeoutException:

Public Member Functions

- **TimeoutException** () throw ()
Default Constructor.
- **TimeoutException** (const **decaf::lang::Exception** &ex) throw ()
Conversion Constructor from some other Exception.
- **TimeoutException** (const **TimeoutException** &ex) throw ()
Copy Constructor.
- **TimeoutException** (const std::exception ***cause**) throw ()
Constructor.
- **TimeoutException** (const char *file, const int lineNumber, const char *msg,...) throw ()
Constructor - Initializes the file name and line number where this message occurred.
- **TimeoutException** (const char *file, const int lineNumber, const std::exception ***cause**, const char *msg,...) throw ()
Constructor - Initializes the file name and line number where this message occurred.
- virtual **TimeoutException** * **clone** () const
Clones this exception.
- virtual ~**TimeoutException** () throw ()

6.549.1 Constructor & Destructor Documentation

6.549.1.1 **decaf::util::concurrent::TimeoutException::TimeoutException** () throw () [inline]

Default Constructor.

6.549.1.2 **decaf::util::concurrent::TimeoutException::TimeoutException** (const **decaf::lang::Exception** & ex) throw () [inline]

Conversion Constructor from some other Exception.

Parameters

ex	An exception that should become this type of Exception
-----------	--

6.549.1.3 `decaf::util::concurrent::TimeoutException::TimeoutException (const TimeoutException & ex) throw () [inline]`

Copy Constructor.

Parameters

<i>ex</i>	The exception to copy from.
-----------	-----------------------------

6.549.1.4 `decaf::util::concurrent::TimeoutException::TimeoutException (const std::exception * cause) throw () [inline]`

Constructor.

Parameters

<i>cause</i>	Pointer to the exception that caused this one to be thrown, the object is cloned caller retains ownership.
--------------	--

6.549.1.5 `decaf::util::concurrent::TimeoutException::TimeoutException (const char * file, const int lineNumber, const char * msg, ...) throw () [inline]`

Constructor - Initializes the file name and line number where this message occurred.

Sets the message to report, using an optional list of arguments to parse into the message

Parameters

<i>file</i>	The file name where exception occurs
<i>lineNumber</i>	The line number where the exception occurred.
<i>msg</i>	The string message to report
<i>...</i>	list of primitives that are formatted into the message

6.549.1.6 `decaf::util::concurrent::TimeoutException::TimeoutException (const char * file, const int lineNumber, const std::exception * cause, const char * msg, ...) throw () [inline]`

Constructor - Initializes the file name and line number where this message occurred.

Sets the message to report, using an optional list of arguments to parse into the message

Parameters

<i>file</i>	The file name where exception occurs
<i>lineNumber</i>	The line number where the exception occurred.
<i>cause</i>	The exception that was the cause for this one to be thrown.

<i>msg</i>	The string message to report
...	list of primitives that are formatted into the message

6.549.1.7 `virtual decaf::util::concurrent::TimeoutException::~TimeoutException () throw () [inline, virtual]`

6.549.2 Member Function Documentation

6.549.2.1 `virtual TimeoutException* decaf::util::concurrent-
::TimeoutException::clone () const [inline,
virtual]`

Clones this exception.

This is useful for cases where you need to preserve the type of the original exception as well as the message. All subclasses should override.

Returns

a new **TimeoutException** (p. 2737) that is a copy of this one.

Reimplemented from **decaf::lang::Exception** (p. 1282).

The documentation for this class was generated from the following file:

- `src/main/decaf/util/concurrent/TimeoutException.h`

6.550 decaf::util::Timer Class Reference

A facility for threads to schedule tasks for future execution in a background thread.

```
#include <src/main/decaf/util/Timer.h>
```

Public Member Functions

- **Timer** ()
- **Timer** (const std::string &name)
*Create a new **Timer** (p. 2739) whose associated thread is assigned the name given.*
- virtual **~Timer** ()
- void **cancel** ()
Terminates this timer, discarding any currently scheduled tasks.
- int **purge** ()
Removes all canceled tasks from this timer's task queue.
- void **schedule** (**TimerTask** *task, long long delay)
Schedules the specified task for execution after the specified delay.

- void **schedule** (const **decaf::lang::Pointer**< **TimerTask** > &task, long long delay)
Schedules the specified task for execution after the specified delay.
- void **schedule** (**TimerTask** *task, const **Date** &time)
Schedules the specified task for execution at the specified time.
- void **schedule** (const **decaf::lang::Pointer**< **TimerTask** > &task, const **Date** &time)
Schedules the specified task for execution at the specified time.
- void **schedule** (**TimerTask** *task, long long delay, long long period)
Schedules the specified task for repeated fixed-delay execution, beginning after the specified delay.
- void **schedule** (const **decaf::lang::Pointer**< **TimerTask** > &task, long long delay, long long period)
Schedules the specified task for repeated fixed-delay execution, beginning after the specified delay.
- void **schedule** (**TimerTask** *task, const **Date** &firstTime, long long period)
Schedules the specified task for repeated fixed-delay execution, beginning at the specified time.
- void **schedule** (const **decaf::lang::Pointer**< **TimerTask** > &task, const **Date** &firstTime, long long period)
Schedules the specified task for repeated fixed-delay execution, beginning at the specified time.
- void **scheduleAtFixedRate** (**TimerTask** *task, long long delay, long long period)
Schedules the specified task for repeated fixed-rate execution, beginning after the specified delay.
- void **scheduleAtFixedRate** (const **decaf::lang::Pointer**< **TimerTask** > &task, long long delay, long long period)
Schedules the specified task for repeated fixed-rate execution, beginning after the specified delay.
- void **scheduleAtFixedRate** (**TimerTask** *task, const **Date** &firstTime, long long period)
Schedules the specified task for repeated fixed-rate execution, beginning at the specified time.
- void **scheduleAtFixedRate** (const **decaf::lang::Pointer**< **TimerTask** > &task, const **Date** &firstTime, long long period)
Schedules the specified task for repeated fixed-rate execution, beginning at the specified time.

6.550.1 Detailed Description

A facility for threads to schedule tasks for future execution in a background thread.

Tasks may be scheduled for one-time execution, or for repeated execution at regular intervals.

Corresponding to each **Timer** (p. 2739) object is a single background thread that is used to execute all of the timer's tasks, sequentially. **Timer** (p. 2739) tasks should complete quickly. If a timer task takes excessive time to complete, it "hogs" the timer's task execution thread. This can, in turn, delay the execution of subsequent tasks, which may "bunch up" and execute in rapid succession when (and if) the offending task finally completes.

This class is thread-safe: multiple threads can share a single **Timer** (p. 2739) object without the need for external synchronization.

This class does not offer real-time guarantees: it schedules tasks using the `wait(long)` method.

Since

1.0

6.550.2 Constructor & Destructor Documentation

6.550.2.1 `decaf::util::Timer::Timer ()`

6.550.2.2 `decaf::util::Timer::Timer (const std::string & name)`

Create a new **Timer** (p. 2739) whose associated thread is assigned the name given.

Parameters

<i>name</i>	The name to assign to this Timer (p. 2739)'s Thread.
-------------	---

6.550.2.3 `virtual decaf::util::Timer::~~Timer ()` [virtual]

6.550.3 Member Function Documentation

6.550.3.1 `void decaf::util::Timer::cancel ()`

Terminates this timer, discarding any currently scheduled tasks.

Does not interfere with a currently executing task (if it exists). Once a timer has been terminated, its execution thread terminates gracefully, and no more tasks may be scheduled on it.

Note that calling this method from within the `run` method of a timer task that was invoked by this timer absolutely guarantees that the ongoing task execution is the last task execution that will ever be performed by this timer.

This method may be called repeatedly; the second and subsequent calls have no effect.

6.550.3.2 int decaf::util::Timer::purge ()

Removes all canceled tasks from this timer's task queue.

Calling this method has no effect on the behavior of the timer, but eliminates the canceled tasks from the queue causing the **Timer** (p.2739) to destroy the **TimerTask** (p.2751) pointer it was originally given, the caller should ensure that they no longer have any references to TimerTasks that were previously scheduled.

Most programs will have no need to call this method. It is designed for use by the rare application that cancels a large number of tasks. Calling this method trades time for space: the runtime of the method may be proportional to $n + c \log n$, where n is the number of tasks in the queue and c is the number of canceled tasks.

This method can be called on a **Timer** (p.2739) object that has no scheduled tasks without error.

Returns

the number of tasks removed from the queue.

6.550.3.3 void decaf::util::Timer::schedule (TimerTask * task, long long delay)

Schedules the specified task for execution after the specified delay.

The **TimerTask** (p.2751) pointer is considered to be owned by the **Timer** (p.2739) class once it has been scheduled, the **Timer** (p.2739) will destroy its **TimerTask** (p.2751)'s once they have been canceled or the **Timer** (p.2739) itself is canceled. A **TimerTask** (p.2751) is considered scheduled only when this method return without throwing an exception, until that time ownership is not considered to have been transferred to the **Timer** (p.2739) and the caller should ensure that the **TimerTask** (p.2751) gets deleted if an exception is thrown and no further attempts to schedule that **TimerTask** (p.2751) instance are planned.

Parameters

<i>task</i>	- task to be scheduled.
<i>delay</i>	- delay in milliseconds before task is to be executed.

Exceptions

<i>NullPointerException</i>	- if the TimerTask (p.2751) value is Null.
<i>IllegalArgumentException</i>	- if delay is negative, or delay + System.currentTimeMillis() (p.2672) is negative.
<i>IllegalStateException</i>	- if task was already scheduled or canceled, or timer was canceled.

6.550.3.4 `void decaf::util::Timer::schedule (const decaf::lang::Pointer< TimerTask
> & task, long long delay)`

Schedules the specified task for execution after the specified delay.

Parameters

<i>task</i>	- task to be scheduled.
<i>delay</i>	- delay in milliseconds before task is to be executed.

Exceptions

<i>NullPointerException</i>	- if the TimerTask (p. 2751) value is Null.
<i>IllegalArgumentException</i>	- if delay is negative, or delay + System.currentTimeMillis() (p. 2672) is negative.
<i>IllegalStateException</i>	- if task was already scheduled or canceled, or timer was canceled.

6.550.3.5 `void decaf::util::Timer::schedule (TimerTask * task, const Date & time)`

Schedules the specified task for execution at the specified time.

If the time is in the past, the task is scheduled for immediate execution.

The **TimerTask** (p. 2751) pointer is considered to be owned by the **Timer** (p. 2739) class once it has been scheduled, the **Timer** (p. 2739) will destroy its **TimerTask** (p. 2751)'s once they have been canceled or the **Timer** (p. 2739) itself is canceled. A **TimerTask** (p. 2751) is considered scheduled only when this method return without throwing an exception, until that time ownership is not considered to have been transferred to the **Timer** (p. 2739) and the caller should ensure that the **TimerTask** (p. 2751) gets deleted if an exception is thrown and no further attempts to schedule that **TimerTask** (p. 2751) instance are planned.

Parameters

<i>task</i>	- task to be scheduled.
<i>time</i>	- time at which task is to be executed.

Exceptions

<i>NullPointerException</i>	- if the TimerTask (p. 2751) value is Null.
<i>IllegalArgumentException</i>	- if time.getTime() is negative.
<i>IllegalStateException</i>	- if task was already scheduled or canceled, timer was canceled, or timer thread terminated.

6.550.3.6 void decaf::util::Timer::schedule (const decaf::lang::Pointer< TimerTask > & task, const Date & time)

Schedules the specified task for execution at the specified time.

If the time is in the past, the task is scheduled for immediate execution.

Parameters

<i>task</i>	- task to be scheduled.
<i>time</i>	- time at which task is to be executed.

Exceptions

<i>NullPointerException</i>	- if the TimerTask (p. 2751) value is Null.
<i>IllegalArgumentException</i>	- if time.getTime() is negative.
<i>IllegalStateException</i>	- if task was already scheduled or canceled, timer was canceled, or timer thread terminated.

6.550.3.7 void decaf::util::Timer::schedule (TimerTask * task, long long delay, long long period)

Schedules the specified task for repeated fixed-delay execution, beginning after the specified delay.

Subsequent executions take place at approximately regular intervals separated by the specified period.

The **TimerTask** (p. 2751) pointer is considered to be owned by the **Timer** (p. 2739) class once it has been scheduled, the **Timer** (p. 2739) will destroy its **TimerTask** (p. 2751)'s once they have been canceled or the **Timer** (p. 2739) itself is canceled. A **TimerTask** (p. 2751) is considered scheduled only when this method return without throwing an exception, until that time ownership is not considered to have been transferred to the **Timer** (p. 2739) and the caller should ensure that the **TimerTask** (p. 2751) gets deleted if an exception is thrown and no further attempts to schedule that **TimerTask** (p. 2751) instance are planned.

In fixed-delay execution, each execution is scheduled relative to the actual execution time of the previous execution. If an execution is delayed for any reason (such as other background activity), subsequent executions will be delayed as well. In the long run, the frequency of execution will generally be slightly lower than the reciprocal of the specified period (assuming the system clock underlying Object.wait(long long) is accurate).

Fixed-delay execution is appropriate for recurring activities that require "smoothness." In other words, it is appropriate for activities where it is more important to keep the frequency accurate in the short run than in the long run. This includes most animation tasks, such as blinking a cursor at regular intervals. It also includes tasks wherein regular activity is performed in response to human input, such as automatically repeating a character as long as a key is held down.

Parameters

<i>task</i>	- task to be scheduled.
<i>delay</i>	- delay in milliseconds before task is to be executed.
<i>period</i>	- time in milliseconds between successive task executions.

Exceptions

<i>NullPointerException</i>	- if the TimerTask (p. 2751) value is Null.
<i>IllegalArgumentException</i>	- if delay is negative, or delay + System.currentTimeMillis() (p. 2672) is negative.
<i>IllegalStateException</i>	- if task was already scheduled or canceled, timer was canceled, or timer thread terminated.

6.550.3.8 `void decaf::util::Timer::schedule (const decaf::lang::Pointer< TimerTask > & task, long long delay, long long period)`

Schedules the specified task for repeated fixed-delay execution, beginning after the specified delay.

Subsequent executions take place at approximately regular intervals separated by the specified period.

In fixed-delay execution, each execution is scheduled relative to the actual execution time of the previous execution. If an execution is delayed for any reason (such as other background activity), subsequent executions will be delayed as well. In the long run, the frequency of execution will generally be slightly lower than the reciprocal of the specified period (assuming the system clock underlying `Object.wait(long long)` is accurate).

Fixed-delay execution is appropriate for recurring activities that require "smoothness." In other words, it is appropriate for activities where it is more important to keep the frequency accurate in the short run than in the long run. This includes most animation tasks, such as blinking a cursor at regular intervals. It also includes tasks wherein regular activity is performed in response to human input, such as automatically repeating a character as long as a key is held down.

Parameters

<i>task</i>	- task to be scheduled.
<i>delay</i>	- delay in milliseconds before task is to be executed.
<i>period</i>	- time in milliseconds between successive task executions.

Exceptions

<i>NullPointerException</i>	- if the TimerTask (p. 2751) value is Null.
<i>IllegalArgumentException</i>	- if delay is negative, or delay + System.currentTimeMillis() (p. 2672) is negative.
<i>IllegalStateException</i>	- if task was already scheduled or canceled, timer was canceled, or timer thread terminated.

6.550.3.9 void decaf::util::Timer::schedule (*TimerTask* * *task*, const *Date* & *firstTime*, long long *period*)

Schedules the specified task for repeated fixed-delay execution, beginning at the specified time.

Subsequent executions take place at approximately regular intervals separated by the specified period.

The **TimerTask** (p. 2751) pointer is considered to be owned by the **Timer** (p. 2739) class once it has been scheduled, the **Timer** (p. 2739) will destroy its **TimerTask** (p. 2751)'s once they have been canceled or the **Timer** (p. 2739) itself is canceled. A **TimerTask** (p. 2751) is considered scheduled only when this method return without throwing an exception, until that time ownership is not considered to have been transferred to the **Timer** (p. 2739) and the caller should ensure that the **TimerTask** (p. 2751) gets deleted if an exception is thrown and no further attempts to schedule that **TimerTask** (p. 2751) instance are planned.

In fixed-delay execution, each execution is scheduled relative to the actual execution time of the previous execution. If an execution is delayed for any reason (such as other background activity), subsequent executions will be delayed as well. In the long run, the frequency of execution will generally be slightly lower than the reciprocal of the specified period (assuming the system clock underlying Object.wait(long long) is accurate).

Fixed-delay execution is appropriate for recurring activities that require "smoothness." In other words, it is appropriate for activities where it is more important to keep the frequency accurate in the short run than in the long run. This includes most animation tasks, such as blinking a cursor at regular intervals. It also includes tasks wherein regular activity is performed in response to human input, such as automatically repeating a character as long as a key is held down.

Parameters

<i>task</i>	- task to be scheduled.
<i>firstTime</i>	- First time at which task is to be executed.
<i>period</i>	- time in milliseconds between successive task executions.

Exceptions

<i>NullPointerException</i>	- if the TimerTask (p. 2751) value is Null.
<i>IllegalArgumentException</i>	- if time.getTime() is negative.
<i>IllegalStateException</i>	- if task was already scheduled or canceled, timer was canceled, or timer thread terminated.

6.550.3.10 void decaf::util::Timer::schedule (const decaf::lang::Pointer< *TimerTask* > & *task*, const *Date* & *firstTime*, long long *period*)

Schedules the specified task for repeated fixed-delay execution, beginning at the specified time.

Subsequent executions take place at approximately regular intervals separated by the specified period.

In fixed-delay execution, each execution is scheduled relative to the actual execution time of the previous execution. If an execution is delayed for any reason (such as other background activity), subsequent executions will be delayed as well. In the long run, the frequency of execution will generally be slightly lower than the reciprocal of the specified period (assuming the system clock underlying `Object.wait(long long)` is accurate).

Fixed-delay execution is appropriate for recurring activities that require "smoothness." In other words, it is appropriate for activities where it is more important to keep the frequency accurate in the short run than in the long run. This includes most animation tasks, such as blinking a cursor at regular intervals. It also includes tasks wherein regular activity is performed in response to human input, such as automatically repeating a character as long as a key is held down.

Parameters

<i>task</i>	- task to be scheduled.
<i>firstTime</i>	- First time at which task is to be executed.
<i>period</i>	- time in milliseconds between successive task executions.

Exceptions

<i>NullPointerException</i>	- if the TimerTask (p. 2751) value is Null.
<i>IllegalArgumentException</i>	- if <code>time.getTime()</code> is negative.
<i>IllegalStateException</i>	- if task was already scheduled or canceled, timer was canceled, or timer thread terminated.

6.550.3.11 `void decaf::util::Timer::scheduleAtFixedRate (TimerTask * task, long long delay, long long period)`

Schedules the specified task for repeated fixed-rate execution, beginning after the specified delay.

Subsequent executions take place at approximately regular intervals, separated by the specified period.

The **TimerTask** (p. 2751) pointer is considered to be owned by the **Timer** (p. 2739) class once it has been scheduled, the **Timer** (p. 2739) will destroy its **TimerTask** (p. 2751)'s once they have been canceled or the **Timer** (p. 2739) itself is canceled. A **TimerTask** (p. 2751) is considered scheduled only when this method return without throwing an exception, until that time ownership is not considered to have been transferred to the **Timer** (p. 2739) and the caller should ensure that the **TimerTask** (p. 2751) gets deleted if an exception is thrown and no further attempts to schedule that **TimerTask** (p. 2751) instance are planned.

In fixed-rate execution, each execution is scheduled relative to the scheduled execution time of the initial execution. If an execution is delayed for any reason (such as garbage collection or other background activity), two or more executions will occur in rapid suc-

cession to "catch up." In the long run, the frequency of execution will be exactly the reciprocal of the specified period (assuming the system clock underlying `Object.wait(long)` is accurate).

Fixed-rate execution is appropriate for recurring activities that are sensitive to absolute time, such as ringing a chime every hour on the hour, or running scheduled maintenance every day at a particular time. It is also appropriate for recurring activities where the total time to perform a fixed number of executions is important, such as a count down timer that ticks once every second for ten seconds. Finally, fixed-rate execution is appropriate for scheduling multiple repeating timer tasks that must remain synchronized with respect to one another.

Parameters

<i>task</i>	- task to be scheduled.
<i>delay</i>	- delay in milliseconds before task is to be executed.
<i>period</i>	- time in milliseconds between successive task executions.

Exceptions

<i>NullPointerException</i>	- if the TimerTask (p. 2751) value is Null.
<i>IllegalArgumentException</i>	- if delay is negative, or delay + System.currentTimeMillis() (p. 2672) is negative.
<i>IllegalStateException</i>	- if task was already scheduled or canceled, timer was canceled, or timer thread terminated.

6.550.3.12 void decaf::util::Timer::scheduleAtFixedRate (const decaf::lang::Pointer< TimerTask > & task, long long delay, long long period)

Schedules the specified task for repeated fixed-rate execution, beginning after the specified delay.

Subsequent executions take place at approximately regular intervals, separated by the specified period.

In fixed-rate execution, each execution is scheduled relative to the scheduled execution time of the initial execution. If an execution is delayed for any reason (such as garbage collection or other background activity), two or more executions will occur in rapid succession to "catch up." In the long run, the frequency of execution will be exactly the reciprocal of the specified period (assuming the system clock underlying `Object.wait(long)` is accurate).

Fixed-rate execution is appropriate for recurring activities that are sensitive to absolute time, such as ringing a chime every hour on the hour, or running scheduled maintenance every day at a particular time. It is also appropriate for recurring activities where the total time to perform a fixed number of executions is important, such as a countdown timer that ticks once every second for ten seconds. Finally, fixed-rate execution is appropriate for scheduling multiple repeating timer tasks that must remain synchronized with respect to one another.

Parameters

<i>task</i>	- task to be scheduled.
<i>delay</i>	- delay in milliseconds before task is to be executed.
<i>period</i>	- time in milliseconds between successive task executions.

Exceptions

<i>NullPointerException</i>	- if the TimerTask (p. 2751) value is Null.
<i>IllegalArgumentException</i>	- if delay is negative, or delay + System.currentTimeMillis() (p. 2672) is negative.
<i>IllegalStateException</i>	- if task was already scheduled or canceled, timer was canceled, or timer thread terminated.

6.550.3.13 `void decaf::util::Timer::scheduleAtFixedRate (TimerTask * task, const Date & firstTime, long long period)`

Schedules the specified task for repeated fixed-rate execution, beginning at the specified time.

Subsequent executions take place at approximately regular intervals, separated by the specified period.

The **TimerTask** (p. 2751) pointer is considered to be owned by the **Timer** (p. 2739) class once it has been scheduled, the **Timer** (p. 2739) will destroy its **TimerTask** (p. 2751)'s once they have been canceled or the **Timer** (p. 2739) itself is canceled. A **TimerTask** (p. 2751) is considered scheduled only when this method return without throwing an exception, until that time ownership is not considered to have been transferred to the **Timer** (p. 2739) and the caller should ensure that the **TimerTask** (p. 2751) gets deleted if an exception is thrown and no further attempts to schedule that **TimerTask** (p. 2751) instance are planned.

In fixed-rate execution, each execution is scheduled relative to the scheduled execution time of the initial execution. If an execution is delayed for any reason (such as garbage collection or other background activity), two or more executions will occur in rapid succession to "catch up." In the long run, the frequency of execution will be exactly the reciprocal of the specified period (assuming the system clock underlying `Object.wait(long)` is accurate).

Fixed-rate execution is appropriate for recurring activities that are sensitive to absolute time, such as ringing a chime every hour on the hour, or running scheduled maintenance every day at a particular time. It is also appropriate for recurring activities where the total time to perform a fixed number of executions is important, such as a countdown timer that ticks once every second for ten seconds. Finally, fixed-rate execution is appropriate for scheduling multiple repeating timer tasks that must remain synchronized with respect to one another.

Parameters

<i>task</i>	- task to be scheduled.
<i>firstTime</i>	- First time at which task is to be executed.
<i>period</i>	- time in milliseconds between successive task executions.

Exceptions

<i>NullPointerException</i>	- if the TimerTask (p. 2751) value is Null.
<i>IllegalArgumentException</i>	- if time.getTime() is negative.
<i>IllegalStateException</i>	- if task was already scheduled or canceled, timer was canceled, or timer thread terminated.

6.550.3.14 void decaf::util::Timer::scheduleAtFixedRate (const decaf::lang::Pointer< TimerTask > & task, const Date & firstTime, long long period)

Schedules the specified task for repeated fixed-rate execution, beginning at the specified time.

Subsequent executions take place at approximately regular intervals, separated by the specified period.

In fixed-rate execution, each execution is scheduled relative to the scheduled execution time of the initial execution. If an execution is delayed for any reason (such as garbage collection or other background activity), two or more executions will occur in rapid succession to "catch up." In the long run, the frequency of execution will be exactly the reciprocal of the specified period (assuming the system clock underlying Object.wait(long) is accurate).

Fixed-rate execution is appropriate for recurring activities that are sensitive to absolute time, such as ringing a chime every hour on the hour, or running scheduled maintenance every day at a particular time. It is also appropriate for recurring activities where the total time to perform a fixed number of executions is important, such as a countdown timer that ticks once every second for ten seconds. Finally, fixed-rate execution is appropriate for scheduling multiple repeating timer tasks that must remain synchronized with respect to one another.

Parameters

<i>task</i>	- task to be scheduled.
<i>firstTime</i>	- First time at which task is to be executed.
<i>period</i>	- time in milliseconds between successive task executions.

Exceptions

<i>NullPointerException</i>	- if the TimerTask (p. 2751) value is Null.
<i>IllegalArgumentException</i>	- if time.getTime() is negative.
<i>IllegalStateException</i>	- if task was already scheduled or canceled, timer was canceled, or timer thread terminated.

The documentation for this class was generated from the following file:

- src/main/decaf/util/Timer.h

6.551 decaf::util::TimerTask Class Reference

A Base class for a task object that can be scheduled for one-time or repeated execution by a **Timer** (p. 2739).

```
#include <src/main/decaf/util/TimerTask.h>
```

Inheritance diagram for decaf::util::TimerTask:

Public Member Functions

- **TimerTask** ()
- virtual **~TimerTask** ()
- bool **cancel** ()
Cancels this timer task.
- long long **scheduledExecutionTime** () const
Returns the scheduled execution time of the most recent actual execution of this task.

Protected Member Functions

- bool **isScheduled** () const
- void **setScheduledTime** (long long time)
- long long **getWhen** () const

Friends

- class **Timer**
- class **TimerImpl**
- class **decaf::internal::util::TimerTaskHeap**

6.551.1 Detailed Description

A Base class for a task object that can be scheduled for one-time or repeated execution by a **Timer** (p. 2739).

Since

1.0

6.551.2 Constructor & Destructor Documentation

6.551.2.1 decaf::util::TimerTask::TimerTask ()

6.551.2.2 virtual **decaf::util::TimerTask::~~TimerTask** () [inline, virtual]

6.551.3 Member Function Documentation

6.551.3.1 bool **decaf::util::TimerTask::cancel** ()

Cancels this timer task.

If the task has been scheduled for one-time execution and has not yet run, or has not yet been scheduled, it will never run. If the task has been scheduled for repeated execution, it will never run again. (If the task is running when this call occurs, the task will run to completion, but will never run again.)

Note that calling this method from within the run method of a repeating timer task absolutely guarantees that the timer task will not run again.

This method may be called repeatedly; the second and subsequent calls have no effect.

Returns

true if this task is scheduled for one-time execution and has not yet run, or this task is scheduled for repeated execution. Returns false if the task was scheduled for one-time execution and has already run, or if the task was never scheduled, or if the task was already canceled. (Loosely speaking, this method returns true if it prevents one or more scheduled executions from taking place.)

6.551.3.2 long long **decaf::util::TimerTask::getWhen** () const [protected]

6.551.3.3 bool **decaf::util::TimerTask::isScheduled** () const [protected]

6.551.3.4 long long **decaf::util::TimerTask::scheduledExecutionTime** () const

Returns the scheduled execution time of the most recent actual execution of this task.

(If this method is invoked while task execution is in progress, the return value is the scheduled execution time of the ongoing task execution.)

This method is typically invoked from within a task's run method, to determine whether the current execution of the task is sufficiently timely to warrant performing the scheduled activity:

```
void run() (p. 2312) { if( System::currentTimeMillis() (p. 2672) - scheduledExecutionTime() (p. 2752) >= MAX_TARDINESS) return; // Too late; skip this execution. // - Perform the task }
```

This method is typically not used in conjunction with fixed-delay execution repeating tasks, as their scheduled execution times are allowed to drift over time, and so are not terribly significant.

Returns

the time at which the most recent execution of this task was scheduled to occur, in the format returned by **Date.getTime()** (p. 1141). The return value is undefined if the task has yet to commence its first execution.

6.551.3.5 **void decaf::util::TimerTask::setScheduledTime (long long *time*)**
[protected]

6.551.4 Friends And Related Function Documentation

6.551.4.1 **friend class decaf::internal::util::TimerTaskHeap** [friend]

6.551.4.2 **friend class Timer** [friend]

6.551.4.3 **friend class TimerImpl** [friend]

The documentation for this class was generated from the following file:

- src/main/decaf/util/**TimerTask.h**

6.552 decaf::internal::util::TimerTaskHeap Class Reference

A Binary Heap implemented specifically for the Timer class in Decaf Util.

```
#include <src/main/decaf/internal/util/TimerTaskHeap.h>
```

Public Member Functions

- **TimerTaskHeap ()**
- virtual **~TimerTaskHeap ()**
- **Pointer< TimerTask > peek ()**
Peaks at the Head of the Heap, returns the task with the nearest scheduled run time.
- bool **isEmpty ()** const
- std::size_t **size ()** const
- void **insert** (const **Pointer< TimerTask >** &task)
Inserts the specified Task into the heap, heap is reordered to reflect the addition of a new element.
- void **remove** (std::size_t pos)
Removes the Task at the specified position from the heap, resorts the heap from the position down to the bottom.
- void **reset ()**
Clear all contents from the heap.
- void **adjustMinimum ()**
Resorts the heap starting at the top.

- `std::size_t deletelfCancelled ()`

Runs through the heap removing all cancelled Tasks from it, this is not normally used but in case a a cancellation of a large number of tasks the user can perform this purge.

- `std::size_t find (const Pointer< TimerTask > &task) const`

Searches the heap for the specified TimerTask element and returns its position in the heap.

6.552.1 Detailed Description

A Binary Heap implemented specifically for the Timer class in Decaf Util.

Since

1.0

6.552.2 Constructor & Destructor Documentation

6.552.2.1 `decaf::internal::util::TimerTaskHeap::TimerTaskHeap ()`

6.552.2.2 `virtual decaf::internal::util::TimerTaskHeap::~~TimerTaskHeap ()`
[virtual]

6.552.3 Member Function Documentation

6.552.3.1 `void decaf::internal::util::TimerTaskHeap::adjustMinimum ()`

Resorts the heap starting at the top.

6.552.3.2 `std::size_t decaf::internal::util::TimerTaskHeap::deletelfCancelled ()`

Runs through the heap removing all cancelled Tasks from it, this is not normally used but in case a a cancellation of a large number of tasks the user can perform this purge.

Returns

the number of task that were removed from the heap because they were cancelled.

6.552.3.3 `std::size_t decaf::internal::util::TimerTaskHeap::find (const Pointer< TimerTask > & task) const`

Searches the heap for the specified TimerTask element and returns its position in the heap.

Returns the unsigned equivalent of -1 if the element is not found.

Returns

the position in the Heap where the Task is stored, or npos.

6.552.3.4 `void decaf::internal::util::TimerTaskHeap::insert (const Pointer< TimerTask > & task)`

Inserts the specified Task into the heap, heap is reordered to reflect the addition of a new element.

Parameters

<i>task</i>	The TimerTask to insert into the heap.
-------------	--

6.552.3.5 `bool decaf::internal::util::TimerTaskHeap::isEmpty () const`

Returns

true if the heap is empty.

6.552.3.6 `Pointer<TimerTask> decaf::internal::util::TimerTaskHeap::peek ()`

Peaks at the Head of the Heap, returns the task with the nearest scheduled run time.

Returns

The TimerTask that is scheduled to be executed next if the Heap is empty a Null Pointer value is returned.

6.552.3.7 `void decaf::internal::util::TimerTaskHeap::remove (std::size_t pos)`

Removes the Task at the specified position from the heap, resorts the heap from the position down to the bottom.

Parameters

<i>pos</i>	The position at which to remove the TimerTask and begin a resort of the heap.
------------	---

6.552.3.8 `void decaf::internal::util::TimerTaskHeap::reset ()`

Clear all contents from the heap.

6.552.3.9 `std::size_t decaf::internal::util::TimerTaskHeap::size () const`

Returns

the size of the heap.

The documentation for this class was generated from the following file:

- `src/main/decaf/internal/util/TimerTaskHeap.h`

6.553 decaf::util::concurrent::TimeUnit Class Reference

A **TimeUnit** (p. 2756) represents time durations at a given unit of granularity and provides utility methods to convert across units, and to perform timing and delay operations in these units.

```
#include <src/main/decaf/util/concurrent/TimeUnit.h>
```

Inheritance diagram for `decaf::util::concurrent::TimeUnit`:

Public Member Functions

- virtual `~TimeUnit ()`
- long long **convert** (long long sourceDuration, const **TimeUnit** &sourceUnit) const
Convert the given time duration in the given unit to this unit.
- long long **toNanos** (long long duration) const
Equivalent to `NANOSECONDS.convert (duration, this)`.
- long long **toMicros** (long long duration) const
Equivalent to `MICROSECONDS.convert (duration, this)`.
- long long **toMillis** (long long duration) const
Equivalent to `MILLISECONDS.convert (duration, this)`.
- long long **toSeconds** (long long duration) const
Equivalent to `SECONDS.convert (duration, this)`.
- long long **toMinutes** (long long duration) const
Equivalent to `MINUTES.convert (duration, this)`.
- long long **toHours** (long long duration) const
Equivalent to `HOURS.convert (duration, this)`.
- long long **toDays** (long long duration) const
Equivalent to `DAYS.convert (duration, this)`.
- void **timedWait** (**Synchronizable** *obj, long long timeout) const
Perform a timed `Object.wait` using this time unit.
- void **timedJoin** (**decaf::lang::Thread** *thread, long long timeout)

Perform a timed `Thread.join` using this time unit.

- void **sleep** (long long timeout) const

Perform a `Thread.sleep` using this unit.

- virtual std::string **toString** () const

*Converts the **TimeUnit** (p. 2756) type to the Name of the **TimeUnit** (p. 2756).*

- virtual int **compareTo** (const **TimeUnit** &value) const
- virtual bool **equals** (const **TimeUnit** &value) const
- virtual bool **operator==** (const **TimeUnit** &value) const
- virtual bool **operator<** (const **TimeUnit** &value) const

Static Public Member Functions

- static const **TimeUnit** & **valueOf** (const std::string &name)

*Returns the **TimeUnit** (p. 2756) constant of this type with the specified name.*

Static Public Attributes

- static const **TimeUnit** **NANOSECONDS**

*The Actual **TimeUnit** (p. 2756) enumerations.*

- static const **TimeUnit** **MICROSECONDS**
- static const **TimeUnit** **MILLISECONDS**
- static const **TimeUnit** **SECONDS**
- static const **TimeUnit** **MINUTES**
- static const **TimeUnit** **HOURS**
- static const **TimeUnit** **DAYS**
- static const **TimeUnit** *const **values** []

*The An Array of **TimeUnit** (p. 2756) Instances.*

Protected Member Functions

- **TimeUnit** (int index, const std::string &name)

Hidden Constructor, this class can not be instantiated directly.

6.553.1 Detailed Description

A **TimeUnit** (p. 2756) represents time durations at a given unit of granularity and provides utility methods to convert across units, and to perform timing and delay operations in these units.

A **TimeUnit** (p. 2756) does not maintain time information, but only helps organize and use time representations that may be maintained separately across various contexts. A nanosecond is defined as one thousandth of a microsecond, a microsecond as one thousandth of a millisecond, a millisecond as one thousandth of a second, a minute as sixty seconds, an hour as sixty minutes, and a day as twenty four hours.

A **TimeUnit** (p. 2756) is mainly used to inform time-based methods how a given timing parameter should be interpreted. For example, the following code will timeout in 50 milliseconds if the lock is not available:

Lock (p. 1682) lock = ...; if (lock.tryLock(50, **TimeUnit.MILLISECONDS** (p. 2764))) ...

while this code will timeout in 50 seconds:

Lock (p. 1682) lock = ...; if (lock.tryLock(50, **TimeUnit.SECONDS** (p. 2764))) ...

Note however, that there is no guarantee that a particular timeout implementation will be able to notice the passage of time at the same granularity as the given **TimeUnit** (p. 2756).

6.553.2 Constructor & Destructor Documentation

6.553.2.1 **decaf::util::concurrent::TimeUnit::TimeUnit** (int *index*, const std::string & *name*) [protected]

Hidden Constructor, this class can not be instantiated directly.

Parameters

<i>index</i>	- Index into the Time Unit set.
<i>name</i>	- Name of the unit type being represented.

6.553.2.2 **virtual decaf::util::concurrent::TimeUnit::~~TimeUnit** () [inline, virtual]

6.553.3 Member Function Documentation

6.553.3.1 **virtual int decaf::util::concurrent::TimeUnit::compareTo** (const **TimeUnit** & *value*) const [virtual]

6.553.3.2 **long long decaf::util::concurrent::TimeUnit::convert** (long long *sourceDuration*, const **TimeUnit** & *sourceUnit*) const

Convert the given time duration in the given unit to this unit.

Conversions from finer to coarser granularities truncate, so lose precision. For example converting 999 milliseconds to seconds results in 0. Conversions from coarser to finer granularities with arguments that would numerically overflow saturate to Long.MIN_VALUE if negative or Long.MAX_VALUE if positive.

For example, to convert 10 minutes to milliseconds, use: **TimeUnit.MILLISECONDS**.-convert(10L, **TimeUnit.MINUTES** (p. 2764))

Parameters

<i>source-Duration</i>	- Duration value to convert.
<i>sourceUnit</i>	- Unit type of the source duration.

Returns

the converted duration in this unit, or `Long.MIN_VALUE` if conversion would negatively overflow, or `Long.MAX_VALUE` if it would positively overflow.

6.553.3.3 `virtual bool decaf::util::concurrent::TimeUnit::equals (const TimeUnit & value) const` [virtual]

6.553.3.4 `virtual bool decaf::util::concurrent::TimeUnit::operator< (const TimeUnit & value) const` [virtual]

6.553.3.5 `virtual bool decaf::util::concurrent::TimeUnit::operator== (const TimeUnit & value) const` [virtual]

6.553.3.6 `void decaf::util::concurrent::TimeUnit::sleep (long long timeout) const`

Perform a `Thread.sleep` using this unit.

This is a convenience method that converts time arguments into the form required by the `Thread.sleep` method.

Parameters

<i>timeout</i>	the minimum time to sleep
----------------	---------------------------

See also

`Thread::sleep`

6.553.3.7 `void decaf::util::concurrent::TimeUnit::timedJoin (decaf::lang::Thread * thread, long long timeout)`

Perform a timed `Thread.join` using this time unit.

This is a convenience method that converts time arguments into the form required by the `Thread.join` method.

Parameters

<i>thread</i>	the thread to wait for
<i>timeout</i>	the maximum time to wait

Exceptions

<i>InterruptedException</i>	if interrupted while waiting.
<i>NullPointerException</i>	if the thread object is null.

See also

Thread::join(long long, long long)

6.553.3.8 void decaf::util::concurrent::TimeUnit::timedWait (Synchronizable * *obj*, long long *timeout*) const

Perform a timed `Object.wait` using this time unit.

This is a convenience method that converts timeout arguments into the form required by the `Object.wait` method.

For example, you could implement a blocking `poll` method (see **BlockingQueue.poll** (p. 543)) using:

```
Object poll( long long timeout, const TimeUnit& unit )
    throw( InterruptedException ) {

    while( empty ) {
        unit.timedWait( this, timeout );
        ...
    }
}
```

Parameters

<i>obj</i>	the object to wait on
<i>timeout</i>	the maximum time to wait.

Exceptions

<i>InterruptedException</i>	if interrupted while waiting.
<i>NullPointerException</i>	if the Synchronizable (p. 2639) object is null.

See also

Synchronizable::wait(long long, long long)

6.553.3.9 long long decaf::util::concurrent::TimeUnit::toDays (long long *duration*) const [inline]

Equivalent to `DAYS.convert(duration, this)`.

Parameters

<i>duration</i>	the duration
-----------------	--------------

Returns

the converted duration.

See also

convert (p. 2758)

6.553.3.10 **long long decaf::util::concurrent::TimeUnit::toHours (long long *duration*)**
const [inline]

Equivalent to `HOURS.convert(duration, this)`.

Parameters

<i>duration</i>	the duration
-----------------	--------------

Returns

the converted duration.

See also

convert (p. 2758)

6.553.3.11 **long long decaf::util::concurrent::TimeUnit::toMicros (long long *duration*)**
const [inline]

Equivalent to `MICROSECONDS.convert(duration, this)`.

Parameters

<i>duration</i>	the duration
-----------------	--------------

Returns

the converted duration, or `Long.MIN_VALUE` if conversion would negatively overflow, or `Long.MAX_VALUE` if it would positively overflow.

See also

convert (p. 2758)

6.553.3.12 **long long decaf::util::concurrent::TimeUnit::toMillis (long long *duration*)**
const [inline]

Equivalent to `MILLISECONDS.convert(duration, this)`.

Parameters

<i>duration</i>	the duration
-----------------	--------------

Returns

the converted duration, or `Long.MIN_VALUE` if conversion would negatively overflow, or `Long.MAX_VALUE` if it would positively overflow.

See also

convert (p. 2758)

Referenced by `decaf::util::concurrent::LinkedBlockingQueue< E >::offer()`, and `decaf::util::concurrent::LinkedBlockingQueue< E >::poll()`.

6.553.3.13 `long long decaf::util::concurrent::TimeUnit::toMinutes (long long duration) const [inline]`

Equivalent to `MINUTES.convert (duration, this)`.

Parameters

<i>duration</i>	the duration
-----------------	--------------

Returns

the converted duration.

See also

convert (p. 2758)

6.553.3.14 `long long decaf::util::concurrent::TimeUnit::toNanos (long long duration) const [inline]`

Equivalent to `NANOSECONDS.convert (duration, this)`.

Parameters

<i>duration</i>	the duration
-----------------	--------------

Returns

the converted duration, or `Long.MIN_VALUE` if conversion would negatively overflow, or `Long.MAX_VALUE` if it would positively overflow.

See also

convert (p. 2758)

6.553.3.15 `long long decaf::util::concurrent::TimeUnit::toSeconds (long long duration
) const [inline]`

Equivalent to `SECONDS.convert(duration, this)`.

Parameters

<i>duration</i>	the duration
-----------------	--------------

Returns

the converted duration.

See also

convert (p. 2758)

6.553.3.16 `virtual std::string decaf::util::concurrent::TimeUnit::toString () const
[virtual]`

Converts the **TimeUnit** (p. 2756) type to the Name of the **TimeUnit** (p. 2756).

Returns

String name of the **TimeUnit** (p. 2756)

6.553.3.17 `static const TimeUnit& decaf::util::concurrent::TimeUnit::valueOf (const
std::string & name) [static]`

Returns the **TimeUnit** (p. 2756) constant of this type with the specified name.

The string must match exactly an identifier used to declare an **TimeUnit** (p. 2756) constant in this type. (Extraneous whitespace characters are not permitted.)

Parameters

<i>name</i>	The Name of the TimeUnit (p. 2756) constant to be returned.
-------------	--

Returns

A constant reference to the **TimeUnit** (p. 2756) Constant with the given name.

Exceptions

<i>IllegalArgument-Exception</i>	if this enum type has no constant with the specified name
----------------------------------	---

6.553.4 Field Documentation

6.553.4.1 `const TimeUnit decaf::util::concurrent::TimeUnit::DAYS` [static]

6.553.4.2 `const TimeUnit decaf::util::concurrent::TimeUnit::HOURS` [static]

6.553.4.3 `const TimeUnit decaf::util::concurrent::TimeUnit::MICROSECONDS`
[static]

6.553.4.4 `const TimeUnit decaf::util::concurrent::TimeUnit::MILLISECONDS`
[static]

6.553.4.5 `const TimeUnit decaf::util::concurrent::TimeUnit::MINUTES`
[static]

6.553.4.6 `const TimeUnit decaf::util::concurrent::TimeUnit::NANOSECONDS`
[static]

The Actual **TimeUnit** (p. 2756) enumerations.

6.553.4.7 `const TimeUnit decaf::util::concurrent::TimeUnit::SECONDS`
[static]

6.553.4.8 `const TimeUnit* const decaf::util::concurrent::TimeUnit::values[]`
[static]

The An Array of **TimeUnit** (p. 2756) Instances.

The documentation for this class was generated from the following file:

- `src/main/decaf/util/concurrent/TimeUnit.h`

6.554 cms::Topic Class Reference

An interface encapsulating a provider-specific topic name.

```
#include <src/main/cms/Topic.h>
```

Inheritance diagram for cms::Topic:

Public Member Functions

- virtual `~Topic ()` throw ()
- virtual `std::string getTopicName ()` const =0
Gets the name of this topic.

6.554.1 Detailed Description

An interface encapsulating a provider-specific topic name.

A **Topic** (p. 2764) is a Publish / Subscribe type **Destination** (p. 1210). All Messages sent to a **Topic** (p. 2764) are broadcast to all Subscribers of that **Topic** (p. 2764) unless the Subscriber defines a **Message** (p. 1839) selector that filters out that **Message** (p. 1839).

Since

1.0

6.554.2 Constructor & Destructor Documentation

6.554.2.1 virtual `cms::Topic::~Topic ()` throw () [virtual]

6.554.3 Member Function Documentation

6.554.3.1 virtual `std::string cms::Topic::getTopicName ()` const [pure virtual]

Gets the name of this topic.

Returns

The topic name.

Exceptions

<i>CMSException</i> (p. 826)	- If an internal error occurs.
--	--------------------------------

Implemented in `activemq::commands::ActiveMQTopic` (p. 418).

The documentation for this class was generated from the following file:

- `src/main/cms/Topic.h`

6.555 activemq::state::Tracked Class Reference

```
#include <src/main/activemq/state/Tracked.h>
```

Inheritance diagram for activemq::state::Tracked:

Public Member Functions

- **Tracked** ()
- **Tracked** (const **Pointer**< **decaf::lang::Runnable** > &runnable)
- virtual **~Tracked** ()
- void **onResponse** ()
- bool **isWaitingForResponse** () const

6.555.1 Constructor & Destructor Documentation

6.555.1.1 **activemq::state::Tracked::Tracked** ()

6.555.1.2 **activemq::state::Tracked::Tracked** (const **Pointer**< **decaf::lang::Runnable** > & *runnable*)

6.555.1.3 virtual **activemq::state::Tracked::~~Tracked** () [*inline, virtual*]

6.555.2 Member Function Documentation

6.555.2.1 bool **activemq::state::Tracked::isWaitingForResponse** () const
[*inline*]

6.555.2.2 void **activemq::state::Tracked::onResponse** ()

The documentation for this class was generated from the following file:

- src/main/activemq/state/**Tracked.h**

6.556 activemq::commands::TransactionId Class Reference

```
#include <src/main/activemq/commands/TransactionId.h>
```

Inheritance diagram for activemq::commands::TransactionId:

Public Types

- typedef **decaf::lang::PointerComparator** < **TransactionId** > **COMPARATOR**

Public Member Functions

- **TransactionId** ()
- **TransactionId** (const **TransactionId** &other)
- virtual ~**TransactionId** ()
- virtual unsigned char **getDataStructureType** () const
*Get the **DataStructure** (p. 1133) Type as defined in CommandTypes.h.*
- virtual **TransactionId** * **cloneDataStructure** () const
Clone this object and return a new instance that the caller now owns, this will be an exact copy of this one.
- virtual void **copyDataStructure** (const **DataStructure** *src)
Copy the contents of the passed object into this objects members, overwriting any existing data.
- virtual std::string **toString** () const
*Returns a string containing the information for this **DataStructure** (p. 1133) such as its type and value of its elements.*
- virtual bool **equals** (const **DataStructure** *value) const
*Compares the **DataStructure** (p. 1133) passed in to this one, and returns if they are equivalent.*
- virtual bool **isLocalTransactionId** () const
- virtual bool **isXATransactionId** () const
- virtual int **compareTo** (const **TransactionId** &value) const
- virtual bool **equals** (const **TransactionId** &value) const
- virtual bool **operator==** (const **TransactionId** &value) const
- virtual bool **operator<** (const **TransactionId** &value) const
- **TransactionId** & **operator=** (const **TransactionId** &other)

Static Public Attributes

- static const unsigned char **ID_TRANSACTIONID** = 0

6.556.1 Member Typedef Documentation

- 6.556.1.1 `typedef decaf::lang::PointerComparator<TransactionId>
 activemq::commands::TransactionId::COMPARATOR`

Reimplemented in **activemq::commands::XATransactionId** (p. 2938), and **activemq::commands::LocalTransactionId** (p. 1675).

6.556.2 Constructor & Destructor Documentation

- 6.556.2.1 `activemq::commands::TransactionId::TransactionId ()`
- 6.556.2.2 `activemq::commands::TransactionId::TransactionId (const
 TransactionId & other)`

6.556.2.3 virtual **activemq::commands::TransactionId::~TransactionId** ()
[virtual]

6.556.3 Member Function Documentation

6.556.3.1 virtual **TransactionId*** **activemq::commands::TransactionId::cloneData-Structure** () const [virtual]

Clone this obbject and return a new instance that the caller now owns, this will be an exact copy of this one.

Returns

new copy of this object.

Implements **activemq::commands::DataStructure** (p. 1133).

Reimplemented in **activemq::commands::XATransactionId** (p. 2938), and **activemq::commands::LocalTransactionId** (p. 1676).

6.556.3.2 virtual int **activemq::commands::TransactionId::compareTo** (const **TransactionId** & *value*) const [virtual]

6.556.3.3 virtual void **activemq::commands::TransactionId::copyDataStructure** (const **DataStructure** * *src*) [virtual]

Copy the contents of the passed object into this objects members, overwriting any existing data.

Returns

src - Source Object

Implements **activemq::commands::DataStructure** (p. 1134).

Reimplemented in **activemq::commands::XATransactionId** (p. 2939), and **activemq::commands::LocalTransactionId** (p. 1676).

6.556.3.4 virtual bool **activemq::commands::TransactionId::equals** (const **DataStructure** * *value*) const [virtual]

Compares the **DataStructure** (p. 1133) passed in to this one, and returns if they are equivalent.

Equivalent here means that they are of the same type, and that each element of the objects are the same.

Returns

true if **DataSet** (p. 1133)'s are Equal.

Implements **activemq::commands::DataSet** (p. 1135).

Reimplemented in **activemq::commands::XATransactionId** (p. 2939), and **activemq::commands::LocalTransactionId** (p. 1676).

6.556.3.5 `virtual bool activemq::commands::TransactionId::equals (const TransactionId & value) const [virtual]`

6.556.3.6 `virtual unsigned char activemq::commands::TransactionId::getDataSetType () const [virtual]`

Get the **DataSet** (p. 1133) Type as defined in CommandTypes.h.

Returns

The type of the data structure

Implements **activemq::commands::DataSet** (p. 1137).

Reimplemented in **activemq::commands::XATransactionId** (p. 2940), and **activemq::commands::LocalTransactionId** (p. 1677).

6.556.3.7 `virtual bool activemq::commands::TransactionId::isLocalTransactionId () const [inline, virtual]`

Reimplemented in **activemq::commands::LocalTransactionId** (p. 1677).

6.556.3.8 `virtual bool activemq::commands::TransactionId::isXATransactionId () const [inline, virtual]`

Reimplemented in **activemq::commands::XATransactionId** (p. 2941).

6.556.3.9 `virtual bool activemq::commands::TransactionId::operator< (const TransactionId & value) const [virtual]`

6.556.3.10 `TransactionId& activemq::commands::TransactionId::operator= (const TransactionId & other)`

6.556.3.11 `virtual bool activemq::commands::TransactionId::operator== (const TransactionId & value) const [virtual]`

6.557

activemq::wireformat::openwire::marshal::generated::TransactionIdMarshaller

Class Reference

2777

6.556.3.12 `virtual std::string activemq::commands::TransactionId::toString() const`
[virtual]

Returns a string containing the information for this **DataSet** (p. 1133) such as its type and value of its elements.

Returns

formatted string useful for debugging.

Reimplemented from **activemq::commands::BaseDataSet** (p. 531).

Reimplemented in **activemq::commands::XATransactionId** (p. 2942), and **activemq::commands::LocalTransactionId** (p. 1678).

6.556.4 Field Documentation

6.556.4.1 `const unsigned char activemq::commands::TransactionId::ID_TRANSACTIONID = 0` [static]

The documentation for this class was generated from the following file:

- src/main/activemq/commands/**TransactionId.h**

6.557 **activemq::wireformat::openwire::marshal::generated::TransactionIdMarshaller Class Reference**

Marshaling code for Open Wire Format for **TransactionIdMarshaller** (p. 2770).

```
#include <src/main/activemq/wireformat/openwire/marshal/generated/-  
TransactionIdMarshaller.h>
```

Inheritance diagram for **activemq::wireformat::openwire::marshal::generated::TransactionIdMarshaller**:

Public Member Functions

- **TransactionIdMarshaller** ()
- virtual **~TransactionIdMarshaller** ()
- virtual void **tightUnmarshal** (**OpenWireFormat** *wireFormat, **commands::DataSet** *dataStructure, **decaf::io::DataInputStream** *dataIn, **utils::BooleanStream** *bs)
Tight Un-marhsal to the given stream.
- virtual int **tightMarshal1** (**OpenWireFormat** *wireFormat, **commands::DataSet** *dataStructure, **utils::BooleanStream** *bs)

Tight Marshal to the given stream.

- virtual void **tightMarshal2** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataOutputStream** *dataOut, **utils::BooleanStream** *bs)

Tight Marshal to the given stream.

- virtual void **looseUnmarshal** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataInputStream** *dataIn)

Loose Un-marshal to the given stream.

- virtual void **looseMarshal** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataOutputStream** *dataOut)

Tight Marshal to the given stream.

6.557.1 Detailed Description

Marshaling code for Open Wire Format for **TransactionIdMarshaller** (p. 2770).

NOTE!: This file is auto generated - do not modify! if you need to make a change, please see the Java Classes in the activemq-openwire-generator module

6.557.2 Constructor & Destructor Documentation

6.557.2.1 **activemq::wireformat::openwire::marshal::generated::TransactionIdMarshaller::TransactionIdMarshaller** ()
[inline]

6.557.2.2 **virtual activemq::wireformat::openwire::marshal::generated::TransactionIdMarshaller::~~TransactionIdMarshaller** () [inline, virtual]

6.557.3 Member Function Documentation

6.557.3.1 **virtual void activemq::wireformat::openwire::marshal::generated::TransactionIdMarshaller::looseMarshal** (**OpenWireFormat** * *format*, **commands::DataStructure** * *command*, **decaf::io::DataOutputStream** * *ds*) [virtual]

Tight Marshal to the given stream.

Parameters

<i>format</i>	- The OpenWireFormat properties
<i>command</i>	- the object to Marshal
<i>ds</i>	- DataOutputStream to marshal to

Exceptions

<i>IOException</i>	if an error occurs.
--------------------	---------------------

6.557

activemq::wireformat::openwire::marshal::generated::TransactionIdMarshaller

Class Reference

2779

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1123).

Reimplemented in **activemq::wireformat::openwire::marshal::generated::LocalTransactionIdMarshaller** (p. 1680), and **activemq::wireformat::openwire::marshal::generated::XATransactionIdMarshaller** (p. 2944).

6.557.3.2 **virtual void activemq::wireformat::openwire::marshal::generated::TransactionIdMarshaller::looseUnmarshal (OpenWireFormat * *format*, commands::DataStructure * *command*, decaf::io::DataInputStream * *dis*)**
[virtual]

Loose Un-marhsal to the given stream.

Parameters

<i>format</i>	- The OpenWireFormat properties
<i>command</i>	- the object to Un-Marshal
<i>dis</i>	- the DataInputStream to Un-Marshal from

Exceptions

<i>IOException</i>	if an error occurs.
--------------------	---------------------

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1125).

Reimplemented in **activemq::wireformat::openwire::marshal::generated::LocalTransactionIdMarshaller** (p. 1680), and **activemq::wireformat::openwire::marshal::generated::XATransactionIdMarshaller** (p. 2945).

6.557.3.3 **virtual int activemq::wireformat::openwire::marshal::generated::TransactionIdMarshaller::tightMarshal1 (OpenWireFormat * *format*, commands::DataStructure * *command*, utils::BooleanStream * *bs*)**
[virtual]

Tight Marhsal to the given stream.

Parameters

<i>format</i>	The OpenWireFormat properties
<i>command</i>	The object to Marshal
<i>bs</i>	The boolean stream to marshal to.

Exceptions

<i>IOException</i>	if an error occurs.
--------------------	---------------------

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1127).

Reimplemented in **activemq::wireformat::openwire::marshal::generated::LocalTransactionIdMarshaller** (p. 1681), and **activemq::wireformat::openwire::marshal::generated::XATransactionIdMarshaller** (p. 2945).

6.557.3.4 **virtual void activemq::wireformat::openwire::marshal::generated::TransactionIdMarshaller::tightMarshal2 (OpenWireFormat * *format*, commands::DataStructure * *command*, decaf::io::DataOutputStream * *ds*, utils::BooleanStream * *bs*) [virtual]**

Tight Marshal to the given stream.

Parameters

<i>format</i>	- The OpenWireFormat properties
<i>command</i>	- the object to Marshal
<i>ds</i>	- the DataOutputStream to Marshal to
<i>bs</i>	- boolean stream to marshal to.

Exceptions

<i>IOException</i>	if an error occurs.
--------------------	---------------------

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1129).

Reimplemented in **activemq::wireformat::openwire::marshal::generated::LocalTransactionIdMarshaller** (p. 1681), and **activemq::wireformat::openwire::marshal::generated::XATransactionIdMarshaller** (p. 2945).

6.557.3.5 **virtual void activemq::wireformat::openwire::marshal::generated::TransactionIdMarshaller::tightUnmarshal (OpenWireFormat * *format*, commands::DataStructure * *command*, decaf::io::DataInputStream * *dis*, utils::BooleanStream * *bs*) [virtual]**

Tight Un-marshal to the given stream.

Parameters

<i>format</i>	- The OpenWireFormat properties
<i>command</i>	- the object to Un-Marshal
<i>dis</i>	- the DataInputStream to Un-Marshal from
<i>bs</i>	- boolean stream to unmarshal from.

Exceptions

<i>IOException</i>	if an error occurs.
--------------------	---------------------

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1131).

Reimplemented in **activemq::wireformat::openwire::marshal::generated::LocalTransactionIdMarshaller** (p. 1682), and **activemq::wireformat::openwire::marshal::generated::XATransactionIdMarshaller** (p. 2946).

The documentation for this class was generated from the following file:

- src/main/activemq/wireformat/openwire/marshal/generated/**TransactionIdMarshaller.h**

6.558 activemq::commands::TransactionInfo Class Reference

```
#include <src/main/activemq/commands/TransactionInfo.h>
```

Inheritance diagram for activemq::commands::TransactionInfo:

Public Member Functions

- **TransactionInfo** ()
- virtual **~TransactionInfo** ()
- virtual unsigned char **getDataStructureType** () const
*Get the **DataStructure** (p. 1133) Type as defined in CommandTypes.h.*
- virtual **TransactionInfo** * **cloneDataStructure** () const
Clone this obbject and return a new instance that the caller now owns, this will be an exact copy of this one.
- virtual void **copyDataStructure** (const **DataStructure** *src)
Copy the contents of the passed object into this objects members, overwriting any existing data.
- virtual std::string **toString** () const
*Returns a string containing the information for this **DataStructure** (p. 1133) such as its type and value of its elements.*
- virtual bool **equals** (const **DataStructure** *value) const
*Compares the **DataStructure** (p. 1133) passed in to this one, and returns if they are equivalent.*
- virtual const **Pointer** < **ConnectionId** > & **getConnectionId** () const
- virtual **Pointer** < **ConnectionId** > & **getConnectionId** ()
- virtual void **setConnectionId** (const **Pointer** < **ConnectionId** > &connectionId)
- virtual const **Pointer** < **TransactionId** > & **getTransactionId** () const
- virtual **Pointer** < **TransactionId** > & **getTransactionId** ()

- virtual void **setTransactionId** (const **Pointer**< **TransactionId** > &transactionId)
- virtual unsigned char **getType** () const
- virtual void **setType** (unsigned char type)
- virtual bool **isTransactionInfo** () const
- virtual **Pointer**< **Command** > **visit** (activemq::state::CommandVisitor *visitor)

Allows a Visitor to visit this command and return a response to the command based on the command type being visited.

Static Public Attributes

- static const unsigned char **ID_TRANSACTIONINFO** = 7

Protected Attributes

- **Pointer**< **ConnectionId** > **connectionId**
- **Pointer**< **TransactionId** > **transactionId**
- unsigned char **type**

6.558.1 Constructor & Destructor Documentation

6.558.1.1 **activemq::commands::TransactionInfo::TransactionInfo** ()

6.558.1.2 **virtual activemq::commands::TransactionInfo::~~TransactionInfo** ()
[virtual]

6.558.2 Member Function Documentation

6.558.2.1 **virtual TransactionInfo* activemq::commands::TransactionInfo::clone-DataStructure** () const [virtual]

Clone this obbject and return a new instance that the caller now owns, this will be an exact copy of this one.

Returns

new copy of this object.

Implements **activemq::commands::DataStructure** (p. 1133).

6.558.2.2 **virtual void activemq::commands::TransactionInfo::copyDataStructure** (const **DataStructure** * *src*) [virtual]

Copy the contents of the passed object into this objects members, overwriting any existing data.

Returns

src - Source Object

Reimplemented from **activemq::commands::BaseCommand** (p. 493).

6.558.2.3 `virtual bool activemq::commands::TransactionInfo::equals (const DataStructure * value) const [virtual]`

Compares the **DataStructure** (p. 1133) passed in to this one, and returns if they are equivalent.

Equivalent here means that they are of the same type, and that each element of the objects are the same.

Returns

true if **DataStructure** (p. 1133)'s are Equal.

Reimplemented from **activemq::commands::BaseCommand** (p. 494).

6.558.2.4 `virtual const Pointer<ConnectionId>& activemq::commands::TransactionInfo::getConnectionId () const [virtual]`

6.558.2.5 `virtual Pointer<ConnectionId>& activemq::commands::TransactionInfo::getConnectionId () [virtual]`

6.558.2.6 `virtual unsigned char activemq::commands::TransactionInfo::getDataType () const [virtual]`

Get the **DataStructure** (p. 1133) Type as defined in CommandTypes.h.

Returns

The type of the data structure

Implements **activemq::commands::DataStructure** (p. 1137).

6.558.2.7 `virtual const Pointer<TransactionId>& activemq::commands::TransactionInfo::getTransactionId () const [virtual]`

6.558.2.8 `virtual Pointer<TransactionId>& activemq::commands::TransactionInfo::getTransactionId () [virtual]`

6.558.2.9 `virtual unsigned char activemq::commands::TransactionInfo::getType () const [virtual]`

Referenced by **activemq::state::CommandVisitorAdapter::processTransactionInfo()**.

6.558.2.10 `virtual bool activemq::commands::TransactionInfo::isTransactionInfo () const [inline, virtual]`

Returns

an answer of true to the **isTransactionInfo()** (p. 2777) query.

Reimplemented from **activemq::commands::BaseCommand** (p. 497).

6.558.2.11 `virtual void activemq::commands::TransactionInfo::setConnectionId (const Pointer< ConnectionId > & connectionId) [virtual]`

6.558.2.12 `virtual void activemq::commands::TransactionInfo::setTransactionId (const Pointer< TransactionId > & transactionId) [virtual]`

6.558.2.13 `virtual void activemq::commands::TransactionInfo::setType (unsigned char type) [virtual]`

6.558.2.14 `virtual std::string activemq::commands::TransactionInfo::toString () const [virtual]`

Returns a string containing the information for this **DataStructure** (p. 1133) such as its type and value of its elements.

Returns

formatted string useful for debugging.

Reimplemented from **activemq::commands::BaseCommand** (p. 498).

6.558.2.15 `virtual Pointer< Command > activemq::commands::Transaction-Info::visit (activemq::state::CommandVisitor * visitor) [virtual]`

Allows a Visitor to visit this command and return a response to the command based on the command type being visited.

The command will call the proper processXXX method in the visitor.

Returns

a **Response** (p. 2298) to the visitor being called or NULL if no response.

Implements **activemq::commands::Command** (p. 871).

6.558.3 Field Documentation

6.559

activemq::wireformat::openwire::marshal::generated::TransactionInfoMarshaller Class Reference 2785

- 6.558.3.1 `Pointer<ConnectionId> activemq::commands::TransactionInfo::connectionId` [protected]
- 6.558.3.2 `const unsigned char activemq::commands::TransactionInfo::ID_TRANSACTIONINFO = 7` [static]
- 6.558.3.3 `Pointer<TransactionId> activemq::commands::TransactionInfo::transactionId` [protected]
- 6.558.3.4 `unsigned char activemq::commands::TransactionInfo::type` [protected]

The documentation for this class was generated from the following file:

- `src/main/activemq/commands/TransactionInfo.h`

6.559 activemq::wireformat::openwire::marshal::generated::TransactionInfoMarshaller Class Reference

Marshaling code for Open Wire Format for **TransactionInfoMarshaller** (p. 2778).

```
#include <src/main/activemq/wireformat/openwire/marshal/generated/-  
TransactionInfoMarshaller.h>
```

Inheritance diagram for `activemq::wireformat::openwire::marshal::generated::TransactionInfoMarshaller`:

Public Member Functions

- **TransactionInfoMarshaller** ()
- virtual `~TransactionInfoMarshaller` ()
- virtual `commands::DataStructure * createObject ()` const
Creates a new instance of the class that this class is a marshaling director for.
- virtual unsigned char `getDataStructureType ()` const
Gets the DataStructureType that this class marshals/unmarshals.
- virtual void `tightUnmarshal (OpenWireFormat *wireFormat, commands::DataStructure *dataStructure, decaf::io::DataInputStream *dataIn, utils::BooleanStream *bs)`
Tight Un-marhsal to the given stream.
- virtual int `tightMarshal1 (OpenWireFormat *wireFormat, commands::DataStructure *dataStructure, utils::BooleanStream *bs)`
Tight Marshal to the given stream.
- virtual void `tightMarshal2 (OpenWireFormat *wireFormat, commands::DataStructure *dataStructure, decaf::io::DataOutputStream *dataOut, utils::BooleanStream *bs)`

Tight Marshal to the given stream.

- virtual void **looseUnmarshal** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataInputStream** *dataIn)

Loose Un-marshal to the given stream.

- virtual void **looseMarshal** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataOutputStream** *dataOut)

Tight Marshal to the given stream.

6.559.1 Detailed Description

Marshaling code for Open Wire Format for **TransactionInfoMarshaller** (p. 2778).

NOTE!: This file is auto generated - do not modify! if you need to make a change, please see the Java Classes in the activemq-openwire-generator module

6.559.2 Constructor & Destructor Documentation

6.559.2.1 **activemq::wireformat::openwire::marshal::generated::TransactionInfoMarshaller::TransactionInfoMarshaller** ()
[inline]

6.559.2.2 **virtual activemq::wireformat::openwire::marshal::generated::TransactionInfoMarshaller::~~TransactionInfoMarshaller** ()
[inline, virtual]

6.559.3 Member Function Documentation

6.559.3.1 **virtual commands::DataStructure* activemq::wireformat::openwire::marshal::generated::TransactionInfoMarshaller::createObject** () const
[virtual]

Creates a new instance of the class that this class is a marshaling director for.

Returns

newly allocated Command

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1120).

6.559.3.2 **virtual unsigned char activemq::wireformat::openwire::marshal::generated::TransactionInfoMarshaller::getDataStructureType** () const
[virtual]

Gets the DataStructureType that this class marshals/unmarshals.

6.559

activemq::wireformat::openwire::marshal::generated::TransactionInfoMarshaller
Class Reference **2787**
~~Returns~~

byte Id of this classes DataStructureType

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller**
(p. 1122).

6.559.3.3 **virtual void activemq::wireformat::openwire::marshal::generated::-**
TransactionInfoMarshaller::looseMarshal (OpenWireFormat * *format*,
commands::DataStructure * *command*, decaf::io::DataOutputStream * *ds*
) [virtual]

Tight Marhsal to the given stream.

Parameters

<i>format</i>	- The OpenWireFormat properties
<i>command</i>	- the object to Marshal
<i>ds</i>	- DataOutputStream to marshal to

Exceptions

<i>IOException</i>	if an error occurs.
--------------------	---------------------

Reimplemented from **activemq::wireformat::openwire::marshal::generated::Base-**
CommandMarshaller (p. 500).

6.559.3.4 **virtual void activemq::wireformat::openwire::marshal::generated::-**
TransactionInfoMarshaller::looseUnmarshal (OpenWireFormat * *format*,
commands::DataStructure * *command*, decaf::io::DataInputStream * *dis*)
[virtual]

Loose Un-marhsal to the given stream.

Parameters

<i>format</i>	- The OpenWireFormat properties
<i>command</i>	- the object to Un-Marshal
<i>dis</i>	- the DataInputStream to Un-Marshal from

Exceptions

<i>IOException</i>	if an error occurs.
--------------------	---------------------

Reimplemented from **activemq::wireformat::openwire::marshal::generated::Base-**
CommandMarshaller (p. 501).

6.559.3.5 `virtual int activemq::wireformat::openwire::marshal::generated::-`
TransactionInfoMarshaller::tightMarshal1 (`OpenWireFormat * format`,
`commands::DataStructure * command`, `utils::BooleanStream * bs`)
`[virtual]`

Tight Marhsal to the given stream.

Parameters

<i>format</i>	The OpenwireFormat properties
<i>command</i>	The object to Marshal
<i>bs</i>	The boolean stream to marshal to.

Exceptions

<i>IOException</i>	if an error occurs.
--------------------	---------------------

Reimplemented from **activemq::wireformat::openwire::marshal::generated::Base-CommandMarshaller** (p. 503).

6.559.3.6 `virtual void activemq::wireformat::openwire::marshal::generated::-`
TransactionInfoMarshaller::tightMarshal2 (`OpenWireFormat * format`,
`commands::DataStructure * command`, `decaf::io::DataOutputStream * ds`,
`utils::BooleanStream * bs`) `[virtual]`

Tight Marhsal to the given stream.

Parameters

<i>format</i>	- The OpenwireFormat properties
<i>command</i>	- the object to Marshal
<i>ds</i>	- the DataOutputStream to Marshal to
<i>bs</i>	- boolean stream to marshal to.

Exceptions

<i>IOException</i>	if an error occurs.
--------------------	---------------------

Reimplemented from **activemq::wireformat::openwire::marshal::generated::Base-CommandMarshaller** (p. 504).

6.559.3.7 `virtual void activemq::wireformat::openwire::marshal::generated::-`
TransactionInfoMarshaller::tightUnmarshal (`OpenWireFormat * format`,
`commands::DataStructure * command`, `decaf::io::DataInputStream * dis`,
`utils::BooleanStream * bs`) `[virtual]`

Tight Un-marhsal to the given stream.

Parameters

<i>format</i>	- The OpenwireFormat properties
<i>command</i>	- the object to Un-Marshal
<i>dis</i>	- the DataInputStream to Un-Marshal from
<i>bs</i>	- boolean stream to unmarshal from.

Exceptions

<i>IOException</i>	if an error occurs.
--------------------	---------------------

Reimplemented from **activemq::wireformat::openwire::marshal::generated::Base-CommandMarshaller** (p. 505).

The documentation for this class was generated from the following file:

- src/main/activemq/wireformat/openwire/marshal/generated/**TransactionInfo-Marshaller.h**

6.560 cms::TransactionInProgressException Class Reference

This exception is thrown when an operation is invalid because a transaction is in progress.

```
#include <src/main/cms/TransactionInProgressException.h>
```

Inheritance diagram for cms::TransactionInProgressException:

Public Member Functions

- **TransactionInProgressException** ()
- **TransactionInProgressException** (const **TransactionInProgressException** &ex)
- **TransactionInProgressException** (const std::string &message)
- **TransactionInProgressException** (const std::string &message, const std::exception *cause)
- **TransactionInProgressException** (const std::string &message, const std::exception *cause, const std::vector< std::pair< std::string, int > > &stack-Trace)
- virtual ~**TransactionInProgressException** () throw ()

6.560.1 Detailed Description

This exception is thrown when an operation is invalid because a transaction is in progress.

For instance, an attempt to call **Session::commit** (p. 2366) when a session is part of a distributed transaction should throw a **TransactionInProgressException** (p. 2782).

Since

2.3

6.560.2 Constructor & Destructor Documentation

6.560.2.1 **cms::TransactionInProgressException::TransactionInProgressException ()**

6.560.2.2 **cms::TransactionInProgressException::TransactionInProgressException (const TransactionInProgressException & *ex*)**

6.560.2.3 **cms::TransactionInProgressException::TransactionInProgressException (const std::string & *message*)**

6.560.2.4 **cms::TransactionInProgressException::TransactionInProgressException (const std::string & *message*, const std::exception * *cause*)**

6.560.2.5 **cms::TransactionInProgressException::TransactionInProgressException (const std::string & *message*, const std::exception * *cause*, const std::vector< std::pair< std::string, int > > & *stackTrace*)**

6.560.2.6 **virtual cms::TransactionInProgressException::~~TransactionInProgressException () throw ()** [virtual]

The documentation for this class was generated from the following file:

- src/main/cms/**TransactionInProgressException.h**

6.561 cms::TransactionRolledBackException Class Reference

This exception must be thrown when a call to **Session.commit** (p. 2366) results in a rollback of the current transaction.

```
#include <src/main/cms/TransactionRolledBackException.h>
```

Inheritance diagram for cms::TransactionRolledBackException:

Public Member Functions

- **TransactionRolledBackException** ()
- **TransactionRolledBackException** (const **TransactionRolledBackException** &ex)
- **TransactionRolledBackException** (const std::string &message)
- **TransactionRolledBackException** (const std::string &message, const std::exception *cause)
- **TransactionRolledBackException** (const std::string &message, const std::exception *cause, const std::vector< std::pair< std::string, int > > &stackTrace)
- virtual ~**TransactionRolledBackException** () throw ()

6.561.1 Detailed Description

This exception must be thrown when a call to **Session.commit** (p.2366) results in a rollback of the current transaction.

Since

2.3

6.561.2 Constructor & Destructor Documentation

6.561.2.1 **cms::TransactionRolledBackException::TransactionRolledBackException** ()

6.561.2.2 **cms::TransactionRolledBackException::TransactionRolledBackException** (const **TransactionRolledBackException** & *ex*)

6.561.2.3 **cms::TransactionRolledBackException::TransactionRolledBackException** (const std::string & *message*)

6.561.2.4 **cms::TransactionRolledBackException::TransactionRolledBackException** (const std::string & *message*, const std::exception * *cause*)

6.561.2.5 **cms::TransactionRolledBackException::TransactionRolledBackException** (const std::string & *message*, const std::exception * *cause*, const std::vector< std::pair< std::string, int > > & *stackTrace*)

6.561.2.6 virtual **cms::TransactionRolledBackException::~~TransactionRolledBackException** () throw () [virtual]

The documentation for this class was generated from the following file:

- `src/main/cms/TransactionRolledBackException.h`

6.562 activemq::state::TransactionState Class Reference

```
#include <src/main/activemq/state/TransactionState.h>
```

Public Member Functions

- **TransactionState** (const **Pointer**< **TransactionId** > &id)
- virtual **~TransactionState** ()
- std::string **toString** () const
- void **addCommand** (const **Pointer**< **Command** > &operation)
- void **checkShutdown** () const
- void **shutdown** ()
- const **LinkedList**< **Pointer** < **Command** > > &**getCommands** () const
- const **Pointer**< **TransactionId** > &**getId** () const
- void **setPrepared** (bool prepared)
- bool **isPrepared** () const
- void **setPreparedResult** (int preparedResult)
- int **getPreparedResult** () const
- void **addProducerState** (const **Pointer**< **ProducerState** > &producerState)
- std::vector< **Pointer** < **ProducerState** > > &**getProducerStates** ()

6.562.1 Constructor & Destructor Documentation

- 6.562.1.1 **activemq::state::TransactionState::TransactionState** (const **Pointer**< **TransactionId** > &id)
- 6.562.1.2 virtual **activemq::state::TransactionState::~~TransactionState** ()
[virtual]

6.562.2 Member Function Documentation

- 6.562.2.1 void **activemq::state::TransactionState::addCommand** (const **Pointer**< **Command** > &operation)
- 6.562.2.2 void **activemq::state::TransactionState::addProducerState** (const **Pointer**< **ProducerState** > &producerState)
- 6.562.2.3 void **activemq::state::TransactionState::checkShutdown** () const
- 6.562.2.4 const **LinkedList**< **Pointer**<**Command**> >& **activemq::state::TransactionState::getCommands** () const
[inline]

- 6.562.2.5 `const Pointer<TransactionId>& activemq::state::TransactionState::getId () const [inline]`
- 6.562.2.6 `int activemq::state::TransactionState::getPreparedResult () const [inline]`
- 6.562.2.7 `std::vector< Pointer<ProducerState> > activemq::state::TransactionState::getProducerStates ()`
- 6.562.2.8 `bool activemq::state::TransactionState::isPrepared () const [inline]`
- 6.562.2.9 `void activemq::state::TransactionState::setPrepared (bool prepared) [inline]`
- 6.562.2.10 `void activemq::state::TransactionState::setPreparedResult (int preparedResult) [inline]`
- 6.562.2.11 `void activemq::state::TransactionState::shutdown () [inline]`
- 6.562.2.12 `std::string activemq::state::TransactionState::toString () const`

The documentation for this class was generated from the following file:

- `src/main/activemq/state/TransactionState.h`

6.563 decaf::internal::util::concurrent::Transferer< E > Class - Template Reference

Shared internal API for dual stacks and queues.

```
#include <src/main/decaf/internal/util/concurrent/Transferer.h>
```

Inheritance diagram for `decaf::internal::util::concurrent::Transferer< E >`:

6.563.1 Detailed Description

```
template<typename E>class decaf::internal::util::concurrent::Transferer< E >
```

Shared internal API for dual stacks and queues.

The documentation for this class was generated from the following file:

- `src/main/decaf/internal/util/concurrent/Transferer.h`

6.564 decaf::internal::util::concurrent::TransferQueue< E > Class Template Reference

This extends Scherer-Scott dual queue algorithm, differing, among other ways, by using modes within nodes rather than marked pointers.

```
#include <src/main/decaf/internal/util/concurrent/TransferQueue.h>
```

Inheritance diagram for decaf::internal::util::concurrent::TransferQueue< E >:

Public Member Functions

- **TransferQueue** ()
*Node class for **TransferQueue** (p. 2787).*
- virtual ~**TransferQueue** ()
- virtual void **transfer** (E *e, bool timed, long long nanos)
Performs a put.
- virtual E * **transfer** (bool timed, long long nanos)
Performs a take.

6.564.1 Detailed Description

```
template<typename E>class decaf::internal::util::concurrent::TransferQueue< E >
```

This extends Scherer-Scott dual queue algorithm, differing, among other ways, by using modes within nodes rather than marked pointers.

The algorithm is a little simpler than that for stacks because fulfillers do not need explicit nodes, and matching is done by CAS'ing QNode.item field from non-null to null (for put) or vice versa (for take).

6.564.2 Constructor & Destructor Documentation

```
6.564.2.1 template<typename E > decaf::internal::util::concurrent::TransferQueue< E >::TransferQueue ( ) [inline]
```

Node class for **TransferQueue** (p. 2787).

Tries to cancel by CAS'ing ref to NULL if that succeeds then we mark as cancelled. Returns true if this node is known to be off the queue because its next pointer has been forgotten due to an advanceHead operation.

6.564 decaf::internal::util::concurrent::TransferQueue< E > Class Template Reference 2795

6.564.2.2 `template<typename E> virtual decaf::internal::util::concurrent-
::TransferQueue< E >::~~TransferQueue () [inline,
virtual]`

6.564.3 Member Function Documentation

6.564.3.1 `template<typename E> virtual void decaf::internal::util::concurrent:-
TransferQueue< E >::transfer (E * e, bool timed, long long nanos)
[inline, virtual]`

Performs a put.

Parameters

<i>e</i>	the item to be handed to a consumer;
<i>timed</i>	if this operation should timeout
<i>nanos</i>	the timeout, in nanoseconds

Exceptions

<i>TimeoutException</i>	if the operation timed out waiting for the consumer to accept the item offered.
<i>InterruptedException</i>	if the thread was interrupted while waiting for the consumer to accept the item offered.

Implements **decaf::internal::util::concurrent::Transferer< E >** (p. 2786).

6.564.3.2 `template<typename E> virtual E* decaf::internal::util::concurrent:-
TransferQueue< E >::transfer (bool timed, long long nanos) [inline,
virtual]`

Performs a take.

Parameters

<i>timed</i>	if this operation should timeout
<i>nanos</i>	the timeout, in nanoseconds

Returns

the item provided or received;

Exceptions

<i>TimeoutException</i>	if the operation timed out waiting for the producer to offer an item.
<i>InterruptedException</i>	if the thread was interrupted while waiting for the producer to offer an item.

Implements **decaf::internal::util::concurrent::Transferer**< E > (p. 2786).

The documentation for this class was generated from the following file:

- src/main/decaf/internal/util/concurrent/**TransferQueue.h**

6.565 decaf::internal::util::concurrent::TransferStack< E > Class Template Reference

```
#include <src/main/decaf/internal/util/concurrent/Transfer-
Stack.h>
```

Inheritance diagram for decaf::internal::util::concurrent::TransferStack< E >:

Public Member Functions

- **TransferStack** ()
- virtual **~TransferStack** ()
- virtual void **transfer** (E *e, bool timed, long long nanos)
Performs a put.
- virtual E * **transfer** (bool timed, long long nanos)
Performs a take.

```
template<typename E> class decaf::internal::util::concurrent::TransferStack< E >
```

6.565.1 Constructor & Destructor Documentation

6.565.1.1 `template<typename E > decaf::internal::util::concurrent::TransferStack< E >::TransferStack ()` [inline]

6.565.1.2 `template<typename E > virtual decaf::internal::util::concurrent::TransferStack< E >::~~TransferStack ()` [inline, virtual]

6.565.2 Member Function Documentation

6.565.2.1 `template<typename E > virtual void decaf::internal::util::concurrent::TransferStack< E >::transfer (E * e, bool timed, long long nanos)` [inline, virtual]

Performs a put.

Parameters

<i>e</i>	the item to be handed to a consumer;
<i>timed</i>	if this operation should timeout
<i>nanos</i>	the timeout, in nanoseconds

Exceptions

<i>TimeoutException</i>	if the operation timed out waiting for the consumer to accept the item offered.
<i>InterruptedException</i>	if the thread was interrupted while waiting for the consumer to accept the item offered.

Implements **decaf::internal::util::concurrent::Transferer< E >** (p. 2786).

6.565.2.2 `template<typename E> virtual E* decaf::internal::util::concurrent::-
TransferStack< E >::transfer (bool timed, long long nanos) [inline,
virtual]`

Performs a take.

Parameters

<i>timed</i>	if this operation should timeout
<i>nanos</i>	the timeout, in nanoseconds

Returns

the item provided or received;

Exceptions

<i>TimeoutException</i>	if the operation timed out waiting for the producer to offer an item.
<i>InterruptedException</i>	if the thread was interrupted while waiting for the producer to offer an item.

Implements **decaf::internal::util::concurrent::Transferer< E >** (p. 2786).

The documentation for this class was generated from the following file:

- `src/main/decaf/internal/util/concurrent/TransferStack.h`

6.566 activemq::transport::Transport Class Reference

Interface for a transport layer for command objects.

```
#include <src/main/activemq/transport/Transport.h>
```

Inheritance diagram for activemq::transport::Transport:

Public Member Functions

- virtual **~Transport** ()
- virtual void **start** ()=0

*Starts the **Transport** (p. 2790), the send methods of a **Transport** (p. 2790) will throw an exception if used before the **Transport** (p. 2790) is started.*
- virtual void **stop** ()=0

*Stops the **Transport** (p. 2790).*
- virtual void **oneway** (const **Pointer**< **Command** > &command)=0

Sends a one-way command.
- virtual **Pointer**< **Response** > **request** (const **Pointer**< **Command** > &command)=0

Sends the given command to the broker and then waits for the response.
- virtual **Pointer**< **Response** > **request** (const **Pointer**< **Command** > &command, unsigned int timeout)=0

Sends the given command to the broker and then waits for the response.
- virtual **Pointer** < **wireformat::WireFormat** > **getWireFormat** () const =0

Gets the WireFormat instance that is in use by this transport.
- virtual void **setWireFormat** (const **Pointer**< **wireformat::WireFormat** > &wireFormat)=0

Sets the WireFormat instance to use.
- virtual void **setTransportListener** (**TransportListener** *listener)=0

Sets the observer of asynchronous events from this transport.
- virtual **TransportListener** * **getTransportListener** () const =0

Gets the observer of asynchronous events from this transport.
- virtual **Transport** * **narrow** (const std::type_info &typeid)=0

*Narrows down a Chain of Transports to a specific **Transport** (p. 2790) to allow a higher level transport to skip intermediate Transports in certain circumstances.*
- virtual bool **isFaultTolerant** () const =0

*Is this **Transport** (p. 2790) fault tolerant, meaning that it will reconnect to a broker on disconnect.*
- virtual bool **isConnected** () const =0

*Is the **Transport** (p. 2790) Connected to its Broker.*
- virtual bool **isClosed** () const =0

*Has the **Transport** (p. 2790) been shutdown and no longer usable.*
- virtual bool **isReconnectSupported** () const =0
- virtual bool **isUpdateURIsSupported** () const =0
- virtual std::string **getRemoteAddress** () const =0
- virtual void **reconnect** (const **decaf::net::URI** &uri)=0

reconnect to another location
- virtual void **updateURIs** (bool rebalance, const **decaf::util::List**< **decaf::net::URI** > &uris)=0

*Updates the set of URIs the **Transport** (p. 2790) can connect to.*

6.566.1 Detailed Description

Interface for a transport layer for command objects.

Callers can send oneway messages or make synchronous requests. Non-response messages will be delivered to the specified listener object upon receipt. A user of the **Transport** (p. 2790) can set an exception listener to be notified of errors that occurs in Threads that the **Transport** (p. 2790) layer runs. Transports should be given an instance of a **WireFormat** object when created so that they can turn the built in Commands to / from the required wire format encoding.

6.566.2 Constructor & Destructor Documentation

6.566.2.1 `virtual activemq::transport::Transport::~Transport () [inline, virtual]`

6.566.3 Member Function Documentation

6.566.3.1 `virtual std::string activemq::transport::Transport::getRemoteAddress () const [pure virtual]`

Returns

the remote address for this connection

Implemented in **activemq::transport::IOTransport** (p. 1551), **activemq::transport::mock::MockTransport** (p. 1952), **activemq::transport::failover::FailoverTransport** (p. 1309), and **activemq::transport::TransportFilter** (p. 2803).

6.566.3.2 `virtual TransportListener* activemq::transport::Transport::getTransportListener () const [pure virtual]`

Gets the observer of asynchronous events from this transport.

Returns

the listener of transport events.

Implemented in **activemq::transport::IOTransport** (p. 1551), **activemq::transport::mock::MockTransport** (p. 1952), **activemq::transport::failover::FailoverTransport** (p. 1309), and **activemq::transport::TransportFilter** (p. 2803).

6.566.3.3 `virtual Pointer<wireformat::WireFormat> activemq::transport::Transport::getWireFormat () const [pure virtual]`

Gets the WireFormat instance that is in use by this transport.

In the case of nested transport this method delegates down to the lowest level transport that actually maintains a `WireFormat` info instance.

Returns

The `WireFormat` the object used to encode / decode commands.

Implemented in `activemq::transport::IOTransport` (p. 1551), `activemq::transport::mock::MockTransport` (p. 1952), `activemq::transport::failover::FailoverTransport` (p. 1310), and `activemq::transport::TransportFilter` (p. 2804).

6.566.3.4 `virtual bool activemq::transport::Transport::isClosed () const [pure virtual]`

Has the **Transport** (p. 2790) been shutdown and no longer usable.

Returns

true if the **Transport** (p. 2790)

Implemented in `activemq::transport::IOTransport` (p. 1552), `activemq::transport::mock::MockTransport` (p. 1952), `activemq::transport::TransportFilter` (p. 2804), `activemq::transport::failover::FailoverTransport` (p. 1311), and `activemq::transport::tcp::TcpTransport` (p. 2696).

6.566.3.5 `virtual bool activemq::transport::Transport::isConnected () const [pure virtual]`

Is the **Transport** (p. 2790) Connected to its Broker.

Returns

true if a connection has been made.

Implemented in `activemq::transport::IOTransport` (p. 1552), `activemq::transport::mock::MockTransport` (p. 1953), `activemq::transport::failover::FailoverTransport` (p. 1311), `activemq::transport::TransportFilter` (p. 2804), and `activemq::transport::tcp::TcpTransport` (p. 2696).

6.566.3.6 `virtual bool activemq::transport::Transport::isFaultTolerant () const [pure virtual]`

Is this **Transport** (p. 2790) fault tolerant, meaning that it will reconnect to a broker on disconnect.

Returns

true if the **Transport** (p. 2790) is fault tolerant.

Implemented in **activemq::transport::IOTransport** (p. 1552), **activemq::transport::mock::MockTransport** (p. 1953), **activemq::transport::failover::FailoverTransport** (p. 1311), **activemq::transport::TransportFilter** (p. 2804), and **activemq::transport::tcp::TcpTransport** (p. 2696).

6.566.3.7 `virtual bool activemq::transport::Transport::isReconnectSupported ()`
`const [pure virtual]`

Returns

true if reconnect is supported.

Implemented in **activemq::transport::mock::MockTransport** (p. 1954), **activemq::transport::failover::FailoverTransport** (p. 1312), **activemq::transport::IOTransport** (p. 1552), and **activemq::transport::TransportFilter** (p. 2805).

6.566.3.8 `virtual bool activemq::transport::Transport::isUpdateURIsSupported ()`
`const [pure virtual]`

Returns

true if updating uris is supported.

Implemented in **activemq::transport::mock::MockTransport** (p. 1954), **activemq::transport::failover::FailoverTransport** (p. 1312), **activemq::transport::IOTransport** (p. 1552), and **activemq::transport::TransportFilter** (p. 2805).

6.566.3.9 `virtual Transport* activemq::transport::Transport::narrow (const`
`std::type_info & typeId) [pure virtual]`

Narrows down a Chain of Transports to a specific **Transport** (p. 2790) to allow a higher level transport to skip intermediate Transports in certain circumstances.

Parameters

<i>typeId</i>	- The type_info of the Object we are searching for.
---------------	---

Returns

the requested Object. or NULL if its not in this chain.

Implemented in **activemq::transport::IOTransport** (p. 1553), **activemq::transport::mock::MockTransport** (p. 1954), **activemq::transport::failover::FailoverTransport** (p. 1312), and **activemq::transport::TransportFilter** (p. 2805).

6.566.3.10 `virtual void activemq::transport::Transport::oneway (const Pointer< Command> & command) [pure virtual]`

Sends a one-way command.

Does not wait for any response from the broker.

Parameters

<i>command</i>	The command to be sent.
----------------	-------------------------

Exceptions

<i>IOException</i>	if an exception occurs during writing of the command.
<i>UnsupportedOperationException</i>	if this method is not implemented by this transport.

Implemented in `activemq::transport::mock::MockTransport` (p. 1954), `activemq::transport::IOTransport` (p. 1553), `activemq::transport::failover::FailoverTransport` (p. 1313), `activemq::transport::TransportFilter` (p. 2806), `activemq::wireformat::openwire::OpenWireFormatNegotiator` (p. 2062), `activemq::transport::correlator::ResponseCorrelator` (p. 2305), `activemq::transport::inactivity::InactivityMonitor` (p. 1427), and `activemq::transport::logging::LoggingTransport` (p. 1710).

6.566.3.11 `virtual void activemq::transport::Transport::reconnect (const decaf::net::URI & uri) [pure virtual]`

reconnect to another location

Parameters

<i>uri</i>	The new URI to connect this Transport (p. 2790) to.
------------	--

Exceptions

<i>IOException</i>	on failure or if reconnect is not supported.
--------------------	--

Implemented in `activemq::transport::failover::FailoverTransport` (p. 1313), and `activemq::transport::TransportFilter` (p. 2807).

6.566.3.12 `virtual Pointer<Response> activemq::transport::Transport::request (const Pointer< Command> & command) [pure virtual]`

Sends the given command to the broker and then waits for the response.

Parameters

<i>command</i>	the command to be sent.
----------------	-------------------------

Returns

the response from the broker.

Exceptions

<i>IOException</i>	if an exception occurs during the read of the command.
<i>Unsupported-Operation</i>	if this method is not implemented by this transport.

Implemented in **activemq::transport::mock::MockTransport** (p. 1955), **activemq::transport::IOTransport** (p. 1553), **activemq::transport::failover::FailoverTransport** (p. 1314), **activemq::transport::TransportFilter** (p. 2807), **activemq::wireformat::openwire::OpenWireFormatNegotiator** (p. 2063), **activemq::transport::correlator::ResponseCorrelator** (p. 2306), and **activemq::transport::logging::LoggingTransport** (p. 1711).

```
6.566.3.13 virtual Pointer<Response> activemq::transport::Transport::request (
    const Pointer< Command > & command, unsigned int timeout ) [pure
    virtual]
```

Sends the given command to the broker and then waits for the response.

Parameters

<i>command</i>	The command to be sent.
<i>timeout</i>	The time to wait for this response.

Returns

the response from the broker.

Exceptions

<i>IOException</i>	if an exception occurs during the read of the command.
<i>Unsupported-Operation</i>	if this method is not implemented by this transport.

Implemented in **activemq::transport::IOTransport** (p. 1554), **activemq::transport::mock::MockTransport** (p. 1955), **activemq::transport::failover::FailoverTransport** (p. 1315), **activemq::wireformat::openwire::OpenWireFormatNegotiator** (p. 2063), **activemq::transport::TransportFilter** (p. 2807), **activemq::transport::correlator::ResponseCorrelator** (p. 2306), and **activemq::transport::logging::LoggingTransport** (p. 1711).

6.566.3.14 virtual void **activemq::transport::Transport::setTransportListener** (
TransportListener * listener) [pure virtual]

Sets the observer of asynchronous events from this transport.

Parameters

<i>listener</i>	the listener of transport events.
-----------------	-----------------------------------

Implemented in **activemq::transport::IOTransport** (p. 1555), **activemq::transport::mock::MockTransport** (p. 1957), **activemq::transport::failover::FailoverTransport** (p. 1316), and **activemq::transport::TransportFilter** (p. 2808).

6.566.3.15 virtual void **activemq::transport::Transport::setWireFormat** (const
Pointer< wireformat::WireFormat > & wireFormat) [pure virtual]

Sets the WireFormat instance to use.

Parameters

<i>wireFormat</i>	The WireFormat the object used to encode / decode commands.
-------------------	---

Implemented in **activemq::transport::IOTransport** (p. 1555), and **activemq::transport::TransportFilter** (p. 2808).

6.566.3.16 virtual void **activemq::transport::Transport::start** () [pure
virtual]

Starts the **Transport** (p. 2790), the send methods of a **Transport** (p. 2790) will throw an exception if used before the **Transport** (p. 2790) is started.

Exceptions

<i>IOException</i>	if an error occurs while starting the Transport (p. 2790).
--------------------	---

Implemented in **activemq::transport::IOTransport** (p. 1555), **activemq::transport::mock::MockTransport** (p. 1958), **activemq::wireformat::openwire::OpenWireFormatNegotiator** (p. 2064), **activemq::transport::TransportFilter** (p. 2808), **activemq::transport::failover::FailoverTransport** (p. 1317), and **activemq::transport::correlator::ResponseCorrelator** (p. 2307).

6.566.3.17 virtual void **activemq::transport::Transport::stop** () [pure
virtual]

Stops the **Transport** (p. 2790).

Exceptions

<i>IOException</i>	if an error occurs while stopping the transport.
--------------------	--

Implemented in **activemq::transport::IOTransport** (p. 1556), **activemq::transport::mock::MockTransport** (p. 1958), **activemq::transport::TransportFilter** (p. 2809), and **activemq::transport::failover::FailoverTransport** (p. 1317).

6.566.3.18 `virtual void activemq::transport::Transport::updateURIs (bool rebalance, const decaf::util::List< decaf::net::URI > & uris) [pure virtual]`

Updates the set of URIs the **Transport** (p. 2790) can connect to.

If the **Transport** (p. 2790) doesn't support updating its URIs then an *IOException* is thrown.

Parameters

<i>rebalance</i>	Indicates if a forced reconnection should be performed as a result of the update.
<i>uris</i>	The new list of URIs that can be used for connection.

Exceptions

<i>IOException</i>	if an error occurs or updates aren't supported.
--------------------	---

Implemented in **activemq::transport::failover::FailoverTransport** (p. 1317), and **activemq::transport::TransportFilter** (p. 2809).

The documentation for this class was generated from the following file:

- `src/main/activemq/transport/Transport.h`

6.567 activemq::transport::TransportFactory Class Reference

Defines the interface for Factories that create Transports or TransportFilters.

```
#include <src/main/activemq/transport/TransportFactory.h>
```

Inheritance diagram for `activemq::transport::TransportFactory`:

Public Member Functions

- `virtual ~TransportFactory ()`
- `virtual Pointer< Transport > create (const decaf::net::URI &location)=0`

*Creates a fully configured **Transport** (p. 2790) instance which could be a chain of filters and transports.*

- virtual **Pointer< Transport > createComposite** (const **decaf::net::URI** &location)=0

*Creates a slimmed down **Transport** (p. 2790) instance which can be used in composite transport instances.*

6.567.1 Detailed Description

Defines the interface for Factories that create Transports or TransportFilters.

The factory should be able to create either a completely configured **Transport** (p. 2790) meaning that it has all the appropriate filters wrapping it, or it should be able to create a slimmed down version that is used in composite transports like Failover or Fanout.

Since

3.0

6.567.2 Constructor & Destructor Documentation

6.567.2.1 virtual **activemq::transport::TransportFactory::~TransportFactory** ()
[inline, virtual]

6.567.3 Member Function Documentation

6.567.3.1 virtual **Pointer<Transport> activemq::transport::TransportFactory::create** (const **decaf::net::URI** & *location*) [pure virtual]

Creates a fully configured **Transport** (p. 2790) instance which could be a chain of filters and transports.

Parameters

<i>location</i>	- URI location to connect to plus any properties to assign.
-----------------	---

Exceptions

<i>ActiveMQException</i>	if an error occurs
--------------------------	--------------------

Implemented in **activemq::transport::failover::FailoverTransportFactory** (p. 1319), **activemq::transport::mock::MockTransportFactory** (p. 1959), and **activemq::transport::tcp::TcpTransportFactory** (p. 2697).

6.567.3.2 `virtual Pointer<Transport> activemq::transport::TransportFactory-
::createComposite (const decaf::net::URI & location) [pure
virtual]`

Creates a slimed down **Transport** (p. 2790) instance which can be used in composite transport instances.

Parameters

<i>location</i>	- URI location to connect to plus any properties to assign.
-----------------	---

Exceptions

<i>ActiveMQException</i>	if an error occurs
--------------------------	--------------------

Implemented in **activemq::transport::mock::MockTransportFactory** (p. 1960), **activemq::transport::failover::FailoverTransportFactory** (p. 1319), and **activemq::transport::tcp::TcpTransportFactory** (p. 2698).

The documentation for this class was generated from the following file:

- src/main/activemq/transport/**TransportFactory.h**

6.568 activemq::transport::TransportFilter Class Reference

A filter on the transport layer.

```
#include <src/main/activemq/transport/TransportFilter.h>
```

Inheritance diagram for activemq::transport::TransportFilter:

Public Member Functions

- **TransportFilter** (const **Pointer**< **Transport** > &next)
Constructor.
- virtual ~**TransportFilter** ()
- virtual void **onCommand** (const **Pointer**< **Command** > &command)
Event handler for the receipt of a command.
- virtual void **onException** (const **decaf::lang::Exception** &ex)
Event handler for an exception from a command transport.
- virtual void **transportInterrupted** ()
The transport has suffered an interruption from which it hopes to recover.
- virtual void **transportResumed** ()
The transport has resumed after an interruption.
- virtual void **oneway** (const **Pointer**< **Command** > &command)

Sends a one-way command.

- virtual **Pointer< Response > request** (const **Pointer< Command > &command**)

Sends the given command to the broker and then waits for the response.

- virtual **Pointer< Response > request** (const **Pointer< Command > &command**, unsigned int timeout)

Sends the given command to the broker and then waits for the response.

- virtual void **setTransportListener** (**TransportListener *listener**)

Sets the observer of asynchronous events from this transport.

- virtual **TransportListener * getTransportListener** () const

Gets the observer of asynchronous events from this transport.

- virtual **Pointer < wireformat::WireFormat > getWireFormat** () const

Gets the WireFormat instance that is in use by this transport.

- virtual void **setWireFormat** (const **Pointer< wireformat::WireFormat > &wireFormat**)

Sets the WireFormat instance to use.

- virtual void **start** ()

*Starts the **Transport** (p. 2790), the send methods of a **Transport** (p. 2790) will throw an exception if used before the **Transport** (p. 2790) is started.*

- virtual void **stop** ()

*Stops the **Transport** (p. 2790).*

- virtual void **close** ()

Closes this object and deallocates the appropriate resources.

- virtual **Transport * narrow** (const std::type_info &typeid)

*Narrows down a Chain of Transports to a specific **Transport** (p. 2790) to allow a higher level transport to skip intermediate Transports in certain circumstances.*

- virtual bool **isFaultTolerant** () const

*Is this **Transport** (p. 2790) fault tolerant, meaning that it will reconnect to a broker on disconnect.*

- virtual bool **isConnected** () const

*Is the **Transport** (p. 2790) Connected to its Broker.*

- virtual bool **isReconnectSupported** () const

- virtual bool **isUpdateURIsSupported** () const

- virtual bool **isClosed** () const

*Has the **Transport** (p. 2790) been shutdown and no longer usable.*

- virtual std::string **getRemoteAddress** () const

- virtual void **reconnect** (const **decaf::net::URI &uri**)

reconnect to another location

- virtual void **updateURIs** (bool rebalance, const **decaf::util::List< decaf::net::URI > &uris**)

*Updates the set of URIs the **Transport** (p. 2790) can connect to.*

Protected Member Functions

- void **fire** (const **decaf::lang::Exception** &ex)
Notify the listener of the thrown Exception.
- void **fire** (const **Pointer**< **Command** > &command)
Notify the listener of the new incoming Command.

Protected Attributes

- **Pointer**< **Transport** > **next**
The transport that this filter wraps around.
- **TransportListener** * **listener**
Listener of this transport.

6.568.1 Detailed Description

A filter on the transport layer.

Transport (p. 2790) filters implement the **Transport** (p. 2790) interface and optionally delegate calls to another **Transport** (p. 2790) object.

Since

1.0

6.568.2 Constructor & Destructor Documentation

6.568.2.1 **activemq::transport::TransportFilter::TransportFilter** (const **Pointer**< **Transport** > & *next*)

Constructor.

Parameters

<i>next</i>	- the next Transport (p. 2790) in the chain
-------------	--

6.568.2.2 **virtual activemq::transport::TransportFilter::~~TransportFilter** ()
[virtual]

6.568.3 Member Function Documentation

6.568.3.1 **virtual void activemq::transport::TransportFilter::close** () [virtual]

Closes this object and deallocates the appropriate resources.

The object is generally no longer usable after calling close.

Exceptions

<i>IOException</i>	if an error occurs while closing.
--------------------	-----------------------------------

Implements **decaf::io::Closeable** (p. 817).

Reimplemented in **activemq::wireformat::openwire::OpenWireFormatNegotiator** (p. 2062), **activemq::transport::tcp::TcpTransport** (p. 2694), **activemq::transport::correlator::ResponseCorrelator** (p. 2304), and **activemq::transport::inactivity::InactivityMonitor** (p. 1427).

6.568.3.2 `void activemq::transport::TransportFilter::fire (const decaf::lang::Exception & ex)` [protected]

Notify the listener of the thrown Exception.

Parameters

<i>ex</i>	- the exception to send to listeners
-----------	--------------------------------------

6.568.3.3 `void activemq::transport::TransportFilter::fire (const Pointer< Command > & command)` [protected]

Notify the listener of the new incoming Command.

Parameters

<i>command</i>	- the command to send to the listener
----------------	---------------------------------------

6.568.3.4 `virtual std::string activemq::transport::TransportFilter::getRemoteAddress () const` [inline, virtual]

Returns

the remote address for this connection

Implements **activemq::transport::Transport** (p. 2792).

6.568.3.5 `virtual TransportListener* activemq::transport::TransportFilter::getTransportListener () const` [inline, virtual]

Gets the observer of asynchronous events from this transport.

Returns

the listener of transport events.

Implements **activemq::transport::Transport** (p. 2792).

6.568.3.6 `virtual Pointer<wireformat::WireFormat> activemq::transport-
::TransportFilter::getWireFormat () const [inline,
virtual]`

Gets the WireFormat instance that is in use by this transport.

In the case of nested transport this method delegates down to the lowest level transport that actually maintains a WireFormat info instance.

Returns

The WireFormat the object used to encode / decode commands.

Implements **activemq::transport::Transport** (p. 2792).

6.568.3.7 `virtual bool activemq::transport::TransportFilter::isClosed () const
[inline, virtual]`

Has the **Transport** (p. 2790) been shutdown and no longer usable.

Returns

true if the **Transport** (p. 2790)

Implements **activemq::transport::Transport** (p. 2793).

Reimplemented in **activemq::transport::tcp::TcpTransport** (p. 2696).

6.568.3.8 `virtual bool activemq::transport::TransportFilter::isConnected () const
[inline, virtual]`

Is the **Transport** (p. 2790) Connected to its Broker.

Returns

true if a connection has been made.

Implements **activemq::transport::Transport** (p. 2793).

Reimplemented in **activemq::transport::tcp::TcpTransport** (p. 2696).

6.568.3.9 `virtual bool activemq::transport::TransportFilter::isFaultTolerant () const
[inline, virtual]`

Is this **Transport** (p. 2790) fault tolerant, meaning that it will reconnect to a broker on disconnect.

Returns

true if the **Transport** (p. 2790) is fault tolerant.

Implements **activemq::transport::Transport** (p. 2793).

Reimplemented in **activemq::transport::tcp::TcpTransport** (p. 2696).

6.568.3.10 `virtual bool activemq::transport::TransportFilter::isReconnectSupported () const [inline, virtual]`

Returns

true if reconnect is supported.

Implements **activemq::transport::Transport** (p. 2794).

6.568.3.11 `virtual bool activemq::transport::TransportFilter::isUpdateURIsSupported () const [inline, virtual]`

Returns

true if updating uris is supported.

Implements **activemq::transport::Transport** (p. 2794).

6.568.3.12 `virtual Transport* activemq::transport::TransportFilter::narrow (const std::type_info & typeId) [virtual]`

Narrows down a Chain of Transports to a specific **Transport** (p. 2790) to allow a higher level transport to skip intermediate Transports in certain circumstances.

Parameters

<i>typeId</i>	- The type_info of the Object we are searching for.
---------------	---

Returns

the requested Object. or NULL if its not in this chain.

Implements **activemq::transport::Transport** (p. 2794).

6.568.3.13 `virtual void activemq::transport::TransportFilter::onCommand (const Pointer< Command > & command) [virtual]`

Event handler for the receipt of a command.

Parameters

<i>command</i>	- the received command object.
----------------	--------------------------------

Implements **activemq::transport::TransportListener** (p. 2811).

Reimplemented in **activemq::wireformat::openwire::OpenWireFormatNegotiator** (p. 2062), **activemq::transport::correlator::ResponseCorrelator** (p. 2305), **activemq::transport::inactivity::InactivityMonitor** (p. 1427), and **activemq::transport::logging::LoggingTransport** (p. 1710).

6.568.3.14 virtual void **activemq::transport::TransportFilter::oneway** (const **Pointer< Command > & command**) [inline, virtual]

Sends a one-way command.

Does not wait for any response from the broker.

Parameters

<i>command</i>	The command to be sent.
----------------	-------------------------

Exceptions

<i>IOException</i>	if an exception occurs during writing of the command.
<i>Unsupported-OperationException</i>	if this method is not implemented by this transport.

Implements **activemq::transport::Transport** (p. 2795).

Reimplemented in **activemq::wireformat::openwire::OpenWireFormatNegotiator** (p. 2062), **activemq::transport::correlator::ResponseCorrelator** (p. 2305), **activemq::transport::inactivity::InactivityMonitor** (p. 1427), and **activemq::transport::logging::LoggingTransport** (p. 1710).

6.568.3.15 virtual void **activemq::transport::TransportFilter::onException** (const **decaf::lang::Exception & ex**) [virtual]

Event handler for an exception from a command transport.

Parameters

<i>ex</i>	The exception to handle.
-----------	--------------------------

Implements **activemq::transport::TransportListener** (p. 2811).

Reimplemented in **activemq::transport::inactivity::InactivityMonitor** (p. 1428).

6.568.3.16 `virtual void activemq::transport::TransportFilter::reconnect (const decaf::net::URI & uri) [virtual]`

reconnect to another location

Parameters

<i>uri</i>	The new URI to connect this Transport (p. 2790) to.
------------	--

Exceptions

<i>IOException</i>	on failure or if reconnect is not supported.
--------------------	--

Implements **activemq::transport::Transport** (p. 2795).

6.568.3.17 `virtual Pointer<Response> activemq::transport::TransportFilter::request (const Pointer< Command > & command) [inline, virtual]`

Sends the given command to the broker and then waits for the response.

Parameters

<i>command</i>	the command to be sent.
----------------	-------------------------

Returns

the response from the broker.

Exceptions

<i>IOException</i>	if an exception occurs during the read of the command.
<i>UnsupportedOperationException</i>	if this method is not implemented by this transport.

Implements **activemq::transport::Transport** (p. 2795).

Reimplemented in **activemq::wireformat::openwire::OpenWireFormatNegotiator** (p. 2063), **activemq::transport::correlator::ResponseCorrelator** (p. 2306), and **activemq::transport::logging::LoggingTransport** (p. 1711).

6.568.3.18 `virtual Pointer<Response> activemq::transport::TransportFilter::request (const Pointer< Command > & command, unsigned int timeout) [inline, virtual]`

Sends the given command to the broker and then waits for the response.

Parameters

<i>command</i>	The command to be sent.
<i>timeout</i>	The time to wait for this response.

Returns

the response from the broker.

Exceptions

<i>IOException</i>	if an exception occurs during the read of the command.
<i>Unsupported-Operation</i>	if this method is not implemented by this transport.

Implements **activemq::transport::Transport** (p. 2796).

Reimplemented in **activemq::wireformat::openwire::OpenWireFormatNegotiator** (p. 2063), **activemq::transport::correlator::ResponseCorrelator** (p. 2306), and **activemq::transport::logging::LoggingTransport** (p. 1711).

6.568.3.19 virtual void **activemq::transport::TransportFilter::setTransportListener** (**TransportListener** * *listener*) [inline, virtual]

Sets the observer of asynchronous events from this transport.

Parameters

<i>listener</i>	the listener of transport events.
-----------------	-----------------------------------

Implements **activemq::transport::Transport** (p. 2797).

6.568.3.20 virtual void **activemq::transport::TransportFilter::setWireFormat** (const **Pointer**< **wireformat::WireFormat** > & *wireFormat*) [inline, virtual]

Sets the WireFormat instance to use.

Parameters

<i>wireFormat</i>	The WireFormat the object used to encode / decode commands.
-------------------	---

Implements **activemq::transport::Transport** (p. 2797).

6.568.3.21 virtual void **activemq::transport::TransportFilter::start** () [virtual]

Starts the **Transport** (p. 2790), the send methods of a **Transport** (p. 2790) will throw an exception if used before the **Transport** (p. 2790) is started.

Exceptions

<i>IOException</i>	if an error occurs while starting the Transport (p. 2790).
--------------------	---

Implements **activemq::transport::Transport** (p. 2797).

Reimplemented in **activemq::wireformat::openwire::OpenWireFormatNegotiator** (p. 2064), and **activemq::transport::correlator::ResponseCorrelator** (p. 2307).

6.568.3.22 `virtual void activemq::transport::TransportFilter::stop () [virtual]`

Stops the **Transport** (p. 2790).

Exceptions

<i>IOException</i>	if an error occurs while stopping the transport.
--------------------	--

Implements **activemq::transport::Transport** (p. 2797).

6.568.3.23 `virtual void activemq::transport::TransportFilter::transportInterrupted () [virtual]`

The transport has suffered an interruption from which it hopes to recover.

Implements **activemq::transport::TransportListener** (p. 2812).

6.568.3.24 `virtual void activemq::transport::TransportFilter::transportResumed () [virtual]`

The transport has resumed after an interruption.

Implements **activemq::transport::TransportListener** (p. 2812).

6.568.3.25 `virtual void activemq::transport::TransportFilter::updateURIs (bool rebalance, const decaf::util::List< decaf::net::URI > & uris) [inline, virtual]`

Updates the set of URIs the **Transport** (p. 2790) can connect to.

If the **Transport** (p. 2790) doesn't support updating its URIs then an *IOException* is thrown.

Parameters

<i>rebalance</i>	Indicates if a forced reconnection should be performed as a result of the update.
<i>uris</i>	The new list of URIs that can be used for connection.

Exceptions

<code>IOException</code>	if an error occurs or updates aren't supported.
--------------------------	---

Implements `activemq::transport::Transport` (p. 2798).

6.568.4 Field Documentation

6.568.4.1 `TransportListener*` `activemq::transport::TransportFilter::listener`
[protected]

Listener of this transport.

6.568.4.2 `Pointer<Transport>` `activemq::transport::TransportFilter::next`
[protected]

The transport that this filter wraps around.

The documentation for this class was generated from the following file:

- `src/main/activemq/transport/TransportFilter.h`

6.569 `activemq::transport::TransportListener` Class Reference

A listener of asynchronous exceptions from a command transport object.

```
#include <src/main/activemq/transport/TransportListener.h>
```

Inheritance diagram for `activemq::transport::TransportListener`:

Public Member Functions

- virtual `~TransportListener()`
- virtual void `onCommand` (const `Pointer<Command>` &command)=0
Event handler for the receipt of a command.
- virtual void `onException` (const `decaf::lang::Exception` &ex)=0
Event handler for an exception from a command transport.
- virtual void `transportInterrupted` ()=0
The transport has suffered an interruption from which it hopes to recover.
- virtual void `transportResumed` ()=0
The transport has resumed after an interruption.

6.569.1 Detailed Description

A listener of asynchronous exceptions from a command transport object.

6.569.2 Constructor & Destructor Documentation

6.569.2.1 `virtual activemq::transport::TransportListener::~~TransportListener ()`
`[inline, virtual]`

6.569.3 Member Function Documentation

6.569.3.1 `virtual void activemq::transport::TransportListener::onCommand (const`
`Pointer< Command > & command) [pure virtual]`

Event handler for the receipt of a command.

The transport passes off all received commands to its listeners, the listener then owns the Object. If there is no registered listener the **Transport** (p. 2790) deletes the command upon receipt.

Parameters

<i>command</i>	The received command object.
----------------	------------------------------

Implemented in `activemq::core::ActiveMQConnection` (p. 204), `activemq::wireformat::openwire::OpenWireFormatNegotiator` (p. 2062), `activemq::transport::correlator::ResponseCorrelator` (p. 2305), `activemq::transport::TransportFilter` (p. 2805), `activemq::transport::inactivity::InactivityMonitor` (p. 1427), `activemq::transport::mock::InternalCommandListener` (p. 1528), `activemq::transport::failover::FailoverTransportListener` (p. 1321), and `activemq::transport::logging::LoggingTransport` (p. 1710).

6.569.3.2 `virtual void activemq::transport::TransportListener::onException (const`
`decaf::lang::Exception & ex) [pure virtual]`

Event handler for an exception from a command transport.

Parameters

<i>ex</i>	The exception being propagated to this listener to handle.
-----------	--

Implemented in `activemq::core::ActiveMQConnection` (p. 205), `activemq::transport::failover::BackupTransport` (p. 488), `activemq::transport::TransportFilter` (p. 2806), `activemq::transport::inactivity::InactivityMonitor` (p. 1428), and `activemq::transport::failover::FailoverTransportListener` (p. 1322).

6.569.3.3 `virtual void activemq::transport::TransportListener::transportInterrupted () [pure virtual]`

The transport has suffered an interruption from which it hopes to recover.

Implemented in `activemq::core::ActiveMQConnection` (p. 212), `activemq::transport::TransportFilter` (p. 2809), `activemq::transport::failover::FailoverTransportListener` (p. 1322), and `activemq::transport::DefaultTransportListener` (p. 1179).

6.569.3.4 `virtual void activemq::transport::TransportListener::transportResumed () [pure virtual]`

The transport has resumed after an interruption.

Implemented in `activemq::core::ActiveMQConnection` (p. 213), `activemq::transport::TransportFilter` (p. 2809), `activemq::transport::failover::FailoverTransportListener` (p. 1322), and `activemq::transport::DefaultTransportListener` (p. 1179).

The documentation for this class was generated from the following file:

- `src/main/activemq/transport/TransportListener.h`

6.570 `activemq::transport::TransportRegistry` Class Reference

Registry of all **Transport** (p. 2790) Factories that are available to the client at runtime.

```
#include <src/main/activemq/transport/TransportRegistry.h>
```

Public Member Functions

- `virtual ~TransportRegistry ()`
- `TransportFactory * findFactory (const std::string &name) const`
*Gets a Registered **TransportFactory** (p. 2798) from the Registry and returns it if there is not a registered format factory with the given name an exception is thrown.*
- `void registerFactory (const std::string &name, TransportFactory *factory)`
*Registers a new **TransportFactory** (p. 2798) with this Registry.*
- `void unregisterFactory (const std::string &name)`
Unregisters the Factory with the given name and deletes that instance of the Factory.
- `void unregisterAllFactories ()`
Removes all Factories and deletes the instances of the Factory objects.
- `std::vector< std::string > getTransportNames () const`
*Retrieves a list of the names of all the Registered **Transport** (p. 2790)'s in this Registry.*

Static Public Member Functions

- static **TransportRegistry** & **getInstance** ()

*Gets the single instance of the **TransportRegistry** (p. 2812).*

6.570.1 Detailed Description

Registry of all **Transport** (p. 2790) Factories that are available to the client at runtime.

New **Transport** (p. 2790)'s must have a factory registered here before a connection attempt is made.

Since

3.0

6.570.2 Constructor & Destructor Documentation

6.570.2.1 **virtual activemq::transport::TransportRegistry::~TransportRegistry** ()
[virtual]

6.570.3 Member Function Documentation

6.570.3.1 **TransportFactory* activemq::transport::TransportRegistry::findFactory**
(const std::string & *name*) const

Gets a Registered **TransportFactory** (p. 2798) from the Registry and returns it if there is not a registered format factory with the given name an exception is thrown.

Parameters

<i>name</i>	The name of the Factory to find in the Registry.
-------------	--

Returns

the Factory registered under the given name.

Exceptions

<i>NoSuchElementException</i>	if no factory is registered with that name.
-------------------------------	---

6.570.3.2 **static TransportRegistry& activemq::transport::TransportRegistry::getInstance** () [static]

Gets the single instance of the **TransportRegistry** (p. 2812).

Returns

reference to the single instance of this Registry

6.570.3.3 `std::vector<std::string> activemq::transport::TransportRegistry::get-TransportNames () const`

Retrieves a list of the names of all the Registered **Transport** (p. 2790)'s in this Registry.

Returns

stl vector of strings with all the **Transport** (p. 2790) names registered.

6.570.3.4 `void activemq::transport::TransportRegistry::registerFactory (const std::string & name, TransportFactory * factory)`

Registers a new **TransportFactory** (p. 2798) with this Registry.

If a Factory with the given name is already registered it is overwritten with the new one. Once a factory is added to the Registry its lifetime is controlled by the Registry, it will be deleted once the Registry has been deleted.

Parameters

<i>name</i>	The name of the new Factory to register.
<i>factory</i>	The new Factory to add to the Registry.

Exceptions

<i>IllegalArgument-Exception</i>	is name is the empty string.
<i>NullPointerException</i>	if the Factory is Null.

6.570.3.5 `void activemq::transport::TransportRegistry::unregisterAllFactories ()`

Removes all Factories and deletes the instances of the Factory objects.

6.570.3.6 `void activemq::transport::TransportRegistry::unregisterFactory (const std::string & name)`

Unregisters the Factory with the given name and deletes that instance of the Factory.

Parameters

<i>name</i>	Name of the Factory to unregister and destroy
-------------	---

The documentation for this class was generated from the following file:

- `src/main/activemq/transport/TransportRegistry.h`

6.571 tree_desc_s Struct Reference

```
#include <src/main/decaf/internal/util/zip/deflate.h>
```

Data Fields

- `ct_data * dyn_tree`
- `int max_code`
- `static_tree_desc * stat_desc`

6.571.1 Field Documentation

6.571.1.1 `ct_data* tree_desc_s::dyn_tree`

6.571.1.2 `int tree_desc_s::max_code`

6.571.1.3 `static_tree_desc* tree_desc_s::stat_desc`

The documentation for this struct was generated from the following file:

- `src/main/decaf/internal/util/zip/deflate.h`

6.572 decaf::lang::Thread::UncaughtExceptionHandler Class - Reference

Interface for handlers invoked when a **Thread** (p.2703) abruptly terminates due to an uncaught exception.

```
#include <src/main/decaf/lang/Thread.h>
```

Public Member Functions

- virtual `~UncaughtExceptionHandler ()`
- virtual void `uncaughtException (const Thread *thread, const Throwable &error)=0 throw ()`

Method invoked when the given thread terminates due to the given uncaught exception.

6.572.1 Detailed Description

Interface for handlers invoked when a **Thread** (p. 2703) abruptly terminates due to an uncaught exception.

6.572.2 Constructor & Destructor Documentation

6.572.2.1 virtual decaf::lang::Thread::UncaughtExceptionHandler-
::~UncaughtExceptionHandler () [inline,
virtual]

6.572.3 Member Function Documentation

6.572.3.1 virtual void decaf::lang::Thread::UncaughtExceptionHandler::uncaught-
Exception (const Thread * *thread*, const Throwable & *error*) throw ()
[pure virtual]

Method invoked when the given thread terminates due to the given uncaught exception.

This method is defined to indicate that it will not throw an exception, throwing and exception from this method will on most systems result in a segmentation fault.

The documentation for this class was generated from the following file:

- src/main/decaf/lang/Thread.h

6.573 decaf::net::UnknownHostException Class Reference

```
#include <src/main/decaf/net/UnknownHostException.h>
```

Inheritance diagram for decaf::net::UnknownHostException:

Public Member Functions

- **UnknownHostException** () throw ()
Default Constructor.
- **UnknownHostException** (const **Exception** &ex) throw ()
Conversion Constructor from some other Exception.
- **UnknownHostException** (const **UnknownHostException** &ex) throw ()
Copy Constructor.
- **UnknownHostException** (const char *file, const int lineNumber, const std::exception *cause, const char *msg,...) throw ()
Constructor - Initializes the file name and line number where this message occurred.
- **UnknownHostException** (const std::exception *cause) throw ()

Constructor.

- **UnknownHostException** (const char *file, const int lineNumber, const char *msg,...) throw ()

Constructor - Initializes the file name and line number where this message occurred.

- virtual **UnknownHostException** * **clone** () const

Clones this exception.

- virtual ~**UnknownHostException** () throw ()

6.573.1 Constructor & Destructor Documentation

6.573.1.1 **decaf::net::UnknownHostException::UnknownHostException () throw ()**
[inline]

Default Constructor.

6.573.1.2 **decaf::net::UnknownHostException::UnknownHostException (const Exception & ex) throw ()** [inline]

Conversion Constructor from some other Exception.

Parameters

<i>ex</i>	An exception that should become this type of Exception
-----------	--

6.573.1.3 **decaf::net::UnknownHostException::UnknownHostException (const UnknownHostException & ex) throw ()** [inline]

Copy Constructor.

Parameters

<i>ex</i>	An exception that should become this type of Exception
-----------	--

6.573.1.4 **decaf::net::UnknownHostException::UnknownHostException (const char * file, const int lineNumber, const std::exception * cause, const char * msg, ...) throw ()** [inline]

Constructor - Initializes the file name and line number where this message occurred.

Sets the message to report, using an optional list of arguments to parse into the message

Parameters

<i>file</i>	The file name where exception occurs
<i>lineNumber</i>	The line number where the exception occurred.

<i>cause</i>	The exception that was the cause for this one to be thrown.
<i>msg</i>	The message to report
...	list of primitives that are formatted into the message

6.573.1.5 `decaf::net::UnknownHostException::UnknownHostException (const std::exception * cause) throw () [inline]`

Constructor.

Parameters

<i>cause</i>	Pointer to the exception that caused this one to be thrown, the object is cloned caller retains ownership.
--------------	--

6.573.1.6 `decaf::net::UnknownHostException::UnknownHostException (const char * file, const int lineNumber, const char * msg, ...) throw () [inline]`

Constructor - Initializes the file name and line number where this message occurred.

Sets the message to report, using an optional list of arguments to parse into the message

Parameters

<i>file</i>	The file name where exception occurs
<i>lineNumber</i>	The line number where the exception occurred.
<i>msg</i>	The message to report
...	list of primitives that are formatted into the message

6.573.1.7 `virtual decaf::net::UnknownHostException::~~UnknownHostException () throw () [inline, virtual]`

6.573.2 Member Function Documentation

6.573.2.1 `virtual UnknownHostException* decaf::net::UnknownHostException::clone () const [inline, virtual]`

Clones this exception.

This is useful for cases where you need to preserve the type of the original exception as well as the message. All subclasses should override.

Returns

a new `Exception` instance that is a copy of this `Exception` object.

Reimplemented from **decaf::io::IOException** (p. 1547).

The documentation for this class was generated from the following file:

- `src/main/decaf/net/UnknownHostException.h`

6.574 decaf::net::UnknownServiceException Class Reference

```
#include <src/main/decaf/net/UnknownServiceException.h>
```

Inheritance diagram for `decaf::net::UnknownServiceException`:

Public Member Functions

- **UnknownServiceException** () throw ()
Default Constructor.
- **UnknownServiceException** (const **Exception** &ex) throw ()
Conversion Constructor from some other Exception.
- **UnknownServiceException** (const **UnknownServiceException** &ex) throw ()
Copy Constructor.
- **UnknownServiceException** (const char *file, const int lineNumber, const std::exception *cause, const char *msg,...) throw ()
Constructor - Initializes the file name and line number where this message occurred.
- **UnknownServiceException** (const std::exception *cause) throw ()
Constructor.
- **UnknownServiceException** (const char *file, const int lineNumber, const char *msg,...) throw ()
Constructor - Initializes the file name and line number where this message occurred.
- virtual **UnknownServiceException** * clone () const
Clones this exception.
- virtual ~**UnknownServiceException** () throw ()

6.574.1 Constructor & Destructor Documentation

6.574.1.1 decaf::net::UnknownServiceException::UnknownServiceException () throw () [inline]

Default Constructor.

6.574.1.2 decaf::net::UnknownServiceException::UnknownServiceException (
const Exception & ex) throw () `[inline]`

Conversion Constructor from some other Exception.

Parameters

<i>ex</i>	An exception that should become this type of Exception
-----------	--

6.574.1.3 decaf::net::UnknownServiceException::UnknownServiceException (
const UnknownServiceException & ex) throw () `[inline]`

Copy Constructor.

Parameters

<i>ex</i>	An exception that should become this type of Exception
-----------	--

6.574.1.4 decaf::net::UnknownServiceException::UnknownServiceException (
const char * file, const int lineNumber, const std::exception * cause, const char *
msg, ...) throw () `[inline]`

Constructor - Initializes the file name and line number where this message occurred.

Sets the message to report, using an optional list of arguments to parse into the message

Parameters

<i>file</i>	The file name where exception occurs
<i>lineNumber</i>	The line number where the exception occurred.
<i>cause</i>	The exception that was the cause for this one to be thrown.
<i>msg</i>	The message to report
<i>...</i>	list of primitives that are formatted into the message

6.574.1.5 decaf::net::UnknownServiceException::UnknownServiceException (
const std::exception * cause) throw () `[inline]`

Constructor.

Parameters

<i>cause</i>	Pointer to the exception that caused this one to be thrown, the object is cloned caller retains ownership.
--------------	--

6.574.1.6 **decaf::net::UnknownServiceException::UnknownServiceException**
 (const char * *file*, const int *lineNumber*, const char * *msg*, ...) throw ()
 [inline]

Constructor - Initializes the file name and line number where this message occurred.

Sets the message to report, using an optional list of arguments to parse into the message

Parameters

<i>file</i>	The file name where exception occurs
<i>lineNumber</i>	The line number where the exception occurred.
<i>msg</i>	The message to report
...	list of primitives that are formatted into the message

6.574.1.7 **virtual decaf::net::UnknownServiceException::~~UnknownServiceException** () throw () [inline, virtual]

6.574.2 Member Function Documentation

6.574.2.1 **virtual UnknownServiceException* decaf::net::UnknownServiceException::clone** () const [inline, virtual]

Clones this exception.

This is useful for cases where you need to preserve the type of the original exception as well as the message. All subclasses should override.

Returns

a new Exception instance that is a copy of this Exception object.

Reimplemented from **decaf::io::IOException** (p. 1547).

The documentation for this class was generated from the following file:

- src/main/decaf/net/**UnknownServiceException.h**

6.575 decaf::io::UnsupportedEncodingException Class Reference

Thrown when the the Character Encoding is not supported.

```
#include <src/main/decaf/io/UnsupportedEncodingException.h>
```

Inheritance diagram for decaf::io::UnsupportedEncodingException:

Public Member Functions

- **UnsupportedEncodingException** () throw ()
Default Constructor.
- **UnsupportedEncodingException** (const lang::Exception &ex) throw ()
Copy Constructor.
- **UnsupportedEncodingException** (const **UnsupportedEncodingException** &ex) throw ()
Copy Constructor.
- **UnsupportedEncodingException** (const char *file, const int lineNumber, const std::exception *cause, const char *msg,...) throw ()
Constructor - Initializes the file name and line number where this message occurred.
- **UnsupportedEncodingException** (const std::exception *cause) throw ()
Constructor.
- **UnsupportedEncodingException** (const char *file, const int lineNumber, const char *msg,...) throw ()
Constructor.
- virtual **UnsupportedEncodingException** * clone () const
Clones this exception.
- virtual ~**UnsupportedEncodingException** () throw ()

6.575.1 Detailed Description

Thrown when the the Character Encoding is not supported.

Since

1.0

6.575.2 Constructor & Destructor Documentation

6.575.2.1 **decaf::io::UnsupportedEncodingException::UnsupportedEncodingException** () throw () `[inline]`

Default Constructor.

6.575.2.2 **decaf::io::UnsupportedEncodingException::UnsupportedEncodingException** (const lang::Exception & ex) throw () `[inline]`

Copy Constructor.

Parameters

<i>ex</i>	the exception to copy
-----------	-----------------------

6.575.2.3 decaf::io::UnsupportedEncodingException::UnsupportedEncodingException (const UnsupportedEncodingException & *ex*) throw ()
[inline]

Copy Constructor.

Parameters

<i>ex</i>	the exception to copy, which is an instance of this type
-----------	--

6.575.2.4 decaf::io::UnsupportedEncodingException::UnsupportedEncodingException (const char * *file*, const int *lineNumber*, const std::exception * *cause*, const char * *msg*, ...) throw () *[inline]*

Constructor - Initializes the file name and line number where this message occurred.

Sets the message to report, using an optional list of arguments to parse into the message

Parameters

<i>file</i>	The file name where exception occurs
<i>lineNumber</i>	The line number where the exception occurred.
<i>cause</i>	The exception that was the cause for this one to be thrown.
<i>msg</i>	The message to report
...	list of primitives that are formatted into the message

6.575.2.5 decaf::io::UnsupportedEncodingException::UnsupportedEncodingException (const std::exception * *cause*) throw ()
[inline]

Constructor.

Parameters

<i>cause</i>	Pointer to the exception that caused this one to be thrown, the object is cloned caller retains ownership.
--------------	--

6.575.2.6 **decaf::io::UnsupportedEncodingException::UnsupportedEncodingException** (const char * *file*, const int *lineNumber*, const char * *msg*, ...) throw ()
[inline]

Constructor.

Parameters

<i>file</i>	The file name where exception occurs
<i>lineNumber</i>	The line number where the exception occurred.
<i>msg</i>	The message to report
...	list of primitives that are formatted into the message

6.575.2.7 **virtual decaf::io::UnsupportedEncodingException::~~UnsupportedEncodingException** () throw () [inline, virtual]

6.575.3 Member Function Documentation

6.575.3.1 **virtual UnsupportedOperationException* decaf::io::UnsupportedEncodingException::clone** () const [inline, virtual]

Clones this exception.

This is useful for cases where you need to preserve the type of the original exception as well as the message. All subclasses should override.

Returns

A new instance of an Exception object that is a copy of this instance.

Reimplemented from **decaf::io::IOException** (p. 1547).

The documentation for this class was generated from the following file:

- src/main/decaf/io/**UnsupportedEncodingException.h**

6.576 decaf::lang::exceptions::UnsupportedOperationException Class Reference

```
#include <src/main/decaf/lang/exceptions/Unsupported-  
OperationException.h>
```

Inheritance diagram for decaf::lang::exceptions::UnsupportedOperationException:

Public Member Functions

- **UnsupportedOperationException** () throw ()
Default Constructor.
- **UnsupportedOperationException** (const **Exception** &ex) throw ()
*Conversion Constructor from some other **Exception** (p. 1279).*
- **UnsupportedOperationException** (const **UnsupportedOperationException** &ex) throw ()
Copy Constructor.
- **UnsupportedOperationException** (const char *file, const int lineNumber, const std::exception *cause, const char *msg,...) throw ()
Constructor - Initializes the file name and line number where this message occurred.
- **UnsupportedOperationException** (const std::exception *cause) throw ()
Constructor.
- **UnsupportedOperationException** (const char *file, const int lineNumber, const char *msg,...) throw ()
Constructor - Initializes the file name and line number where this message occurred.
- virtual **UnsupportedOperationException** * clone () const
Clones this exception.
- virtual ~**UnsupportedOperationException** () throw ()

6.576.1 Constructor & Destructor Documentation

6.576.1.1 **decaf::lang::exceptions::UnsupportedOperationException**
Exception::UnsupportedOperationException () throw ()
 [inline]

Default Constructor.

6.576.1.2 **decaf::lang::exceptions::UnsupportedOperationException::-**
UnsupportedOperationException (const Exception & ex) throw ()
 [inline]

Conversion Constructor from some other **Exception** (p. 1279).

Parameters

ex	An exception that should become this type of Exception (p. 1279)
-----------	---

6.576.1.3 **decaf::lang::exceptions::UnsupportedOperationException**
Exception::UnsupportedOperationException (const
UnsupportedOperationException & ex) throw () [inline]

Copy Constructor.

Parameters

<i>ex</i>	An exception that should become this type of Exception (p. 1279)
-----------	---

6.576.1.4 **decaf::lang::exceptions::UnsupportedOperationException::-**
UnsupportedOperationException (*const char * file*, *const int lineNumber*,
*const std::exception * cause*, *const char * msg*, ...) throw () [inline]

Constructor - Initializes the file name and line number where this message occurred.

Sets the message to report, using an optional list of arguments to parse into the message

Parameters

<i>file</i>	The file name where exception occurs
<i>lineNumber</i>	The line number where the exception occurred.
<i>cause</i>	The exception that was the cause for this one to be thrown.
<i>msg</i>	The message to report
...	list of primitives that are formatted into the message

6.576.1.5 **decaf::lang::exceptions::UnsupportedOperationException::-**
UnsupportedOperationException (*const std::exception * cause*) throw ()
[inline]

Constructor.

Parameters

<i>cause</i>	Pointer (p. 2083) to the exception that caused this one to be thrown, the object is cloned caller retains ownership.
--------------	---

6.576.1.6 **decaf::lang::exceptions::UnsupportedOperationException::-**
UnsupportedOperationException (*const char * file*, *const int lineNumber*,
*const char * msg*, ...) throw () [inline]

Constructor - Initializes the file name and line number where this message occurred.

Sets the message to report, using an optional list of arguments to parse into the message

Parameters

<i>file</i>	The file name where exception occurs
<i>lineNumber</i>	The line number where the exception occurred.
<i>msg</i>	The message to report
...	list of primitives that are formatted into the message

6.576.1.7 `virtual decaf::lang::exceptions::UnsupportedOperationException-
::~UnsupportedOperationException () throw () [inline,
virtual]`

6.576.2 Member Function Documentation

6.576.2.1 `virtual UnsupportedOperationException* decaf::lang::exceptions-
::UnsupportedOperationException::clone () const [inline,
virtual]`

Clones this exception.

This is useful for cases where you need to preserve the type of the original exception as well as the message. All subclasses should override.

Returns

an new **Exception** (p. 1279) instance that is a copy of this one.

Reimplemented from **decaf::lang::Exception** (p. 1282).

Reimplemented in **decaf::nio::ReadOnlyBufferException** (p. 2246).

The documentation for this class was generated from the following file:

- `src/main/decaf/lang/exceptions/UnsupportedOperationException.h`

6.577 cms::UnsupportedOperationException Class Reference

This exception must be thrown when a CMS client attempts use a CMS method that is not implemented or not supported by the CMS Provider in use.

```
#include <src/main/cms/UnsupportedOperationException.h>
```

Inheritance diagram for `cms::UnsupportedOperationException`:

Public Member Functions

- **UnsupportedOperationException** ()
- **UnsupportedOperationException** (const **UnsupportedOperationException** &ex)
- **UnsupportedOperationException** (const std::string &message)
- **UnsupportedOperationException** (const std::string &message, const std::exception *cause)
- **UnsupportedOperationException** (const std::string &message, const std::exception *cause, const std::vector< std::pair< std::string, int > > &stack-Trace)
- virtual **~UnsupportedOperationException** () throw ()

6.577.1 Detailed Description

This exception must be thrown when a CMS client attempts use a CMS method that is not implemented or not supported by the CMS Provider in use.

Since

2.0

6.577.2 Constructor & Destructor Documentation

6.577.2.1 `cms::UnsupportedOperationException::UnsupportedOperationException ()`

6.577.2.2 `cms::UnsupportedOperationException::UnsupportedOperationException (const UnsupportedOperationException & ex)`

6.577.2.3 `cms::UnsupportedOperationException::UnsupportedOperationException (const std::string & message)`

6.577.2.4 `cms::UnsupportedOperationException::UnsupportedOperationException (const std::string & message, const std::exception * cause)`

6.577.2.5 `cms::UnsupportedOperationException::UnsupportedOperationException (const std::string & message, const std::exception * cause, const std::vector< std::pair< std::string, int > > & stackTrace)`

6.577.2.6 `virtual cms::UnsupportedOperationException::~~UnsupportedOperationException () throw () [virtual]`

The documentation for this class was generated from the following file:

- `src/main/cms/UnsupportedOperationException.h`

6.578 decaf::net::URI Class Reference

This class represents an instance of a **URI** (p. 2828) as defined by RFC 2396.

```
#include <src/main/decaf/net/URI.h>
```

Inheritance diagram for decaf::net::URI:

Public Member Functions

- **URI** ()
Default Constructor, same as calling a Constructor with all fields empty.
- **URI** (const **URI** &uri)
*Constructs a **URI** (p. 2828) as a copy of another **URI** (p. 2828).*
- **URI** (const std::string &uri)
*Constructs a **URI** (p. 2828) from the given string.*
- **URI** (const std::string &scheme, const std::string &ssp, const std::string &fragment)
*Constructs a **URI** (p. 2828) from the given components.*
- **URI** (const std::string &scheme, const std::string &userInfo, const std::string &host, int port, const std::string &path, const std::string &query, const std::string &fragment)
*Constructs a **URI** (p. 2828) from the given components.*
- **URI** (const std::string &scheme, const std::string &host, const std::string &path, const std::string &fragment)
*Constructs a **URI** (p. 2828) from the given components.*
- **URI** (const std::string &scheme, const std::string &authority, const std::string &path, const std::string &query, const std::string &fragment)
*Constructs a **URI** (p. 2828) from the given components.*
- virtual ~**URI** ()
- virtual int **compareTo** (const **URI** &value) const
Compares this object with the specified object for order.
- virtual bool **equals** (const **URI** &value) const
- virtual bool **operator==** (const **URI** &value) const
Compares equality between this object and the one passed.
- virtual bool **operator<** (const **URI** &value) const
Compares this object to another and returns true if this object is considered to be less than the one passed.
- std::string **getAuthority** () const
- std::string **getFragment** () const
- std::string **getHost** () const
- std::string **getPath** () const
- int **getPort** () const
- std::string **getQuery** () const
- std::string **getScheme** () const
- std::string **getUserInfo** () const
- std::string **getRawAuthority** () const
*Returns the raw authority component of this **URI** (p. 2828).*
- std::string **getRawFragment** () const
*Returns the raw fragment component of this **URI** (p. 2828).*
- std::string **getRawPath** () const
*Returns the raw path component of this **URI** (p. 2828).*
- std::string **getRawQuery** () const

- Returns the raw query component of this **URI** (p. 2828).*
- `std::string getRawSchemeSpecificPart ()` const
- Returns the raw scheme-specific part of this **URI** (p. 2828).*
- `std::string getSchemeSpecificPart ()` const
- Returns the decoded scheme-specific part of this **URI** (p. 2828).*
- `std::string getRawUserInfo ()` const
- Returns the raw user-information component of this **URI** (p. 2828).*
- `bool isAbsolute ()` const
- Tells whether or not this **URI** (p. 2828) is absolute.*
- `bool isOpaque ()` const
- Tells whether or not this **URI** (p. 2828) is opaque.*
- `URI normalize ()` const
- Normalizes this **URI** (p. 2828)'s path.*
- `URI parseServerAuthority ()` const
- Attempts to parse this **URI** (p. 2828)'s authority component, if defined, into user-information, host, and port components.*
- `URI relativize (const URI &uri)` const
- Relativizes the given **URI** (p. 2828) against this **URI** (p. 2828).*
- `URI resolve (const std::string &str)` const
- Constructs a new **URI** (p. 2828) by parsing the given string and then resolving it against this **URI** (p. 2828).*
- `URI resolve (const URI &uri)` const
- Resolves the given **URI** (p. 2828) against this **URI** (p. 2828).*
- `std::string toString ()` const
- Returns the content of this **URI** (p. 2828) as a string.*
- `URL toURL ()` const
- Constructs a **URL** (p. 2868) from this **URI** (p. 2828).*

Static Public Member Functions

- `static URI create (const std::string uri)`
- Creates a **URI** (p. 2828) by parsing the given string.*

6.578.1 Detailed Description

This class represents an instance of a **URI** (p. 2828) as defined by RFC 2396.

6.578.2 Constructor & Destructor Documentation

6.578.2.1 decaf::net::URI::URI ()

Default Constructor, same as calling a Constructor with all fields empty.

6.578.2.2 `decaf::net::URI::URI (const URI & uri)`

Constructs a **URI** (p. 2828) as a copy of another **URI** (p. 2828).

Parameters

<i>uri</i>	- uri to copy
------------	---------------

Exceptions

URISyntax-Exception (p. 2856)	if the URI (p. 2828) passed is malformed.
---	--

6.578.2.3 `decaf::net::URI::URI (const std::string & uri)`

Constructs a **URI** (p. 2828) from the given string.

Parameters

<i>uri</i>	- string uri to parse.
------------	------------------------

Exceptions

URISyntax-Exception (p. 2856)	if the URI (p. 2828) passed is malformed.
---	--

6.578.2.4 `decaf::net::URI::URI (const std::string & scheme, const std::string & ssp, const std::string & fragment)`

Constructs a **URI** (p. 2828) from the given components.

Parameters

<i>scheme</i>	- the uri scheme
<i>ssp</i>	- Scheme specific part
<i>fragment</i>	- Fragment

Exceptions

URISyntax-Exception (p. 2856)	if the URI (p. 2828) passed is malformed.
---	--

6.578.2.5 `decaf::net::URI::URI (const std::string & scheme, const std::string & userInfo,
const std::string & host, int port, const std::string & path, const std::string & query,
const std::string & fragment)`

Constructs a **URI** (p. 2828) from the given components.

Parameters

<i>scheme</i>	- Scheme name
<i>userInfo</i>	- User name and authorization information
<i>host</i>	- Host name
<i>port</i>	- Port number
<i>path</i>	- Path
<i>query</i>	- Query
<i>fragment</i>	- Fragment

Exceptions

URISyntax-Exception (p. 2856)	if the URI (p. 2828) passed is malformed.
---	--

6.578.2.6 `decaf::net::URI::URI (const std::string & scheme, const std::string & host, const
std::string & path, const std::string & fragment)`

Constructs a **URI** (p. 2828) from the given components.

Parameters

<i>scheme</i>	- Scheme name
<i>host</i>	- Host name
<i>path</i>	- Path
<i>fragment</i>	- Fragment

Exceptions

URISyntax-Exception (p. 2856)	if the URI (p. 2828) passed is malformed.
---	--

6.578.2.7 `decaf::net::URI::URI (const std::string & scheme, const std::string & authority,
const std::string & path, const std::string & query, const std::string & fragment)`

Constructs a **URI** (p. 2828) from the given components.

Parameters

<i>scheme</i>	- Scheme name
<i>authority</i>	- Authority
<i>path</i>	- Path
<i>query</i>	- Query
<i>fragment</i>	- Fragment

Exceptions

URISyntaxException (p. 2856)	if the URI (p. 2828) passed is malformed.
--	--

6.578.2.8 `virtual decaf::net::URI::~~URI () [inline, virtual]`

6.578.3 Member Function Documentation

6.578.3.1 `virtual int decaf::net::URI::compareTo (const URI & value) const [virtual]`

Compares this object with the specified object for order.

Returns a negative integer, zero, or a positive integer as this object is less than, equal to, or greater than the specified object.

Parameters

<i>value</i>	- the value to compare to this one.
--------------	-------------------------------------

Returns

zero if equal minus one if less than and one if greater than.

6.578.3.2 `static URI decaf::net::URI::create (const std::string uri) [static]`

Creates a **URI** (p. 2828) by parsing the given string.

This convenience factory method works as if by invoking the `URI(string)` constructor; any **URISyntaxException** (p. 2856) thrown by the constructor is caught and wrapped in a new `IllegalArgumentException` object, which is then thrown.

Parameters

<i>uri</i>	- URI (p. 2828) string to parse
------------	--

Exceptions

<i>IllegalArgument-Exception</i>	
----------------------------------	--

6.578.3.3 virtual bool **decaf::net::URI::equals** (const **URI** & *value*) const [virtual]

Returns

true if this value is considered equal to the passed value.

6.578.3.4 std::string **decaf::net::URI::getAuthority** () const

Returns

the decoded authority component of this **URI** (p. 2828).

6.578.3.5 std::string **decaf::net::URI::getFragment** () const

Returns

the decoded fragment component of this **URI** (p. 2828).

6.578.3.6 std::string **decaf::net::URI::getHost** () const

Returns

the host component of this **URI** (p. 2828).

6.578.3.7 std::string **decaf::net::URI::getPath** () const

Returns

the path component of this **URI** (p. 2828).

6.578.3.8 int **decaf::net::URI::getPort** () const

Returns

the port component of this **URI** (p. 2828).

6.578.3.9 std::string **decaf::net::URI::getQuery** () const

Returns

the query component of this **URI** (p. 2828).

6.578.3.10 `std::string decaf::net::URI::getRawAuthority () const`

Returns the raw authority component of this **URI** (p. 2828).

The authority component of a **URI** (p. 2828), if defined, only contains the commercial-at character ('@') and characters in the unreserved, punct, escaped, and other categories. If the authority is server-based then it is further constrained to have valid user-information, host, and port components.

Returns

the raw authority component of the **URI** (p. 2828)

6.578.3.11 `std::string decaf::net::URI::getRawFragment () const`

Returns the raw fragment component of this **URI** (p. 2828).

The fragment component of a **URI** (p. 2828), if defined, only contains legal **URI** (p. 2828) characters.

Returns

the raw fragment component of this **URI** (p. 2828)

6.578.3.12 `std::string decaf::net::URI::getRawPath () const`

Returns the raw path component of this **URI** (p. 2828).

The path component of a **URI** (p. 2828), if defined, only contains the slash character ('/'), the commercial-at character ('@'), and characters in the unreserved, punct, escaped, and other categories.

Returns

the raw path component of this **URI** (p. 2828)

6.578.3.13 `std::string decaf::net::URI::getRawQuery () const`

Returns the raw query component of this **URI** (p. 2828).

The query component of a **URI** (p. 2828), if defined, only contains legal **URI** (p. 2828) characters.

Returns

the raw query component of the **URI** (p. 2828).

6.578.3.14 `std::string decaf::net::URI::getRawSchemeSpecificPart () const`

Returns the raw scheme-specific part of this **URI** (p. 2828).

The scheme-specific part is never undefined, though it may be empty. The scheme-specific part of a **URI** (p. 2828) only contains legal **URI** (p. 2828) characters.

Returns

the raw scheme special part of the uri

6.578.3.15 `std::string decaf::net::URI::getRawUserInfo () const`

Returns the raw user-information component of this **URI** (p. 2828).

The user-information component of a **URI** (p. 2828), if defined, only contains characters in the unreserved, punct, escaped, and other categories.

Returns

the raw user-information component of the **URI** (p. 2828)

6.578.3.16 `std::string decaf::net::URI::getScheme () const`**Returns**

the scheme component of this **URI** (p. 2828)

6.578.3.17 `std::string decaf::net::URI::getSchemeSpecificPart () const`

Returns the decoded scheme-specific part of this **URI** (p. 2828).

The string returned by this method is equal to that returned by the `getRawSchemeSpecificPart` method except that all sequences of escaped octets are decoded.

Returns

the raw scheme specific part of the uri.

6.578.3.18 `std::string decaf::net::URI::getUserInfo () const`**Returns**

the user info component of this **URI** (p. 2828)

6.578.3.19 `bool decaf::net::URI::isAbsolute () const`

Tells whether or not this **URI** (p. 2828) is absolute.

A **URI** (p. 2828) is absolute if, and only if, it has a scheme component.

Returns

true if, and only if, this **URI** (p. 2828) is absolute

6.578.3.20 `bool decaf::net::URI::isOpaque () const`

Tells whether or not this **URI** (p. 2828) is opaque.

A **URI** (p. 2828) is opaque if, and only if, it is absolute and its scheme-specific part does not begin with a slash character ('/'). An opaque **URI** (p. 2828) has a scheme, a scheme-specific part, and possibly a fragment; all other components are undefined.

Returns

true if, and only if, this **URI** (p. 2828) is opaque

6.578.3.21 `URI decaf::net::URI::normalize () const`

Normalizes this **URI** (p. 2828)'s path.

If this **URI** (p. 2828) is opaque, or if its path is already in normal form, then this **URI** (p. 2828) is returned. Otherwise a new **URI** (p. 2828) is constructed that is identical to this **URI** (p. 2828) except that its path is computed by normalizing this **URI** (p. 2828)'s path in a manner consistent with RFC 2396, section 5.2, step 6, sub-steps c through f; that is:

1. All "." segments are removed.
2. If a ".." segment is preceded by a non-".." segment then both of these segments are removed. This step is repeated until it is no longer applicable.
3. If the path is relative, and if its first segment contains a colon character (':'), then a "." segment is prepended. This prevents a relative **URI** (p. 2828) with a path such as "a:b/c/d" from later being re-parsed as an opaque **URI** (p. 2828) with a scheme of "a" and a scheme-specific part of "b/c/d". (Deviation from RFC 2396)

A normalized path will begin with one or more "." segments if there were insufficient non-".." segments preceding them to allow their removal. A normalized path will begin with a "." segment if one was inserted by step 3 above. Otherwise, a normalized path will not contain any "." or ".." segments.

Returns

A **URI** (p. 2828) equivalent to this **URI** (p. 2828), but whose path is in normal form

6.578.3.22 virtual bool decaf::net::URI::operator< (const **URI** & *value*) const [virtual]

Compares this object to another and returns true if this object is considered to be less than the one passed.

This

Parameters

<i>value</i>	- the value to be compared to this one.
--------------	---

Returns

true if this object is equal to the one passed.

6.578.3.23 virtual bool decaf::net::URI::operator== (const **URI** & *value*) const [virtual]

Compares equality between this object and the one passed.

Parameters

<i>value</i>	- the value to be compared to this one.
--------------	---

Returns

true if this object is equal to the one passed.

6.578.3.24 **URI** decaf::net::URI::parseServerAuthority () const

Attempts to parse this **URI** (p.2828)'s authority component, if defined, into user-information, host, and port components.

If this **URI** (p.2828)'s authority component has already been recognized as being server-based then it will already have been parsed into user-information, host, and port components. In this case, or if this **URI** (p.2828) has no authority component, this method simply returns this **URI** (p.2828).

Otherwise this method attempts once more to parse the authority component into user-information, host, and port components, and throws an exception describing why the authority component could not be parsed in that way.

Returns

A **URI** (p.2828) whose authority field has been parsed as a server-based authority

Exceptions

URISyntax-Exception (p.2856)	- If the authority component of this URI (p.2828) is defined but cannot be parsed as a server-based authority.
--	---

6.578.3.25 URI decaf::net::URI::relativize (const URI & uri) const

Relativizes the given **URI** (p. 2828) against this **URI** (p. 2828).

The relativization of the given **URI** (p. 2828) against this **URI** (p. 2828) is computed as follows:

1. If either this **URI** (p. 2828) or the given **URI** (p. 2828) are opaque, or if the scheme and authority components of the two URIs are not identical, or if the path of this **URI** (p. 2828) is not a prefix of the path of the given **URI** (p. 2828), then the given **URI** (p. 2828) is returned.
2. Otherwise a new relative hierarchical **URI** (p. 2828) is constructed with query and fragment components taken from the given **URI** (p. 2828) and with a path component computed by removing this **URI** (p. 2828)'s path from the beginning of the given **URI** (p. 2828)'s path.

Parameters

<i>uri</i>	- The URI (p. 2828) to be relativized against this URI (p. 2828)
------------	--

Returns

The resulting **URI** (p. 2828)

6.578.3.26 URI decaf::net::URI::resolve (const std::string & str) const

Constructs a new **URI** (p. 2828) by parsing the given string and then resolving it against this **URI** (p. 2828).

This convenience method works as if invoking it were equivalent to evaluating the expression `resolve(URI::create(str))`.

Parameters

<i>str</i>	- The string to be parsed into a URI (p. 2828)
------------	---

Returns

The resulting **URI** (p. 2828)

Exceptions

<i>IllegalArgument-Exception</i>	- If the given string violates RFC 2396
----------------------------------	---

6.578.3.27 URI decaf::net::URI::resolve (const URI & uri) const

Resolves the given **URI** (p. 2828) against this **URI** (p. 2828).

If the given **URI** (p. 2828) is already absolute, or if this **URI** (p. 2828) is opaque, then a copy of the given **URI** (p. 2828) is returned.

If the given **URI** (p. 2828)'s fragment component is defined, its path component is empty, and its scheme, authority, and query components are undefined, then a **URI** (p. 2828) with the given fragment but with all other components equal to those of this **URI** (p. 2828) is returned. This allows a **URI** (p. 2828) representing a standalone fragment reference, such as "#foo", to be usefully resolved against a base **URI** (p. 2828).

Otherwise this method constructs a new hierarchical **URI** (p. 2828) in a manner consistent with RFC 2396, section 5.2; that is:

1. A new **URI** (p. 2828) is constructed with this **URI** (p. 2828)'s scheme and the given **URI** (p. 2828)'s query and fragment components. 2. If the given **URI** (p. 2828) has an authority component then the new **URI** (p. 2828)'s authority and path are taken from the given **URI** (p. 2828). 3. Otherwise the new **URI** (p. 2828)'s authority component is copied from this **URI** (p. 2828), and its path is computed as follows:

1. If the given **URI** (p. 2828)'s path is absolute then the new **URI** (p. 2828)'s path is taken from the given **URI** (p. 2828). 2. Otherwise the given **URI** (p. 2828)'s path is relative, and so the new **URI** (p. 2828)'s path is computed by resolving the path of the given **URI** (p. 2828) against the path of this **URI** (p. 2828). This is done by concatenating all but the last segment of this **URI** (p. 2828)'s path, if any, with the given **URI** (p. 2828)'s path and then normalizing the result as if by invoking the `normalize` method.

The result of this method is absolute if, and only if, either this **URI** (p. 2828) is absolute or the given **URI** (p. 2828) is absolute.

Parameters

<i>uri</i>	- The URI (p. 2828) to be resolved against this URI (p. 2828)
------------	---

Returns

The resulting **URI** (p. 2828)

6.578.3.28 std::string decaf::net::URI::toString () const

Returns the content of this **URI** (p. 2828) as a string.

If this **URI** (p. 2828) was created by invoking one of the constructors in this class then a string equivalent to the original input string, or to the string computed from the originally-given components, as appropriate, is returned. Otherwise this **URI** (p. 2828) was created by normalization, resolution, or relativization, and so a string is constructed from this **URI** (p. 2828)'s components according to the rules specified in RFC 2396, section 5.2, step 7.

Returns

the string form of this **URI** (p. 2828)

6.578.3.29 URL decaf::net::URI::toURL () const

Constructs a **URL** (p. 2868) from this **URI** (p. 2828).

This convenience method works as if invoking it were equivalent to evaluating the expression `new URL (p. 2868)(this.toString())` after first checking that this **URI** (p. 2828) is absolute.

Returns

A **URL** (p. 2868) constructed from this **URI** (p. 2828)

Exceptions

<i>IllegalArgument-Exception</i>	- If this URL (p. 2868) is not absolute
<i>MalformedURL-Exception</i> (p. 1766)	- If a protocol handler for the URL (p. 2868) could not be found, or if some other error occurred while constructing the URL (p. 2868)

The documentation for this class was generated from the following file:

- `src/main/decaf/net/URI.h`

6.579 decaf::internal::net::URIEncoderDecoder Class Reference

```
#include <src/main/decaf/internal/net/URIEncoderDecoder.h>
```

Public Member Functions

- **URIEncoderDecoder** ()
- virtual **~URIEncoderDecoder** ()

Static Public Member Functions

- static void **validate** (const std::string &s, const std::string &legal)
Validate a string by checking if it contains any characters other than:
- static void **validateSimple** (const std::string &s, const std::string &legal)
Validate a string by checking if it contains any characters other than:
- static std::string **quoteIllegal** (const std::string &s, const std::string &legal)
All characters except letters ('a'..'z', 'A'..'Z') and numbers ('0'..'9') and legal characters are converted into their hexadecimal value prepended by ".
- static std::string **encodeOthers** (const std::string &s)
Other characters, which are chars that are not US-ASCII, and are not ISO Control or are not ISO Space chars are not preserved.

- static std::string **decode** (const std::string &s)

Decodes the string argument which is assumed to be encoded in the x-www-form-urlencoded MIME content type using the UTF-8 encoding scheme.

6.579.1 Constructor & Destructor Documentation

6.579.1.1 **decaf::internal::net::URLEncoderDecoder::URLEncoderDecoder** ()

6.579.1.2 **virtual decaf::internal::net::URLEncoderDecoder::~~URLEncoderDecoder** () [inline, virtual]

6.579.2 Member Function Documentation

6.579.2.1 **static std::string decaf::internal::net::URLEncoderDecoder::decode** (const std::string & s) [static]

Decodes the string argument which is assumed to be encoded in the x-www-form-urlencoded MIME content type using the UTF-8 encoding scheme.

" and two following hex digit characters are converted to the equivalent byte value. All other characters are passed through unmodified.

e.g. "A%20B%20C %24%25" -> "A B C \$%"

Parameters

s	- The encoded string.
---	-----------------------

Returns

The decoded version.

6.579.2.2 **static std::string decaf::internal::net::URLEncoderDecoder::encodeOthers** (const std::string & s) [static]

Other characters, which are chars that are not US-ASCII, and are not ISO Control or are not ISO Space chars are not preserved.

They are converted into their hexadecimal value prepended by ".

For example: Euro currency symbol -> "%E2%82%AC".

Parameters

s	- the string to be converted
---	------------------------------

Returns

the converted string

6.579.2.3 `static std::string decaf::internal::net::URLEncoderDecoder::quotelllegal (const std::string & s, const std::string & legal) [static]`

All characters except letters ('a'..'z', 'A'..'Z') and numbers ('0'..'9') and legal characters are converted into their hexadecimal value prepended by "%".

For example: '#' -> "%23"

Other characters, which are chars that are not US-ASCII, and are not ISO Control or are not ISO Space chars, are preserved.

Parameters

<i>s</i>	- the string to be converted
<i>legal</i>	- the characters allowed to be preserved in the string s

Returns

converted string

6.579.2.4 `static void decaf::internal::net::URLEncoderDecoder::validate (const std::string & s, const std::string & legal) [static]`

Validate a string by checking if it contains any characters other than:

1. letters ('a'..'z', 'A'..'Z')
2. numbers ('0'..'9')
3. characters in the legalset parameter
4. characters that are not ISO Control or are not ISO Space characters)

Parameters

<i>s</i>	- the string to be validated
<i>legal</i>	- the characters allowed in the string s

Exceptions

<i>URISyntaxException</i>	if the uri string is not well formed.
---------------------------	---------------------------------------

6.579.2.5 `static void decaf::internal::net::URLEncoderDecoder::validateSimple (const std::string & s, const std::string & legal) [static]`

Validate a string by checking if it contains any characters other than:

1. letters ('a'..'z', 'A'..'Z')
2. numbers ('0'..'9')
3. characters in the legalset parameter

Parameters

<i>s</i>	- the string to be validated
<i>legal</i>	- the characters allowed in the string s

Exceptions

<code>URISyntaxException</code>	if the uri string is not well formed.
---------------------------------	---------------------------------------

The documentation for this class was generated from the following file:

- `src/main/decaf/internal/net/URIEncoderDecoder.h`

6.580 decaf::internal::net::URIHelper Class Reference

Helper class used by the URI classes in encoding and decoding of URI's.

```
#include <src/main/decaf/internal/net/URIHelper.h>
```

Public Member Functions

- **URIHelper** (const std::string &unreserved, const std::string &punct, const std::string &reserved, const std::string &someLegal, const std::string &allLegal)
*Setup the **URIHelper** (p. 2844) with values assigned to the various fields that are used in the validation process.*
- **URIHelper** ()
Sets up the filter strings with sane defaults.
- virtual ~**URIHelper** ()
- **URIType parseURI** (const std::string &uri, bool forceServer)
Parse the passed in URI.
- void **validateScheme** (const std::string &uri, const std::string &scheme, int index)
Validate the schema portin of the URI.
- void **validateSsp** (const std::string &uri, const std::string &ssp, std::size_t index)
Validate that the URI Ssp Segment contains no invalid encodings.
- void **validateAuthority** (const std::string &uri, const std::string &authority, std::size_t index)
Validate that the URI Authority Segment contains no invalid encodings.
- void **validatePath** (const std::string &uri, const std::string &path, std::size_t index)
Validate that the URI Path Segment contains no invalid encodings.
- void **validateQuery** (const std::string &uri, const std::string &query, std::size_t index)
Validate that the URI Query Segment contains no invalid encodings.
- void **validateFragment** (const std::string &uri, const std::string &fragment, std::size_t index)
Validate that the URI fragment contains no invalid encodings.
- **URIType parseAuthority** (bool forceServer, const std::string &authority)

determine the host, port and user-info if the authority parses successfully to a server based authority

- void **validateUserinfo** (const std::string &uri, const std::string &userinfo, std::size_t index)

Check the supplied user info for validity.

- bool **isValidHost** (bool forceServer, const std::string &host)

distinguish between IPv4, IPv6, domain name and validate it based on its type

- bool **isValidDomainName** (const std::string &host)

Validates the string past to determine if it is a well formed domain name.

- bool **isValidIPv4Address** (const std::string &host)

Validate if the host value is a well formed IPv4 address, this is the form XXX.XXX.XX-X.XXX were X is any number 0-9.

- bool **isValidIPv6Address** (const std::string &ipAddress)

Determines if the given address is valid according to the IPv6 spec.

- bool **isValidIPv4Word** (const std::string &word)

Check is the string passed contains a Valid IPv4 word, which is an integer in the range of 0 to 255.

- bool **isValidHexChar** (char c)

Determines if the given char is a valid Hex char.

6.580.1 Detailed Description

Helper class used by the URI classes in encoding and decoding of URI's.

6.580.2 Constructor & Destructor Documentation

- 6.580.2.1 **decaf::internal::net::URIHelper::URIHelper** (const std::string & *unreserved*, const std::string & *punct*, const std::string & *reserved*, const std::string & *someLegal*, const std::string & *allLegal*)

Setup the **URIHelper** (p. 2844) with values assigned to the various fields that are used in the validation process.

The defaults are overridden by these values.

Parameters

<i>unreserved</i>	- characters not reserved for use.
<i>punct</i>	- allowable punctuation symbols.
<i>reserved</i>	- characters not allowed for general use in the URI.
<i>someLegal</i>	- characters that are legal in certain cases.
<i>allLegal</i>	- characters that are always legal.

6.580.2.2 decaf::internal::net::URIHelper::URIHelper ()

Sets up the filter strings with sane defaults.

6.580.2.3 virtual decaf::internal::net::URIHelper::~~URIHelper () [inline, virtual]

6.580.3 Member Function Documentation

6.580.3.1 bool decaf::internal::net::URIHelper::isValidDomainName (const std::string & *host*)

Validates the string past to determine if it is a well formed domain name.

Parameters

<i>host</i>	- domain name to validate.
-------------	----------------------------

Returns

true if host is well formed.

6.580.3.2 bool decaf::internal::net::URIHelper::isValidHexChar (char *c*)

Determines if the given char is a valid Hex char.

Valid chars are A-F (upper or lower case) and 0-9.

Parameters

<i>c</i>	- char to inspect
----------	-------------------

Returns

true if *c* is a valid hex char.

6.580.3.3 bool decaf::internal::net::URIHelper::isValidHost (bool *forceServer*, const std::string & *host*)

distinguish between IPv4, IPv6, domain name and validate it based on its type

Parameters

<i>forceServer</i>	- true if the forceServer mode should be active.
<i>host</i>	- Host string to validate.

Returns

true if the host value if a valid domain name.

Exceptions

<i>URISyntaxException</i>	if the host is invalid and forceServer is true.
---------------------------	---

6.580.3.4 bool decaf::internal::net::URIHelper::isValidIPv4Word (const std::string & word)

Check is the string passed contains a Valid IPv4 word, which is an integer in the range of 0 to 255.

Parameters

<i>word</i>	- string value to check.
-------------	--------------------------

Returns

true if the word is a valid IPv4 word.

6.580.3.5 bool decaf::internal::net::URIHelper::isValidIPv6Address (const std::string & ipAddress)

Determines if the given address is valid according to the IPv6 spec.

Parameters

<i>ipAddress</i>	- string ip address value to validate.
------------------	--

Returns

true if the address string is valid.

6.580.3.6 bool decaf::internal::net::URIHelper::isValidIPv4Address (const std::string & host)

Validate if the host value is a well formed IPv4 address, this is the form XXX.XXX.XXX.-XXX were X is any number 0-9.

and XXX is not greater than 255.

Parameters

<i>host</i>	- IPv4 address string to parse.
-------------	---------------------------------

Returns

true if host is a well formed IPv4 address.

**6.580.3.7 URIType decaf::internal::net::URIHelper::parseAuthority (bool
forceServer, const std::string & authority)**

determine the host, port and user-info if the authority parses successfully to a server based authority

behavior in error cases: if forceServer is true, throw URISyntaxException with the proper diagnostic messages. if forceServer is false assume this is a registry based uri, and just return leaving the host, port and user-info fields undefined.

and there are some error cases where URISyntaxException is thrown regardless of the forceServer parameter e.g. mal-formed ipv6 address

Parameters

<i>forceServer</i>	
<i>authority</i>	

Returns

a **URIType** (p. 2860) instance containing the parsed data.

Exceptions

<i>URISyntaxException</i>	
---------------------------	--

**6.580.3.8 URIType decaf::internal::net::URIHelper::parseURI (const std::string & uri,
bool forceServer)**

Parse the passed in URI.

Parameters

<i>uri</i>	- the URI to Parse
<i>forceServer</i>	- if true invalid URI data throws an Exception

Returns

a **URIType** (p. 2860) instance containing the parsed data.

Exceptions

<i>URISyntaxException</i>	if forceServer is true and the URI is invalid.
---------------------------	--

6.580.3.9 void decaf::internal::net::URIHelper::validateAuthority (const std::string & *uri*, const std::string & *authority*, std::size_t *index*)

Validate that the URI Authority Segment contains no invalid encodings.

Parameters

<i>uri</i>	- the full uri.
<i>authority</i>	- the Authority to check.
<i>index</i>	- position in the uri where Authority starts.

Exceptions

<i>URISyntaxException</i>	if the fragment has errors.
---------------------------	-----------------------------

6.580.3.10 void decaf::internal::net::URIHelper::validateFragment (const std::string & *uri*, const std::string & *fragment*, std::size_t *index*)

Validate that the URI fragment contains no invalid encodings.

Parameters

<i>uri</i>	- the full uri.
<i>fragment</i>	- the fragment to check.
<i>index</i>	- position in the uri where fragment starts.

Exceptions

<i>URISyntaxException</i>	if the fragment has errors.
---------------------------	-----------------------------

6.580.3.11 void decaf::internal::net::URIHelper::validatePath (const std::string & *uri*, const std::string & *path*, std::size_t *index*)

Validate that the URI Path Segment contains no invalid encodings.

Parameters

<i>uri</i>	- the full uri.
<i>path</i>	- the path to check.
<i>index</i>	- position in the uri where path starts.

Exceptions

<i>URISyntaxException</i>	if the fragment has errors.
---------------------------	-----------------------------

6.580.3.12 void decaf::internal::net::URIHelper::validateQuery (const std::string & *uri*,
const std::string & *query*, std::size_t *index*)

Validate that the URI Query Segment contains no invalid encodings.

Parameters

<i>uri</i>	- the full uri.
<i>query</i>	- the query to check.
<i>index</i>	- position in the uri where fragment starts.

Exceptions

<i>URISyntaxException</i>	if the fragment has errors.
---------------------------	-----------------------------

6.580.3.13 void decaf::internal::net::URIHelper::validateScheme (const std::string & *uri*,
const std::string & *scheme*, int *index*)

Validate the schema portin of the URI.

Parameters

<i>uri</i>	- the URI to check.
<i>scheme</i>	- the schema section of the URI.
<i>index</i>	- index in uri where schema starts.

Exceptions

<i>URISyntaxException</i>	if the fragment has errors.
---------------------------	-----------------------------

6.580.3.14 void decaf::internal::net::URIHelper::validateSsp (const std::string & *uri*,
const std::string & *ssp*, std::size_t *index*)

Validate that the URI Ssp Segment contains no invalid encodings.

Parameters

<i>uri</i>	- the full uri.
<i>ssp</i>	- the SSP to check.
<i>index</i>	- position in the uri where Ssp starts.

Exceptions

<i>URISyntaxException</i>	if the fragment has errors.
---------------------------	-----------------------------

6.580.3.15 `void decaf::internal::net::URIHelper::validateUserinfo (const std::string & uri, const std::string & userinfo, std::size_t index)`

Check the supplied user info for validity.

Parameters

<i>uri</i>	- the uri to parse.
<i>userinfo</i>	- supplied user info
<i>index</i>	- index into the URI string where the data is located.

Returns

true if valid

Exceptions

<i>URISyntaxException</i> if an error occurs
--

The documentation for this class was generated from the following file:

- `src/main/decaf/internal/net/URIHelper.h`

6.581 activemq::transport::failover::URIPool Class Reference

```
#include <src/main/activemq/transport/failover/URIPool.h>
```

Public Member Functions

- **URIPool ()**
Create an Empty URI Pool.
- **URIPool (const decaf::util::List< URI > &uris)**
Creates a new URI Pool using the given list as the initial Free List.
- **~URIPool ()**
- **URI getURI ()**
Fetches the next available URI from the pool, if there are no more URIs free when this method is called it throws a NoSuchElementException.
- **void addURI (const URI &uri)**
Adds a URI to the free list, callers that have previously taken one using the getURI method should always return the URI when they close the resource that was connected to that URI.
- **void addURIs (const LinkedList< URI > &uris)**
Adds a List of URIs to this Pool, the method checks for duplicates already in the pool and does not add those.

- void **removeURI** (const **URI** &uri)
Remove a given URI from the Free List.
- bool **isRandomize** () const
Is the URI that is given randomly picked from the pool or is each one taken in sequence.
- void **setRandomize** (bool value)
Sets if the URI's that are taken from the pool are chosen Randomly or are taken in the order they are in the list.

6.581.1 Constructor & Destructor Documentation

6.581.1.1 activemq::transport::failover::URIPool::URIPool ()

Create an Empty URI Pool.

6.581.1.2 activemq::transport::failover::URIPool::URIPool (const decaf::util::List< URI > &uris)

Creates a new URI Pool using the given list as the initial Free List.

Parameters

<i>uris</i>	- List of URI to place in the Pool.
-------------	-------------------------------------

6.581.1.3 activemq::transport::failover::URIPool::~~URIPool ()

6.581.2 Member Function Documentation

6.581.2.1 void activemq::transport::failover::URIPool::addURI (const URI &uri)

Adds a URI to the free list, callers that have previously taken one using the `getURI` method should always return the URI when they close the resource that was connected to that URI.

Parameters

<i>uri</i>	- a URI previously taken from the pool.
------------	---

6.581.2.2 void activemq::transport::failover::URIPool::addURIs (const LinkedList< URI > &uris)

Adds a List of URIs to this Pool, the method checks for duplicates already in the pool and does not add those.

Parameters

<i>uris</i>	- List of URIs to add into the Pool.
-------------	--------------------------------------

6.581.2.3 URI `activemq::transport::failover::URIPool::getURI ()`

Fetches the next available URI from the pool, if there are no more URIs free when this method is called it throws a `NoSuchElementException`.

Receiving the exception is not an indication that a URI won't be available in the future, the caller should react accordingly.

Returns

the next free URI in the Pool.

Exceptions

<i>NoSuchElementException</i>	if there are none free currently.
-------------------------------	-----------------------------------

**6.581.2.4 `bool activemq::transport::failover::URIPool::isRandomize () const`
[inline]**

Is the URI that is given randomly picked from the pool or is each one taken in sequence.

Returns

true if URI gets are random.

6.581.2.5 `void activemq::transport::failover::URIPool::removeURI (const URI & uri)`

Remove a given URI from the Free List.

Parameters

<i>uri</i>	- the URI to find and remove from the free list
------------	---

**6.581.2.6 `void activemq::transport::failover::URIPool::setRandomize (bool value)`
[inline]**

Sets if the URI's that are taken from the pool are chosen Randomly or are taken in the order they are in the list.

Parameters

<i>value</i>	- true indicates URI gets are random.
--------------	---------------------------------------

The documentation for this class was generated from the following file:

- src/main/activemq/transport/failover/**URIPool.h**

6.582 activemq::util::URISupport Class Reference

```
#include <src/main/activemq/util/URISupport.h>
```

Static Public Member Functions

- static void **parseURL** (const std::string &**URI**, **decaf::util::Properties** &properties)
Parses the properties out of the provided Broker URI and sets them in the passed Properties Object.
- static **CompositeData** **parseComposite** (const **URI** &uri)
*Parses a Composite URI into a Composite Data instance, the Composite URI takes the for scheme://(uri1,uri2,...uriN)?param1=value1, each of the composite URIs is stored in the **CompositeData** (p. 890)'s internal list.*
- static **decaf::util::Properties** **parseQuery** (std::string query)
Parse the Query portion of a URI String and return a Simple Properties object containing the parameter names as keys, and the parameter values and values of the Properties.
- static void **parseQuery** (std::string query, **decaf::util::Properties** *properties)
Parse the Query portion of a URI String and return a Simple Properties object containing the parameter names as keys, and the parameter values and values of the Properties.
- static std::string **createQueryString** (const **Properties** &options)
Given a properties object create a string that can be appended to a URI as a valid Query string.

6.582.1 Member Function Documentation

6.582.1.1 static std::string **activemq::util::URISupport::createQueryString** (const **Properties** & *options*) `[static]`

Given a properties object create a string that can be appended to a URI as a valid Query string.

Parameters

<i>options</i>	Properties object containing key / value query values.
----------------	--

Returns

a valid URI query string.

Exceptions

<i>URISyntaxException</i>	if the string in the Properties object can't be encoded into a valid URI Query string.
---------------------------	--

6.582.1.2 static CompositeData activemq::util::URISupport::parseComposite (const URI & uri) [static]

Parses a Composite URI into a Composite Data instance, the Composite URI takes the for scheme://(uri1,uri2,...uriN)?param1=value1, each of the composite URIs is stored in the **CompositeData** (p. 890)'s internal list.

Parameters

<i>uri</i>	- The Composite URI to parse.
------------	-------------------------------

Returns

a new **CompositeData** (p. 890) object with the parsed data

Exceptions

<i>URISyntaxException</i>	if the URI is not well formed.
---------------------------	--------------------------------

6.582.1.3 static decaf::util::Properties activemq::util::URISupport::parseQuery (std::string query) [static]

Parse the Query portion of a URI String and return a Simple Properties object containing the parameter names as keys, and the parameter values and values of the Properties.

Parameters

<i>query</i>	The query string to parse and extract the encoded properties.
--------------	---

Returns

Properties object with the parsed output.

Exceptions

<i>IllegalArgumentException</i>	if the Query string is not well formed.
---------------------------------	---

6.582.1.4 static void **activemq::util::URISupport::parseQuery** (std::string *query*,
decaf::util::Properties * *properties*) [static]

Parse the Query portion of a URI String and return a Simple Properties object containing the parameter names as keys, and the parameter values and values of the Properties.

Parameters

<i>query</i>	- the query string to parse.
<i>properties</i>	- object pointer to get the parsed output.

Exceptions

<i>IllegalArgumentException</i>	if the Query string is not well formed.
---------------------------------	---

6.582.1.5 static void **activemq::util::URISupport::parseURL** (const std::string & *URI*,
decaf::util::Properties & *properties*) [static]

Parses the properties out of the provided Broker URI and sets them in the passed - Properties Object.

Parameters

<i>URI</i>	a Broker URI to parse
<i>properties</i>	a Properties object to set the parsed values in

Exceptions

<i>IllegalArgumentException</i>	if the passed URI is invalid
---------------------------------	------------------------------

The documentation for this class was generated from the following file:

- src/main/activemq/util/**URISupport.h**

6.583 decaf::net::URISyntaxException Class Reference

```
#include <src/main/decaf/net/URISyntaxException.h>
```

Inheritance diagram for decaf::net::URISyntaxException:

Public Member Functions

- **URISyntaxException** () throw ()
Default Constructor.
- **URISyntaxException** (const **Exception** &ex) throw ()
Conversion Constructor from some other Exception.
- **URISyntaxException** (const **URISyntaxException** &ex) throw ()
Copy Constructor.
- **URISyntaxException** (const char *file, const int lineNumber, const std::exception *cause, const char *msg,...) throw ()
Constructor - Initializes the file name and line number where this message occurred.
- **URISyntaxException** (const std::exception *cause) throw ()
Constructor.
- **URISyntaxException** (const char *file, const int lineNumber, const char *msg **DECAF_UNUSED**) throw ()
Constructor - Initializes the file name and line number where this message occurred.
- **URISyntaxException** (const char *file, const int lineNumber, const std::string &input, const std::string &reason) throw ()
Constructor - Initializes the file name and line number where this message occurred.
- **URISyntaxException** (const char *file, const int lineNumber, const std::string &input, const std::string &reason, int index) throw ()
Constructor - Initializes the file name and line number where this message occurred.
- virtual **URISyntaxException** * **clone** () const
Clones this exception.
- virtual ~**URISyntaxException** () throw ()
- std::string **getInput** () const
- std::string **getReason** () const
- int **getIndex** () const

6.583.1 Constructor & Destructor Documentation

6.583.1.1 **decaf::net::URISyntaxException::URISyntaxException ()** throw ()
[inline]

Default Constructor.

6.583.1.2 **decaf::net::URISyntaxException::URISyntaxException (const Exception & ex)** throw () [inline]

Conversion Constructor from some other Exception.

Parameters

ex	An exception that should become this type of Exception
----	--

6.583.1.3 `decaf::net::URISyntaxException::URISyntaxException (const URISyntaxException & ex) throw () [inline]`

Copy Constructor.

Parameters

<i>ex</i>	An exception that should become this type of Exception
-----------	--

6.583.1.4 `decaf::net::URISyntaxException::URISyntaxException (const char * file, const int lineNumber, const std::exception * cause, const char * msg, ...) throw () [inline]`

Constructor - Initializes the file name and line number where this message occurred.

Sets the message to report, using an optional list of arguments to parse into the message

Parameters

<i>file</i>	The file name where exception occurs
<i>lineNumber</i>	The line number where the exception occurred.
<i>cause</i>	The exception that was the cause for this one to be thrown.
<i>msg</i>	The message to report
<i>...</i>	list of primitives that are formatted into the message

6.583.1.5 `decaf::net::URISyntaxException::URISyntaxException (const std::exception * cause) throw () [inline]`

Constructor.

Parameters

<i>cause</i>	Pointer to the exception that caused this one to be thrown, the object is cloned caller retains ownership.
--------------	--

6.583.1.6 `decaf::net::URISyntaxException::URISyntaxException (const char * file, const int lineNumber, const char *msg DECAF_UNUSED) throw () [inline]`

Constructor - Initializes the file name and line number where this message occurred.

Sets the message to report, using an optional list of arguments to parse into the message

Parameters

<i>file</i>	The file name where exception occurs
<i>lineNumber</i>	The line number where the exception occurred.

<i>msg</i>	The message to report
...	list of primitives that are formatted into the message

6.583.1.7 `decaf::net::URISyntaxException::URISyntaxException (const char * file, const int lineNumber, const std::string & input, const std::string & reason) throw () [inline]`

Constructor - Initializes the file name and line number where this message occurred.
Sets the input string that caused the error and the reason for the error.

Parameters

<i>file</i>	The file name where exception occurs.
<i>lineNumber</i>	The line number where the exception occurred.
<i>input</i>	The URL (p. 2868) that caused the exception.
<i>reason</i>	The reason for the failure.

6.583.1.8 `decaf::net::URISyntaxException::URISyntaxException (const char * file, const int lineNumber, const std::string & input, const std::string & reason, int index) throw () [inline]`

Constructor - Initializes the file name and line number where this message occurred.
Sets the input string that caused the error and the reason for the error.

Parameters

<i>file</i>	The file name where exception occurs
<i>lineNumber</i>	The line number where the exception occurred.
<i>input</i>	The input URI (p. 2828) that caused the exception
<i>reason</i>	The reason for the failure.
<i>index</i>	The index in the URI (p. 2828) string where the error occurred.

6.583.1.9 `virtual decaf::net::URISyntaxException::~~URISyntaxException () throw () [inline, virtual]`

6.583.2 Member Function Documentation

6.583.2.1 `virtual URISyntaxException* decaf::net::URISyntaxException::clone () const [inline, virtual]`

Clones this exception.

This is useful for cases where you need to preserve the type of the original exception as well as the message. All subclasses should override.

Returns

a new Exception instance that is a copy of this Exception object.

Reimplemented from **decaf::lang::Exception** (p. 1282).

6.583.2.2 `int decaf::net::URISyntaxException::getIndex () const` `[inline]`

Returns

the index in the input string where the error occurred or -1

6.583.2.3 `std::string decaf::net::URISyntaxException::getInput () const`
`[inline]`

Returns

the Input string that cause this exception or ""

6.583.2.4 `std::string decaf::net::URISyntaxException::getReason () const`
`[inline]`

Returns

the Reason given for this failure, or ""

The documentation for this class was generated from the following file:

- `src/main/decaf/net/URISyntaxException.h`

6.584 decaf::internal::net::URIType Class Reference

Basic type object that holds data that composes a given URI.

```
#include <src/main/decaf/internal/net/URIType.h>
```

Public Member Functions

- **URIType** (const std::string &source)
- **URIType** ()
- virtual **~URIType** ()
- std::string **getSource** () const
*Gets the source URI string that was parsed to obtain this **URIType** (p. 2860) instance and the resulting data.,*
- void **setSource** (const std::string &source)

*Sets the source URI string that was parsed to obtain this **URIType** (p. 2860) instance and the resulting data.,*

- std::string **getScheme** () const
Gets the Scheme of the URI, e.g.
- void **setScheme** (const std::string &scheme)
Sets the Scheme of the URI, e.g.
- std::string **getSchemeSpecificPart** () const
Gets the Scheme Specific Part of the URI.
- void **setSchemeSpecificPart** (const std::string &schemeSpecificPart)
Sets the Scheme Specific Part of the URI.
- std::string **getAuthority** () const
Gets the Authority of the URI.
- void **setAuthority** (const std::string &authority)
Sets the Authority of the URI.
- std::string **getUserInfo** () const
Gets the user info part of the URI, e.g.
- void **setUserInfo** (const std::string &userinfo)
Sets the user info part of the URI, e.g.
- std::string **getHost** () const
Gets the Host name part of the URI.
- void **setHost** (const std::string &host)
Sets the Host name part of the URI.
- int **getPort** () const
Gets the port part of the URI.
- void **setPort** (int port)
Sets the port part of the URI.
- std::string **getPath** () const
Gets the Path part of the URI.
- void **setPath** (const std::string &path)
Sets the Path part of the URI.
- std::string **getQuery** () const
Gets the Query part of the URI.
- void **setQuery** (const std::string &query)
Sets the Query part of the URI.
- std::string **getFragment** () const
Gets the Fragment part of the URI.
- void **setFragment** (const std::string &fragment)
Sets the Fragment part of the URI.
- bool **isOpaque** () const
Gets if the URI is Opaque.
- void **setOpaque** (bool opaque)
Sets if the URI is Opaque.
- bool **isAbsolute** () const

Gets if the URI is Absolute.

- void **setAbsolute** (bool absolute)

Sets if the URI is Absolute.

- bool **isServerAuthority** () const

Gets if the URI is a Server Authority.

- void **setServerAuthority** (bool serverAuthority)

Sets if the URI is a Server Authority.

- bool **isValid** () const

Gets if the URI is valid, meaning that the source has been set and parsed and all relevant data fields have been set.

- void **setValid** (bool valid)

Sets if the URI is valid, meaning that the source has been set and parsed and all relevant data fields have been set.

6.584.1 Detailed Description

Basic type object that holds data that composes a given URI.

6.584.2 Constructor & Destructor Documentation

6.584.2.1 **decaf::internal::net::URIType::URIType** (const std::string & source)
[inline]

6.584.2.2 **decaf::internal::net::URIType::URIType** () [inline]

6.584.2.3 **virtual decaf::internal::net::URIType::~~URIType** () [inline, virtual]

6.584.3 Member Function Documentation

6.584.3.1 **std::string decaf::internal::net::URIType::getAuthority** () const
[inline]

Gets the Authority of the URI.

Returns

Authority part string.

6.584.3.2 **std::string decaf::internal::net::URIType::getFragment** () const
[inline]

Gets the Fragment part of the URI.

Returns

Fragment part string.

6.584.3.3 `std::string decaf::internal::net::URIType::getHost () const` `[inline]`

Gets the Host name part of the URI.

Returns

Host name part string.

6.584.3.4 `std::string decaf::internal::net::URIType::getPath () const` `[inline]`

Gets the Path part of the URI.

Returns

Path part string.

6.584.3.5 `int decaf::internal::net::URIType::getPort () const` `[inline]`

Gets the port part of the URI.

Returns

port part string, -1 if not set.

6.584.3.6 `std::string decaf::internal::net::URIType::getQuery () const` `[inline]`

Gets the Query part of the URI.

Returns

Query part string.

**6.584.3.7 `std::string decaf::internal::net::URIType::getScheme () const`
`[inline]`**

Gets the Scheme of the URI, e.g.

scheme ("http"/"ftp"/...).

Returns

scheme part string.

6.584.3.8 `std::string decaf::internal::net::URIType::getSchemeSpecificPart () const` `[inline]`

Gets the Scheme Specific Part of the URI.

Returns

scheme specific part string.

6.584.3.9 `std::string decaf::internal::net::URIType::getSource () const` `[inline]`

Gets the source URI string that was parsed to obtain this **URIType** (p. 2860) instance and the resulting data,.

Returns

the source URI string

6.584.3.10 `std::string decaf::internal::net::URIType::getUserInfo () const` `[inline]`

Gets the user info part of the URI, e.g.

user name, as in `http://user:passwd@host:port/`

Returns

user info part string.

6.584.3.11 `bool decaf::internal::net::URIType::isAbsolute () const` `[inline]`

Gets if the URI is Absolute.

Returns

true if Absolute.

6.584.3.12 `bool decaf::internal::net::URIType::isOpaque () const` `[inline]`

Gets if the URI is Opaque.

Returns

true if opaque.

6.584.3.13 **bool decaf::internal::net::URIType::isServerAuthority () const**
[inline]

Gets if the URI is a Server Authority.

Returns

true if Server Authority.

6.584.3.14 **bool decaf::internal::net::URIType::isValid () const** [inline]

Gets if the URI is valid, meaning that the source has been set and parsed and all relevant data fields have been set.

Returns

true if the **URIType** (p. 2860) contains valid data.

6.584.3.15 **void decaf::internal::net::URIType::setAbsolute (bool *absolute*)**
[inline]

Sets if the URI is Absolute.

Parameters

<i>absolute</i>	- true if Absolute.
-----------------	---------------------

6.584.3.16 **void decaf::internal::net::URIType::setAuthority (const std::string & *authority*)** [inline]

Sets the Authority of the URI.

Parameters

<i>authority</i>	Authority part string.
------------------	------------------------

6.584.3.17 **void decaf::internal::net::URIType::setFragment (const std::string & *fragment*)** [inline]

Sets the Fragment part of the URI.

Parameters

<i>fragment</i>	- Fragment part string.
-----------------	-------------------------

6.584.3.18 void decaf::internal::net::URIType::setHost (const std::string & *host*)
[inline]

Sets the Host name part of the URI.

Parameters

<i>host</i>	- Host name part string.
-------------	--------------------------

6.584.3.19 void decaf::internal::net::URIType::setOpaque (bool *opaque*)
[inline]

Sets if the URI is Opaque.

Parameters

<i>opaque</i>	true if opaque.
---------------	-----------------

6.584.3.20 void decaf::internal::net::URIType::setPath (const std::string & *path*)
[inline]

Sets the Path part of the URI.

Parameters

<i>path</i>	- Path part string.
-------------	---------------------

6.584.3.21 void decaf::internal::net::URIType::setPort (int *port*) [inline]

Sets the port part of the URI.

Parameters

<i>port</i>	- port part string, -1 if not set.
-------------	------------------------------------

6.584.3.22 void decaf::internal::net::URIType::setQuery (const std::string & *query*)
[inline]

Sets the Query part of the URI.

Parameters

<i>query</i>	- Query part string.
--------------	----------------------

6.584.3.23 `void decaf::internal::net::URIType::setScheme (const std::string & scheme) [inline]`

Sets the Scheme of the URI, e.g.

scheme ("http"/"ftp"/...).

Parameters

<i>scheme</i>	- scheme part string.
---------------	-----------------------

6.584.3.24 `void decaf::internal::net::URIType::setSchemeSpecificPart (const std::string & schemeSpecificPart) [inline]`

Sets the Scheme Specific Part of the URI.

Parameters

<i>scheme-SpecificPart</i>	- scheme specific part string.
----------------------------	--------------------------------

6.584.3.25 `void decaf::internal::net::URIType::setServerAuthority (bool serverAuthority) [inline]`

Sets if the URI is a Server Authority.

Parameters

<i>server-Authority</i>	- true if Server Authority.
-------------------------	-----------------------------

6.584.3.26 `void decaf::internal::net::URIType::setSource (const std::string & source) [inline]`

Sets the source URI string that was parsed to obtain this **URIType** (p. 2860) instance and the resulting data,.

Parameters

<i>source</i>	- the source URI string
---------------	-------------------------

6.584.3.27 `void decaf::internal::net::URIType::setUserInfo (const std::string & userinfo) [inline]`

Sets the user info part of the URI, e.g.

user name, as in `http://user:passwd@host:port/`

Parameters

<i>userinfo</i>	- user info part string.
-----------------	--------------------------

6.584.3.28 void **decaf::internal::net::URIType::setValid** (bool *valid*) [inline]

Sets if the URI is valid, meaning that the source has been set and parsed and all relevant data fields have been set.

Parameters

<i>valid</i>	- true if the URIType (p. 2860) contains valid data.
--------------	---

The documentation for this class was generated from the following file:

- `src/main/decaf/internal/net/URIType.h`

6.585 decaf::net::URL Class Reference

Class **URL** (p. 2868) represents a Uniform Resource Locator, a pointer to a "resource" on the World Wide Web.

```
#include <src/main/decaf/net/URL.h>
```

Public Member Functions

- **URL** ()
- **URL** (const std::string &url)
- virtual ~**URL** ()

6.585.1 Detailed Description

Class **URL** (p. 2868) represents a Uniform Resource Locator, a pointer to a "resource" on the World Wide Web.

A resource can be something as simple as a file or a directory, or it can be a reference to a more complicated object, such as a query to a database or to a search engine. More information on the types of URLs and their formats can be found at:

<http://www.ksc.nasa.gov/facts/internet/url-primer.html>

In general, a **URL** (p. 2868) can be broken into several parts. The previous example of a **URL** (p. 2868) indicates that the protocol to use is http (HyperText Transfer Protocol) and that the information resides on a host machine named `www.ksc.nasa.gov`. The information on that host machine is named `/facts/internet/url-primer.html`. The exact meaning of

this name on the host machine is both protocol dependent and host dependent. The information normally resides in a file, but it could be generated on the fly. This component of the **URL** (p. 2868) is called the path component.

A **URL** (p. 2868) can optionally specify a "port", which is the port number to which the TCP connection is made on the remote host machine. If the port is not specified, the default port for the protocol is used instead. For example, the default port for http is 80. An alternative port could be specified as:

```
http://www.ksc.nasa.gov:80/facts/internet/url-primer.html
```

The syntax of **URL** (p. 2868) is defined by RFC 2396: Uniform Resource Identifiers (**URI** (p. 2828)): Generic Syntax, amended by RFC 2732: Format for Literal IPv6 Addresses in URLs. The Literal IPv6 address format also supports `scope_ids`. The syntax and usage of `scope_ids` is described here.

A **URL** (p. 2868) may have appended to it a "fragment", also known as a "ref" or a "reference". The fragment is indicated by the sharp sign character "#" followed by more characters. For example,

```
http://www.apache.org/cms/index.html#chapter1
```

This fragment is not technically part of the **URL** (p. 2868). Rather, it indicates that after the specified resource is retrieved, the application is specifically interested in that part of the document that has the tag `chapter1` attached to it. The meaning of a tag is resource specific.

An application can also specify a "relative URL", which contains only enough information to reach the resource relative to another **URL** (p. 2868). Relative URLs are frequently used within HTML pages. For example, if the contents of the **URL** (p. 2868):

```
http://www.apache.org/cms/index.html
```

contained within it the relative **URL** (p. 2868):

```
FAQ.html
```

it would be a shorthand for:

```
http://www.apache.org/cms/FAQ.html
```

The relative **URL** (p. 2868) need not specify all the components of a **URL** (p. 2868). If the protocol, host name, or port number is missing, the value is inherited from the fully specified **URL** (p. 2868). The file component must be specified. The optional fragment is not inherited.

The **URL** (p. 2868) class does not itself encode or decode any **URL** (p. 2868) components according to the escaping mechanism defined in RFC2396. It is the responsibility of the caller to encode any fields, which need to be escaped prior to calling **URL** (p. 2868), and also to decode any escaped fields, that are returned from **URL** (p. 2868). Furthermore, because **URL** (p. 2868) has no knowledge of **URL** (p. 2868) escaping, it does not recognise equivalence between the encoded or decoded form of the same **URL** (p. 2868). For example, the two URLs:

```
http://foo.com/hello world/ and http://foo.com/hello%20world
```

would be considered not equal to each other.

Note, the **URI** (p. 2828) class does perform escaping of its component fields in certain circumstances. The recommended way to manage the encoding and decoding of URLs is to use **URI** (p. 2828), and to convert between these two classes using `toURI()` and `URI.toURL()` (p. 2840).

The **URLEncoder** (p. 2871) and **URLDecoder** (p. 2870) classes can also be used, but only for HTML form encoding, which is not the same as the encoding scheme defined in RFC2396.

6.585.2 Constructor & Destructor Documentation

6.585.2.1 `decaf::net::URL::URL ()`

6.585.2.2 `decaf::net::URL::URL (const std::string & url)`

6.585.2.3 `virtual decaf::net::URL::~~URL ()` `[inline, virtual]`

The documentation for this class was generated from the following file:

- `src/main/decaf/net/URL.h`

6.586 decaf::net::URLDecoder Class Reference

```
#include <src/main/decaf/net/URLDecoder.h>
```

Public Member Functions

- `virtual ~URLDecoder ()`

Static Public Member Functions

- `static std::string decode (const std::string &value)`

Decodes the string argument which is assumed to be encoded in the x-www-form-urlencoded MIME content type.

6.586.1 Constructor & Destructor Documentation

6.586.1.1 `virtual decaf::net::URLDecoder::~~URLDecoder ()` `[inline, virtual]`

6.586.2 Member Function Documentation

6.586.2.1 `static std::string decaf::net::URLDecoder::decode (const std::string & value)`
`[static]`

Decodes the string argument which is assumed to be encoded in the `x-www-form-urlencoded` MIME content type.

'+' will be converted to space, '%' and two following hex digit characters are converted to the equivalent byte value. All other characters are passed through unmodified.

e.g. "A+B+C %24%25" -> "A B C \$%"

Parameters

<i>value</i>	- string The encoded string.
--------------	------------------------------

Returns

The decoded version as a string.

The documentation for this class was generated from the following file:

- `src/main/decaf/net/URLDecoder.h`

6.587 decaf::net::URLEncoder Class Reference

```
#include <src/main/decaf/net/URLEncoder.h>
```

Public Member Functions

- `virtual ~URLEncoder ()`

Static Public Member Functions

- `static std::string encode (const std::string &value)`

This class contains a utility method for converting a string to the format required by the `application/x-www-form-urlencoded` MIME content type.

6.587.1 Constructor & Destructor Documentation

6.587.1.1 `virtual decaf::net::URLEncoder::~URLEncoder ()` `[inline, virtual]`

6.587.2 Member Function Documentation

6.587.2.1 `static std::string decaf::net::URLEncoder::encode (const std::string & value)`
`[static]`

This class contains a utility method for converting a string to the format required by the `application/x-www-form-urlencoded` MIME content type.

All characters except letters ('a'..'z', 'A'..'Z') and numbers ('0'..'9') and characters '.', '-', '*', '_' are converted into their hexadecimal value prepended by '%'.
 For example: '#' -> '%23'

For example: '#' -> '%23'

In addition, spaces are substituted by '+'

Parameters

<code>value</code>	- the string to be converted
--------------------	------------------------------

Returns

the converted string

The documentation for this class was generated from the following file:

- `src/main/decaf/net/URLEncoder.h`

6.588 activemq::util::Usage Class Reference

```
#include <src/main/activemq/util/Usage.h>
```

Inheritance diagram for `activemq::util::Usage`:

Public Member Functions

- virtual `~Usage ()`
- virtual void `waitForSpace ()=0`
*Waits forever for more space to be returned to this **Usage** (p. 2872) Manager.*
- virtual void `waitForSpace (unsigned int timeout)=0`
*Waits for more space to be returned to this **Usage** (p. 2872) Manager, times out when the given time span in milliseconds elapses.*
- virtual void `enqueueUsage (unsigned long long value)=0`
Tries to increase the usage by value amount but blocks if this object is currently full.
- virtual void `increaseUsage (unsigned long long value)=0`
Increases the usage by the value amount.
- virtual void `decreaseUsage (unsigned long long value)=0`
Decreases the usage by the value amount.
- virtual bool `isFull () const =0`
*Returns true if this **Usage** (p. 2872) instance is full, i.e.*

6.588.1 Constructor & Destructor Documentation

6.588.1.1 `virtual activemq::util::Usage::~~Usage () [inline, virtual]`

6.588.2 Member Function Documentation

6.588.2.1 `virtual void activemq::util::Usage::decreaseUsage (unsigned long long value) [pure virtual]`

Decreases the usage by the value amount.

Parameters

<i>value</i>	Amount of space to return to the pool
--------------	---------------------------------------

Implemented in **activemq::util::MemoryUsage** (p. 1819).

6.588.2.2 `virtual void activemq::util::Usage::enqueueUsage (unsigned long long value) [pure virtual]`

Tries to increase the usage by value amount but blocks if this object is currently full.

Parameters

<i>value</i>	Amount of usage in bytes to add.
--------------	----------------------------------

Implemented in **activemq::util::MemoryUsage** (p. 1819).

6.588.2.3 `virtual void activemq::util::Usage::increaseUsage (unsigned long long value) [pure virtual]`

Increases the usage by the value amount.

Parameters

<i>value</i>	Amount of usage to add.
--------------	-------------------------

Implemented in **activemq::util::MemoryUsage** (p. 1820).

6.588.2.4 `virtual bool activemq::util::Usage::isFull () const [pure virtual]`

Returns true if this **Usage** (p. 2872) instance is full, i.e.

Usage (p. 2872) $\geq 100\%$

Returns

true if **Usage** (p. 2872) is at the Full point.

Implemented in **activemq::util::MemoryUsage** (p. 1820).

6.588.2.5 `virtual void activemq::util::Usage::waitForSpace () [pure virtual]`

Waits forever for more space to be returned to this **Usage** (p. 2872) Manager.

Implemented in **activemq::util::MemoryUsage** (p. 1821).

6.588.2.6 `virtual void activemq::util::Usage::waitForSpace (unsigned int timeout) [pure virtual]`

Waits for more space to be returned to this **Usage** (p. 2872) Manager, times out when the given time span in milliseconds elapses.

Parameters

<i>timeout</i>	The time to wait for more space.
----------------	----------------------------------

Implemented in **activemq::util::MemoryUsage** (p. 1821).

The documentation for this class was generated from the following file:

- src/main/activemq/util/**Usage.h**

6.589 decaf::io::UTFDataFormatException Class Reference

Thrown from classes that attempt to read or write a UTF-8 encoded string and an encoding error is encountered.

```
#include <src/main/decaf/io/UTFDataFormatException.h>
```

Inheritance diagram for decaf::io::UTFDataFormatException:

Public Member Functions

- **UTFDataFormatException** () throw ()
Default Constructor.
- **UTFDataFormatException** (const **lang::Exception** &ex) throw ()
Copy Constructor.
- **UTFDataFormatException** (const **UTFDataFormatException** &ex) throw ()
Copy Constructor.

- **UTFDataFormatException** (const char *file, const int lineNumber, const std::exception *cause, const char *msg,...) throw ()
Constructor - Initializes the file name and line number where this message occurred.
- **UTFDataFormatException** (const std::exception *cause) throw ()
Constructor.
- **UTFDataFormatException** (const char *file, const int lineNumber, const char *msg,...) throw ()
Constructor.
- virtual **UTFDataFormatException * clone** () const
Clones this exception.
- virtual **~UTFDataFormatException** () throw ()

6.589.1 Detailed Description

Thrown from classes that attempt to read or write a UTF-8 encoded string and an encoding error is encountered.

Since

1.0

6.589.2 Constructor & Destructor Documentation

6.589.2.1 **decaf::io::UTFDataFormatException::UTFDataFormatException ()**
 throw () [inline]

Default Constructor.

6.589.2.2 **decaf::io::UTFDataFormatException::UTFDataFormatException (const lang::Exception & ex)** throw () [inline]

Copy Constructor.

Parameters

ex	the exception to copy
----	-----------------------

6.589.2.3 **decaf::io::UTFDataFormatException::UTFDataFormatException (const UTFDataFormatException & ex)** throw () [inline]

Copy Constructor.

Parameters

ex	the exception to copy, which is an instance of this type
----	--

6.589.2.4 `decaf::io::UTFDataFormatException::UTFDataFormatException (const char * file, const int lineNumber, const std::exception * cause, const char * msg, ...) throw () [inline]`

Constructor - Initializes the file name and line number where this message occurred.

Sets the message to report, using an optional list of arguments to parse into the message

Parameters

<i>file</i>	The file name where exception occurs
<i>lineNumber</i>	The line number where the exception occurred.
<i>cause</i>	The exception that was the cause for this one to be thrown.
<i>msg</i>	The message to report
...	list of primitives that are formatted into the message

6.589.2.5 `decaf::io::UTFDataFormatException::UTFDataFormatException (const std::exception * cause) throw () [inline]`

Constructor.

Parameters

<i>cause</i>	Pointer to the exception that caused this one to be thrown, the object is cloned caller retains ownership.
--------------	--

6.589.2.6 `decaf::io::UTFDataFormatException::UTFDataFormatException (const char * file, const int lineNumber, const char * msg, ...) throw () [inline]`

Constructor.

Parameters

<i>file</i>	The file name where exception occurs
<i>lineNumber</i>	The line number where the exception occurred.
<i>msg</i>	The message to report
...	list of primitives that are formatted into the message

6.589.2.7 `virtual decaf::io::UTFDataFormatException::~~UTFDataFormatException () throw () [inline, virtual]`

6.589.3 Member Function Documentation

```
6.589.3.1 virtual UTFDataFormatException* decaf::io::UTF-
DataFormatException::clone ( ) const [inline,
virtual]
```

Clones this exception.

This is useful for cases where you need to preserve the type of the original exception as well as the message. All subclasses should override.

Returns

A new instance of an Exception object that is a copy of this instance.

Reimplemented from **decaf::io::IOException** (p. 1547).

The documentation for this class was generated from the following file:

- src/main/decaf/io/UTFDataFormatException.h

6.590 decaf::util::UUID Class Reference

A class that represents an immutable universally unique identifier (**UUID** (p. 2877)).

```
#include <src/main/decaf/util/UUID.h>
```

Inheritance diagram for decaf::util::UUID:

Public Member Functions

- **UUID** (long long mostSigBits, long long leastSigBits)
*Constructs a new **UUID** (p. 2877) using the specified data.*
- virtual **~UUID** ()
- virtual int **compareTo** (const **UUID** &value) const
*Compare the given **UUID** (p. 2877) to this one.*
- virtual bool **equals** (const **UUID** &value) const
*Compares this **UUID** (p. 2877) to the one given, returns true if they are equal.*
- virtual bool **operator==** (const **UUID** &value) const
Compares equality between this object and the one passed.
- virtual bool **operator<** (const **UUID** &value) const
Compares this object to another and returns true if this object is considered to be less than the one passed.
- virtual std::string **toString** () const
*Returns a String object representing this **UUID** (p. 2877).*
- virtual long long **getLeastSignificantBits** () const
- virtual long long **getMostSignificantBits** () const

- virtual long long **node** ()
*The node value associated with this **UUID** (p. 2877).*
- virtual long long **timestamp** ()
*The timestamp value associated with this **UUID** (p. 2877).*
- virtual int **clockSequence** ()
*The clock sequence value associated with this **UUID** (p. 2877).*
- virtual int **variant** ()
*The variant number associated with this **UUID** (p. 2877).*
- virtual int **version** ()
*The version number associated with this **UUID** (p. 2877).*

Static Public Member Functions

- static **UUID randomUUID** ()
*Static factory to retrieve a type 4 (pseudo randomly generated) **UUID** (p. 2877).*
- static **UUID nameUUIDFromBytes** (const std::vector< char > &name)
*Static factory to retrieve a type 3 (name based) **UUID** (p. 2877) based on the specified byte array.*
- static **UUID nameUUIDFromBytes** (const char *name, std::size_t size)
*Static factory to retrieve a type 3 (name based) **UUID** (p. 2877) based on the specified byte array.*
- static **UUID fromString** (const std::string &name)
*Creates a **UUID** (p. 2877) from the string standard representation as described in the **toString()** (p. 2883) method.*

6.590.1 Detailed Description

A class that represents an immutable universally unique identifier (**UUID** (p. 2877)).

A **UUID** (p. 2877) represents a 128-bit value.

There exist different variants of these global identifiers. The methods of this class are for manipulating the Leach-Salz variant, although the constructors allow the creation of any variant of **UUID** (p. 2877) (described below).

The layout of a variant 2 (Leach-Salz) **UUID** (p. 2877) is as follows: The most significant long consists of the following unsigned fields:

```
0xFFFFFFFF00000000 time_low 0x00000000FFFF0000 time_mid 0x000000000000-
F000 version 0x00000000000000FF time_hi
```

The least significant long consists of the following unsigned fields:

```
0xC000000000000000 variant 0x3FFF000000000000 clock_seq 0x0000FFFFFFFF-
FF node
```

The variant field contains a value which identifies the layout of the **UUID** (p. 2877). The bit layout described above is valid only for a **UUID** (p. 2877) with a variant value of 2, which indicates the Leach-Salz variant.

The version field holds a value that describes the type of this **UUID** (p. 2877). There are four different basic types of UUIDs: time-based, DCE security, name-based, and randomly generated UUIDs. These types have a version value of 1, 2, 3 and 4, respectively.

For more information including algorithms used to create UUIDs, see the Internet-Draft UUIDs and GUIDs or the standards body definition at ISO/IEC 11578:1996.

6.590.2 Constructor & Destructor Documentation

6.590.2.1 `decaf::util::UUID::UUID (long long mostSigBits, long long leastSigBits)`

Constructs a new **UUID** (p. 2877) using the specified data.

mostSigBits is used for the most significant 64 bits of the **UUID** (p. 2877) and *leastSigBits* becomes the least significant 64 bits of the **UUID** (p. 2877).

Parameters

<i>mostSigBits</i>	
<i>leastSigBits</i>	

6.590.2.2 `virtual decaf::util::UUID::~~UUID ()` [virtual]

6.590.3 Member Function Documentation

6.590.3.1 `virtual int decaf::util::UUID::clockSequence ()` [virtual]

The clock sequence value associated with this **UUID** (p. 2877).

The 14 bit clock sequence value is constructed from the clock sequence field of this **UUID** (p. 2877). The clock sequence field is used to guarantee temporal uniqueness in a time-based **UUID** (p. 2877).

The *clockSequence* value is only meaningful in a time-based **UUID** (p. 2877), which has version type 1. If this **UUID** (p. 2877) is not a time-based **UUID** (p. 2877) then this method throws `UnsupportedOperationException`.

Returns

the *clockSequence* associated with a V1 **UUID** (p. 2877)

Exceptions

<i>Unsupported- OperationException</i>	if this UUID (p. 2877) version does not support this operation.
--	--

6.590.3.2 `virtual int decaf::util::UUID::compareTo (const UUID & value) const`
[virtual]

Compare the given **UUID** (p. 2877) to this one.

Parameters

<i>value</i>	- the UUID (p. 2877) to compare to
--------------	---

6.590.3.3 `virtual bool decaf::util::UUID::equals (const UUID & value) const`
[virtual]

Compares this **UUID** (p. 2877) to the one given, returns true if they are equal.

Parameters

<i>value</i>	The UUID (p. 2877) to compare to.
--------------	--

Returns

true if UUIDs are the same.

6.590.3.4 `static UUID decaf::util::UUID::fromString (const std::string & name)`
[static]

Creates a **UUID** (p. 2877) from the string standard representation as described in the **toString()** (p. 2883) method.

Parameters

<i>name</i>	A string to be used to construct a UUID (p. 2877).
-------------	---

Returns

type 3 **UUID** (p. 2877)

Exceptions

<i>IllegalArgumentException</i>	if the UUID (p. 2877) string given is invalid.
---------------------------------	---

6.590.3.5 `virtual long long decaf::util::UUID::getLeastSignificantBits () const`
[virtual]

Returns

the most significant 64 bits of this **UUID** (p. 2877)'s 128 bit value.

6.590.3.6 `virtual long long decaf::util::UUID::getMostSignificantBits () const`
`[virtual]`

Returns

the most significant 64 bits of this **UUID** (p. 2877)'s 128 bit value.

6.590.3.7 `static UUID decaf::util::UUID::nameUUIDFromBytes (const std::vector<`
`char > & name) [static]`

Static factory to retrieve a type 3 (name based) **UUID** (p. 2877) based on the specified byte array.

Parameters

<i>name</i>	A byte array to be used to construct a UUID (p. 2877).
-------------	---

Returns

type 3 **UUID** (p. 2877)

6.590.3.8 `static UUID decaf::util::UUID::nameUUIDFromBytes (const char * name,`
`std::size_t size) [static]`

Static factory to retrieve a type 3 (name based) **UUID** (p. 2877) based on the specified byte array.

Parameters

<i>name</i>	A byte array to be used to construct a UUID (p. 2877).
<i>size</i>	The size of the byte array, or number of bytes to use.

Returns

type 3 **UUID** (p. 2877)

6.590.3.9 `virtual long long decaf::util::UUID::node () [virtual]`

The node value associated with this **UUID** (p. 2877).

The 48 bit node value is constructed from the node field of this **UUID** (p. 2877). This field is intended to hold the IEEE 802 address of the machine that generated this **UUID** (p. 2877) to guarantee spatial uniqueness.

The node value is only meaningful in a time-based **UUID** (p. 2877), which has version type 1. If this **UUID** (p. 2877) is not a time-based **UUID** (p. 2877) then this method throws `UnsupportedOperationException`.

Returns

the node value of this **UUID** (p. 2877)

Exceptions

<i>Unsupported- OperationException</i>	if this UUID (p. 2877) version does not support this operation.
--	--

6.590.3.10 `virtual bool decaf::util::UUID::operator< (const UUID & value) const`
[virtual]

Compares this object to another and returns true if this object is considered to be less than the one passed.

This

Parameters

<i>value</i>	- the value to be compared to this one.
--------------	---

Returns

true if this object is equal to the one passed.

6.590.3.11 `virtual bool decaf::util::UUID::operator== (const UUID & value) const`
[virtual]

Compares equality between this object and the one passed.

Parameters

<i>value</i>	- the value to be compared to this one.
--------------	---

Returns

true if this object is equal to the one passed.

6.590.3.12 `static UUID decaf::util::UUID::randomUUID ()` [static]

Static factory to retrieve a type 4 (pseudo randomly generated) **UUID** (p. 2877).

The **UUID** (p. 2877) is generated using a cryptographically strong pseudo random number generator.

Returns

type 4 **UUID** (p. 2877)

6.590.3.13 virtual long long decaf::util::UUID::timestamp () [virtual]

The timestamp value associated with this **UUID** (p. 2877).

The 60 bit timestamp value is constructed from the time_low, time_mid, and time_hi fields of this **UUID** (p. 2877). The resulting timestamp is measured in 100-nanosecond units since midnight, October 15, 1582 UTC.

The timestamp value is only meaningful in a time-based **UUID** (p. 2877), which has version type 1. If this **UUID** (p. 2877) is not a time-based **UUID** (p. 2877) then this method throws UnsupportedOperationException.

Returns

the timestamp associated with a V1 **UUID** (p. 2877)

Exceptions

<i>Unsupported-OperationException</i>	if this UUID (p. 2877) version does not support this operation.
---------------------------------------	--

6.590.3.14 virtual std::string decaf::util::UUID::toString () const [virtual]

Returns a String object representing this **UUID** (p. 2877).

UUID (p. 2877)'s are formatted as: 00112233-4455-6677-8899-AABBCCDDEEFF whose length is 36.

Returns

formatted string for this **UUID** (p. 2877)

6.590.3.15 virtual int decaf::util::UUID::variant () [virtual]

The variant number associated with this **UUID** (p. 2877).

The variant number describes the layout of the **UUID** (p. 2877). The variant number has the following meaning:

* 0 Reserved for NCS backward compatibility * 2 The Leach-Salz variant (used by this class) * 6 Reserved, Microsoft Corporation backward compatibility * 7 Reserved for future definition

Returns

the variant associated with a V1 **UUID** (p. 2877)

Exceptions

<i>Unsupported- OperationException</i>	if this UUID (p. 2877) version does not support this operation.
--	--

6.590.3.16 virtual int decaf::util::UUID::version () [virtual]

The version number associated with this **UUID** (p. 2877).

The version number describes how this **UUID** (p. 2877) was generated. The version number has the following meaning:

* 1 Time-based **UUID** (p. 2877) * 2 DCE security **UUID** (p. 2877) * 3 Name-based **UUID** (p. 2877) * 4 Randomly generated **UUID** (p. 2877)

Returns

the version associated with a V1 **UUID** (p. 2877)

Exceptions

<i>Unsupported- OperationException</i>	if this UUID (p. 2877) version does not support this operation.
--	--

The documentation for this class was generated from the following file:

- src/main/decaf/util/**UUID.h**

6.591 activemq::wireformat::WireFormat Class Reference

Provides a mechanism to marshal commands into and out of packets or into and out of streams, Channels and Datagrams.

```
#include <src/main/activemq/wireformat/WireFormat.h>
```

Inheritance diagram for activemq::wireformat::WireFormat:

Public Member Functions

- virtual ~**WireFormat** ()

- virtual void **marshal** (const **Pointer**< **commands::Command** > &command, const **activemq::transport::Transport** *transport, **decaf::io::DataOutputStream** *out)=0

Stream based marshaling of a Command, this method blocks until the entire Command has been written out to the output stream.

- virtual **Pointer** < **commands::Command** > **unmarshal** (const **activemq::transport::Transport** *transport, **decaf::io::DataInputStream** *in)=0

Stream based unmarshaling, blocks on reads on the input stream until a complete command has been read and unmarshaled into the correct form.

- virtual void **setVersion** (int version)=0

Set the Version.

- virtual int **getVersion** () const =0

Get the Version.

- virtual bool **hasNegotiator** () const =0

*Returns true if this **WireFormat** (p.2884) has a Negotiator that needs to wrap the Transport that uses it.*

- virtual bool **inReceive** () const =0

Indicates if the WireFormat object is in the process of receiving a message.

- virtual **Pointer** < **transport::Transport** > **createNegotiator** (const **Pointer**< **transport::Transport** > &transport)=0

If the Transport Provides a Negotiator this method will create and return a new instance of the Negotiator.

6.591.1 Detailed Description

Provides a mechanism to marshal commands into and out of packets or into and out of streams, Channels and Datagrams.

Version

Revision:

1.1

6.591.2 Constructor & Destructor Documentation

- 6.591.2.1 virtual **activemq::wireformat::WireFormat::~WireFormat** ()
[inline, virtual]

6.591.3 Member Function Documentation

6.591.3.1 virtual **Pointer**<**transport::Transport**> **activemq::wireformat::WireFormat::createNegotiator** (const **Pointer**< **transport::Transport** > & *transport*) [pure virtual]

If the Transport Provides a Negotiator this method will create and return a new instance of the Negotiator.

Parameters

<i>transport</i>	The Transport to Wrap the Negotiator around.
------------------	--

Returns

new instance of a **WireFormatNegotiator** (p. 2904) as a **Pointer**<**Transport**> (p. 2083).

Exceptions

<i>UnsupportedOperationException</i>	if the WireFormat (p. 2884) doesn't have a Negotiator.
--------------------------------------	---

Implemented in **activemq::wireformat::stomp::StompWireFormat** (p. 2598), and **activemq::wireformat::openwire::OpenWireFormat** (p. 2049).

6.591.3.2 virtual int **activemq::wireformat::WireFormat::getVersion** () const [pure virtual]

Get the Version.

Returns

the version of the wire format

Implemented in **activemq::wireformat::openwire::OpenWireFormat** (p. 2051), and **activemq::wireformat::stomp::StompWireFormat** (p. 2599).

6.591.3.3 virtual bool **activemq::wireformat::WireFormat::hasNegotiator** () const [pure virtual]

Returns true if this **WireFormat** (p. 2884) has a Negotiator that needs to wrap the - Transport that uses it.

Returns

true if the **WireFormat** (p. 2884) provides a Negotiator.

Implemented in **activemq::wireformat::stomp::StompWireFormat** (p. 2599), and **activemq::wireformat::openwire::OpenWireFormat** (p. 2051).

6.591.3.4 `virtual bool activemq::wireformat::WireFormat::inReceive () const`
`[pure virtual]`

Indicates if the WireFormat object is in the process of receiving a message.

This is useful for monitoring inactivity and the **WireFormat** (p. 2884) is processing a large message which takes longer than some configured timeout to unmarshal, the inactivity monitor can query the **WireFormat** (p. 2884) instance to determine if its busy or not and not mark the connection as inactive if so.

Returns

true if the **WireFormat** (p. 2884) object is unmarshaling a message.

Implemented in **activemq::wireformat::openwire::OpenWireFormat** (p. 2051), and **activemq::wireformat::stomp::StompWireFormat** (p. 2599).

6.591.3.5 `virtual void activemq::wireformat::WireFormat::marshal (`
`const Pointer< commands::Command > & command, const`
`activemq::transport::Transport * transport, decaf::io::DataOutputStream`
`* out) [pure virtual]`

Stream based marshaling of a Command, this method blocks until the entire Command has been written out to the output stream.

Parameters

<i>command</i>	The Command to Marshal
<i>transport</i>	The Transport that called this method.
<i>out</i>	The output stream to write the command to.

Exceptions

<i>IOException</i>	if an I/O error occurs.
--------------------	-------------------------

Implemented in **activemq::wireformat::openwire::OpenWireFormat** (p. 2053), and **activemq::wireformat::stomp::StompWireFormat** (p. 2599).

6.591.3.6 `virtual void activemq::wireformat::WireFormat::setVersion (int version)`
`[pure virtual]`

Set the Version.

Parameters

<i>version</i>	the version of the wire format
----------------	--------------------------------

Implemented in **activemq::wireformat::openwire::OpenWireFormat** (p. 2056), and **activemq::wireformat::stomp::StompWireFormat** (p. 2600).

6.591.3.7 virtual **Pointer**<**commands::Command**> **activemq::wireformat::WireFormat::unmarshal** (const **activemq::transport::Transport** * *transport*, **decaf::io::DataInputStream** * *in*) [pure virtual]

Stream based unmarshaling, blocks on reads on the input stream until a complete command has been read and unmarshaled into the correct form.

Returns a Pointer to the newly unmarshaled Command.

Parameters

<i>transport</i>	Pointer to the transport that is making this request.
<i>in</i>	The input stream to read the command from.

Returns

the newly marshaled Command, caller owns the pointer

Exceptions

<i>IOException</i>	if an I/O error occurs.
--------------------	-------------------------

Implemented in **activemq::wireformat::openwire::OpenWireFormat** (p. 2058), and **activemq::wireformat::stomp::StompWireFormat** (p. 2600).

The documentation for this class was generated from the following file:

- src/main/activemq/wireformat/**WireFormat.h**

6.592 activemq::wireformat::WireFormatFactory Class Reference

The **WireFormatFactory** (p. 2888) is the interface that all **WireFormatFactory** (p. 2888) classes must extend.

```
#include <src/main/activemq/wireformat/WireFormatFactory.h>
```

Inheritance diagram for **activemq::wireformat::WireFormatFactory**:

Public Member Functions

- virtual **~WireFormatFactory** ()
- virtual **Pointer**< **WireFormat** > **createWireFormat** (const **decaf::util::Properties** &properties)=0
*Creates a new **WireFormat** (p. 2884) Object passing it a set of properties from which it can obtain any optional settings.*

6.592.1 Detailed Description

The **WireFormatFactory** (p. 2888) is the interface that all **WireFormatFactory** (p. 2888) classes must extend.

The Factory creates a **WireFormat** (p. 2884) Object based on the properties that are set in the passed `Properties` object.

6.592.2 Constructor & Destructor Documentation

6.592.2.1 `virtual activemq::wireformat::WireFormatFactory::~WireFormatFactory () [inline, virtual]`

6.592.3 Member Function Documentation

6.592.3.1 `virtual Pointer<WireFormat> activemq::wireformat::WireFormatFactory::createWireFormat (const decaf::util::Properties & properties) [pure virtual]`

Creates a new **WireFormat** (p. 2884) Object passing it a set of properties from which it can obtain any optional settings.

Parameters

<i>properties</i>	The Properties for this WireFormat (p. 2884).
-------------------	--

Returns

Pointer to a new instance of a **WireFormat** (p. 2884) object.

Exceptions

<i>IllegalStateException</i>	if the factory has not been initialized.
------------------------------	--

Implemented in `activemq::wireformat::openwire::OpenWireFormatFactory` (p. 2060), and `activemq::wireformat::stomp::StompWireFormatFactory` (p. 2601).

The documentation for this class was generated from the following file:

- `src/main/activemq/wireformat/WireFormatFactory.h`

6.593 activemq::commands::WireFormatInfo Class Reference

```
#include <src/main/activemq/commands/WireFormatInfo.h>
```

Inheritance diagram for `activemq::commands::WireFormatInfo`:

Public Member Functions

- **WireFormatInfo** ()
- virtual **~WireFormatInfo** ()
- virtual unsigned char **getDataStructureType** () const
*Get the **DataStructure** (p. 1133) Type as defined in CommandTypes.h.*
- virtual **DataStructure** * **cloneDataStructure** () const
Clone this object and return a new instance that the caller now owns, this will be an exact copy of this one.
- virtual void **copyDataStructure** (const **DataStructure** *src)
Copy the contents of the passed object into this objects members, overwriting any existing data.
- virtual std::string **toString** () const
*Returns a string containing the information for this **DataStructure** (p. 1133) such as its type and value of its elements.*
- virtual bool **equals** (const **DataStructure** *value) const
*Compares the **DataStructure** (p. 1133) passed in to this one, and returns if they are equivalent.*
- virtual bool **isMarshalAware** () const
Determine if the class implementing this interface is really wanting to be told about marshaling.
- virtual **decaf::lang::Pointer** < **commands::Command** > **visit** (**activemq::state::CommandVisitor** *visitor) throw (exceptions::ActiveMQException)
Allows a Visitor to visit this command and return a response to the command based on the command type being visited.
- int **getVersion** () const
Get the current Wireformat Version.
- void **setVersion** (int version)
Set the current Wireformat Version.
- long long **getMaxInactivityDuration** () const
Returns the currently configured Max Inactivity duration.
- void **setMaxInactivityDuration** (long long maxInactivityDuration)
Sets the Max inactivity duration value.
- long long **getMaxInactivityDurationInitalDelay** () const
Returns the currently configured Max Inactivity Intial Delay duration.
- void **setMaxInactivityDurationInitalDelay** (long long maxInactivityDuration-InitalDelay)
Sets the Max inactivity initial delay duration value.
- bool **isStackTraceEnabled** () const
Checks if the stackTraceEnabled flag is on.
- void **setStackTraceEnabled** (bool stackTraceEnabled)
Sets if the stackTraceEnabled flag is on.
- bool **isTcpNoDelayEnabled** () const
Checks if the tcpNoDelayEnabled flag is on.
- void **setTcpNoDelayEnabled** (bool tcpNoDelayEnabled)

- Sets if the tcpNoDelayEnabled flag is on.*

 - bool **isCacheEnabled** () const

Checks if the cacheEnabled flag is on.

 - void **setCacheEnabled** (bool cacheEnabled)

Sets if the cacheEnabled flag is on.

 - int **getCacheSize** () const

Gets the Cache Size setting.

 - void **setCacheSize** (int value)

Sets the Cache Size setting.

 - bool **isTightEncodingEnabled** () const

Checks if the tightEncodingEnabled flag is on.

 - void **setTightEncodingEnabled** (bool tightEncodingEnabled)

Sets if the tightEncodingEnabled flag is on.

 - bool **isSizePrefixDisabled** () const

Checks if the sizePrefixDisabled flag is on.

 - void **setSizePrefixDisabled** (bool sizePrefixDisabled)

Sets if the sizePrefixDisabled flag is on.

 - const std::vector< unsigned char > & **getMagic** () const

Get the Magic field.

 - void **setMagic** (const std::vector< unsigned char > &magic)

Sets the value of the magic field.

 - const std::vector< unsigned char > & **getMarshaledProperties** () const

Get the marshalledProperties field.

 - void **setMarshaledProperties** (const std::vector< unsigned char > &marshalledProperties)

Sets the value of the marshalledProperties field.

 - virtual const **util::PrimitiveMap** & **getProperties** () const

*Gets the Properties for this **Command** (p. 866).*

 - virtual **util::PrimitiveMap** & **getProperties** ()

*Gets the Properties for this **Command** (p. 866).*

 - virtual void **setProperties** (const **util::PrimitiveMap** &map)

*Sets the Properties for this **Command** (p. 866).*

 - bool **isValid** () const

*Determines if we think this is a Valid **WireFormatInfo** (p. 2889) command.*

 - virtual bool **isWireFormatInfo** () const
 - virtual void **beforeMarshal** (**wireformat::WireFormat** *wireFormat AMQCPP_UNUSED)
 - virtual void **afterUnmarshal** (**wireformat::WireFormat** *wireFormat AMQCPP_UNUSED)

Static Public Attributes

- static const unsigned char **ID_WIREFORMATINFO** = 1

6.593.1 Constructor & Destructor Documentation

6.593.1.1 `activemq::commands::WireFormatInfo::WireFormatInfo ()`

6.593.1.2 `virtual activemq::commands::WireFormatInfo::~~WireFormatInfo ()`
[virtual]

6.593.2 Member Function Documentation

6.593.2.1 `virtual void activemq::commands::WireFormatInfo::afterUnmarshal (wireformat::WireFormat *wireFormat AMQCPP_UNUSED)` [virtual]

Reimplemented from `activemq::commands::BaseDataStructure` (p. 529).

6.593.2.2 `virtual void activemq::commands::WireFormatInfo::beforeMarshal (wireformat::WireFormat *wireFormat AMQCPP_UNUSED)` [virtual]

Reimplemented from `activemq::commands::BaseDataStructure` (p. 530).

6.593.2.3 `virtual DataStructure* activemq::commands::WireFormatInfo::clone-DataStructure () const` [virtual]

Clone this object and return a new instance that the caller now owns, this will be an exact copy of this one.

Returns

new copy of this object.

Implements `activemq::commands::DataStructure` (p. 1133).

6.593.2.4 `virtual void activemq::commands::WireFormatInfo::copyDataStructure (const DataStructure *src)` [virtual]

Copy the contents of the passed object into this objects members, overwriting any existing data.

Returns

src - Source Object

Reimplemented from `activemq::commands::BaseCommand` (p. 493).

6.593.2.5 `virtual bool activemq::commands::WireFormatInfo::equals (const DataStructure *value) const` [virtual]

Compares the `DataStructure` (p. 1133) passed in to this one, and returns if they are equivalent.

Equivalent here means that they are of the same type, and that each element of the objects are the same.

Returns

true if **DataStructure** (p. 1133)'s are Equal.

Reimplemented from **activemq::commands::BaseCommand** (p. 494).

6.593.2.6 `int activemq::commands::WireFormatInfo::getCacheSize () const`

Gets the Cache Size setting.

Returns

currently set cache size.

6.593.2.7 `virtual unsigned char activemq::commands::WireFormatInfo::getDataStructureType () const` [virtual]

Get the **DataStructure** (p. 1133) Type as defined in CommandTypes.h.

Returns

The type of the data structure

Implements **activemq::commands::DataStructure** (p. 1137).

6.593.2.8 `const std::vector<unsigned char>& activemq::commands::WireFormatInfo::getMagic () const` [inline]

Get the Magic field.

Returns

const reference to a `std::vector<char>`

6.593.2.9 `const std::vector<unsigned char>& activemq::commands::WireFormatInfo::getMarshaledProperties () const` [inline]

Get the marshaledProperties field.

Returns

const reference to a `std::vector<char>`

6.593.2.10 `long long activemq::commands::WireFormatInfo::getMaxInactivity-Duration () const`

Returns the currently configured Max Inactivity duration.

Returns

the set inactivity duration value.

6.593.2.11 `long long activemq::commands::WireFormatInfo::getMaxInactivity-DurationInitialDelay () const`

Returns the currently configured Max Inactivity Initial Delay duration.

Returns

the set inactivity duration initial delay value.

6.593.2.12 `virtual const util::PrimitiveMap& activemq::commands::WireFormatInfo::getProperties () const [inline, virtual]`

Gets the Properties for this **Command** (p. 866).

Returns

the Properties object for this **Command** (p. 866).

6.593.2.13 `virtual util::PrimitiveMap& activemq::commands::WireFormatInfo::getProperties () [inline, virtual]`

Gets the Properties for this **Command** (p. 866).

Returns

the Properties object for this **Command** (p. 866).

6.593.2.14 `int activemq::commands::WireFormatInfo::getVersion () const [inline]`

Get the current Wireformat Version.

Returns

int that identifies the version

6.593.2.15 **bool activemq::commands::WireFormatInfo::isCacheEnabled () const**

Checks if the cacheEnabled flag is on.

Returns

true if the flag is on.

6.593.2.16 **virtual bool activemq::commands::WireFormatInfo::isMarshalAware ()
const [inline, virtual]**

Determine if the class implementing this interface is really wanting to be told about marshaling.

Normally if you didn't want to be marshal aware you just wouldn't implement this interface but since this is C++ and we don't have true interfaces we need a flat inheritance hierarchy, so we always implement this.

Returns

true if this class cares about marshaling.

Reimplemented from **activemq::commands::BaseDataStructure** (p. 530).

6.593.2.17 **bool activemq::commands::WireFormatInfo::isSizePrefixDisabled ()
const**

Checks if the sizePrefixDisabled flag is on.

Returns

true if the flag is on.

6.593.2.18 **bool activemq::commands::WireFormatInfo::isStackTraceEnabled ()
const**

Checks if the stackTraceEnabled flag is on.

Returns

true if the flag is on.

6.593.2.19 **bool activemq::commands::WireFormatInfo::isTcpNoDelayEnabled ()
const**

Checks if the tcpNoDelayEnabled flag is on.

Returns

true if the flag is on.

6.593.2.20 `bool activemq::commands::WireFormatInfo::isTightEncodingEnabled () const`

Checks if the tightEncodingEnabled flag is on.

Returns

true if the flag is on.

6.593.2.21 `bool activemq::commands::WireFormatInfo::isValid () const`

Determines if we think this is a Valid **WireFormatInfo** (p. 2889) command.

Returns

true if its valid.

6.593.2.22 `virtual bool activemq::commands::WireFormatInfo::isWireFormatInfo () const [inline, virtual]`

Returns

answers true to the isWireFormatInfo query

Reimplemented from **activemq::commands::BaseCommand** (p. 497).

6.593.2.23 `void activemq::commands::WireFormatInfo::setCacheEnabled (bool cacheEnabled)`

Sets if the cacheEnabled flag is on.

Parameters

<i>cache-Enabled</i>	- true to turn flag is on
----------------------	---------------------------

6.593.2.24 `void activemq::commands::WireFormatInfo::setCacheSize (int value)`

Sets the Cache Size setting.

Parameters

<i>value</i>	- value to set to the cache size.
--------------	-----------------------------------

6.593.2.25 `void activemq::commands::WireFormatInfo::setMagic (const
std::vector< unsigned char > & magic) [inline]`

Sets the value of the magic field.

Parameters

<i>magic</i>	- const std::vector<char>
--------------	---------------------------

6.593.2.26 `void activemq::commands::WireFormatInfo::setMarshaledProperties (const
std::vector< unsigned char > & marshalledProperties) [inline]`

Sets the value of the marshalledProperties field.

Parameters

<i>marshalled- Properties</i>	The Byte Array vector that contains the marshaled form of the Message (p. 1821) properties, this is the data sent over the wire.
-----------------------------------	---

6.593.2.27 `void activemq::commands::WireFormatInfo::setMaxInactivityDuration (long long
maxInactivityDuration)`

Sets the Max inactivity duration value.

Parameters

<i>max- Inactivity- Duration</i>	- max time a client can be inactive.
--	--------------------------------------

6.593.2.28 `void activemq::commands::WireFormatInfo::setMaxInactivity-
DurationInitalDelay (long long maxInactivityDurationInitalDelay
)`

Sets the Max inactivity initial delay duration value.

Parameters

<i>max- Inactivity- Duration- InitalDelay</i>	- time before the inactivity delay is checked.
---	--

6.593.2.29 `virtual void activemq::commands::WireFormatInfo::setProperties (const util::PrimitiveMap & map) [inline, virtual]`

Sets the Properties for this **Command** (p. 866).

Parameters

<i>map</i>	- PrimitiveMap to copy
------------	------------------------

6.593.2.30 `void activemq::commands::WireFormatInfo::setSizePrefixDisabled (bool sizePrefixDisabled)`

Sets if the sizePrefixDisabled flag is on.

Parameters

<i>sizePrefix-Disabled</i>	- true to turn flag is on
----------------------------	---------------------------

6.593.2.31 `void activemq::commands::WireFormatInfo::setStackTraceEnabled (bool stackTraceEnabled)`

Sets if the stackTraceEnabled flag is on.

Parameters

<i>stackTrace-Enabled</i>	- ture to turn flag is on
---------------------------	---------------------------

6.593.2.32 `void activemq::commands::WireFormatInfo::setTcpNoDelayEnabled (bool tcpNoDelayEnabled)`

Sets if the tcpNoDelayEnabled flag is on.

Parameters

<i>tcpNoDelay-Enabled</i>	- ture to turn flag is on
---------------------------	---------------------------

6.593.2.33 `void activemq::commands::WireFormatInfo::setTightEncodingEnabled (bool tightEncodingEnabled)`

Sets if the tightEncodingEnabled flag is on.

Parameters

<i>tight- Encoding- Enabled</i>	- true to turn flag is on
---	---------------------------

6.593.2.34 `void activemq::commands::WireFormatInfo::setVersion (int version)`
`[inline]`

Set the current Wireformat Version.

Parameters

<i>version</i>	- int that identifies the version
----------------	-----------------------------------

6.593.2.35 `virtual std::string activemq::commands::WireFormatInfo::toString ()`
`const [virtual]`

Returns a string containing the information for this **DataStructure** (p. 1133) such as its type and value of its elements.

Returns

formatted string useful for debugging.

Reimplemented from **activemq::commands::BaseCommand** (p. 498).

6.593.2.36 `virtual decaf::lang::Pointer<commands::Command>`
`activemq::commands::WireFormatInfo::visit (activemq::state::-`
`CommandVisitor * visitor) throw (exceptions::ActiveMQException)`
`[virtual]`

Allows a Visitor to visit this command and return a response to the command based on the command type being visited.

The command will call the proper processXXX method in the visitor.

Returns

a **Response** (p. 2298) to the visitor being called or NULL if no response.

Implements **activemq::commands::Command** (p. 871).

6.593.3 Field Documentation

6.593.3.1 `const unsigned char activemq::commands::WireFormatInfo::ID_WIREFOR-`
`MATINFO = 1 [static]`

The documentation for this class was generated from the following file:

6.594

activemq::wireformat::openwire::marshal::generated::WireFormatInfoMarshaller
Class Reference 2907

• [src/main/activemq/commands/WireFormatInfo.h](#)

6.594 activemq::wireformat::openwire::marshal::generated::WireFormatInfoMarshaller Class Reference

Marshaling code for Open Wire Format for **WireFormatInfoMarshaller** (p. 2900).

```
#include <src/main/activemq/wireformat/openwire/marshal/generated/-  
WireFormatInfoMarshaller.h>
```

Inheritance diagram for `activemq::wireformat::openwire::marshal::generated::WireFormatInfoMarshaller`:

Public Member Functions

- **WireFormatInfoMarshaller** ()
- virtual **~WireFormatInfoMarshaller** ()
- virtual **commands::DataStructure * createObject** () const
Creates a new instance of the class that this class is a marshaling director for.
- virtual unsigned char **getDataStructureType** () const
Gets the DataStructureType that this class marshals/unmarshals.
- virtual void **tightUnmarshal** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataInputStream** *dataIn, **utils::BooleanStream** *bs)
Tight Un-marhsal to the given stream.
- virtual int **tightMarshal1** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **utils::BooleanStream** *bs)
Tight Marhsal to the given stream.
- virtual void **tightMarshal2** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataOutputStream** *dataOut, **utils::BooleanStream** *bs)
Tight Marhsal to the given stream.
- virtual void **looseUnmarshal** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataInputStream** *dataIn)
Loose Un-marhsal to the given stream.
- virtual void **looseMarshal** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataOutputStream** *dataOut)
Tight Marhsal to the given stream.

6.594.1 Detailed Description

Marshaling code for Open Wire Format for **WireFormatInfoMarshaller** (p. 2900).

NOTE!: This file is auto generated - do not modify! if you need to make a change, please see the Java Classes in the activemq-openwire-generator module

6.594.2 Constructor & Destructor Documentation

6.594.2.1 **activemq::wireformat::openwire::marshal::generated::WireFormatInfoMarshaller::WireFormatInfoMarshaller ()**
[inline]

6.594.2.2 **virtual activemq::wireformat::openwire::marshal::generated::WireFormatInfoMarshaller::~~WireFormatInfoMarshaller ()** [inline, virtual]

6.594.3 Member Function Documentation

6.594.3.1 **virtual commands::DataStructure* activemq::wireformat::openwire::marshal::generated::WireFormatInfoMarshaller::createObject ()** const
[virtual]

Creates a new instance of the class that this class is a marshaling director for.

Returns

newly allocated Command

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1120).

6.594.3.2 **virtual unsigned char activemq::wireformat::openwire::marshal::generated::WireFormatInfoMarshaller::getDataStructureType ()** const
[virtual]

Gets the DataStructureType that this class marshals/unmarshals.

Returns

byte Id of this classes DataStructureType

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1122).

6.594

activemq::wireformat::openwire::marshal::generated::WireFormatInfoMarshaller
Class Reference 2909

6.594.3.3 `virtual void activemq::wireformat::openwire::marshal::generated::WireFormatInfoMarshaller::looseMarshal (OpenWireFormat * format, commands::DataStructure * command, decaf::io::DataOutputStream * ds) [virtual]`

Tight Marhsal to the given stream.

Parameters

<i>format</i>	- The OpenWireFormat properties
<i>command</i>	- the object to Marshal
<i>ds</i>	- DataOutputStream to marshal to

Exceptions

<i>IOException</i>	if an error occurs.
--------------------	---------------------

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1123).

6.594.3.4 `virtual void activemq::wireformat::openwire::marshal::generated::WireFormatInfoMarshaller::looseUnmarshal (OpenWireFormat * format, commands::DataStructure * command, decaf::io::DataInputStream * dis) [virtual]`

Loose Un-marhsal to the given stream.

Parameters

<i>format</i>	- The OpenWireFormat properties
<i>command</i>	- the object to Un-Marshal
<i>dis</i>	- the DataInputStream to Un-Marshal from

Exceptions

<i>IOException</i>	if an error occurs.
--------------------	---------------------

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1125).

6.594.3.5 `virtual int activemq::wireformat::openwire::marshal::generated::WireFormatInfoMarshaller::tightMarshal1 (OpenWireFormat * format, commands::DataStructure * command, utils::BooleanStream * bs) [virtual]`

Tight Marhsal to the given stream.

Parameters

<i>format</i>	The OpenwireFormat properties
<i>command</i>	The object to Marshal
<i>bs</i>	The boolean stream to marshal to.

Exceptions

<i>IOException</i>	if an error occurs.
--------------------	---------------------

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1127).

6.594.3.6 `virtual void activemq::wireformat::openwire::marshal::generated::WireFormatInfoMarshaller::tightMarshal2 (OpenWireFormat * format, commands::DataStructure * command, decaf::io::DataOutputStream * ds, utils::BooleanStream * bs) [virtual]`

Tight Marshal to the given stream.

Parameters

<i>format</i>	- The OpenwireFormat properties
<i>command</i>	- the object to Marshal
<i>ds</i>	- the DataOutputStream to Marshal to
<i>bs</i>	- boolean stream to marshal to.

Exceptions

<i>IOException</i>	if an error occurs.
--------------------	---------------------

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1129).

6.594.3.7 `virtual void activemq::wireformat::openwire::marshal::generated::WireFormatInfoMarshaller::tightUnmarshal (OpenWireFormat * format, commands::DataStructure * command, decaf::io::DataInputStream * dis, utils::BooleanStream * bs) [virtual]`

Tight Un-marshal to the given stream.

Parameters

<i>format</i>	- The OpenwireFormat properties
<i>command</i>	- the object to Un-Marshal
<i>dis</i>	- the DataInputStream to Un-Marshal from
<i>bs</i>	- boolean stream to unmarshal from.

Exceptions

<i>IOException</i>	if an error occurs.
--------------------	---------------------

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1131).

The documentation for this class was generated from the following file:

- src/main/activemq/wireformat/openwire/marshal/generated/**WireFormatInfo-Marshaller.h**

6.595 activemq::wireformat::WireFormatNegotiator Class Reference

Defines a **WireFormatNegotiator** (p. 2904) which allows a **WireFormat** (p. 2884) to.

```
#include <src/main/activemq/wireformat/WireFormatNegotiator.h>
```

Inheritance diagram for activemq::wireformat::WireFormatNegotiator:

Public Member Functions

- **WireFormatNegotiator** (const **Pointer**< **transport::Transport** > &next)
*Creates a new instance of a **WireFormat** (p. 2884) Negotiator wrapping the Transport passed.*
- virtual ~**WireFormatNegotiator** ()

6.595.1 Detailed Description

Defines a **WireFormatNegotiator** (p. 2904) which allows a **WireFormat** (p. 2884) to.

6.595.2 Constructor & Destructor Documentation

6.595.2.1 **activemq::wireformat::WireFormatNegotiator::WireFormatNegotiator** (const **Pointer**< **transport::Transport** > &next) [inline]

Creates a new instance of a **WireFormat** (p. 2884) Negotiator wrapping the Transport passed.

Parameters

<i>next</i>	The next Transport in the chain
-------------	---------------------------------

6.595.2.2 virtual **activemq::wireformat::WireFormatNegotiator::~~WireFormatNegotiator** () [inline, virtual]

The documentation for this class was generated from the following file:

- src/main/activemq/wireformat/**WireFormatNegotiator.h**

6.596 **activemq::wireformat::WireFormatRegistry** Class Reference

Registry of all **WireFormat** (p. 2884) Factories that are available to the client at runtime.

```
#include <src/main/activemq/wireformat/WireFormatRegistry.h>
```

Public Member Functions

- virtual **~WireFormatRegistry** ()
- **WireFormatFactory** * **findFactory** (const std::string &name) const
*Gets a Registered **WireFormatFactory** (p. 2888) from the Registry and returns it if there is not a registered format factory with the given name an exception is thrown.*
- void **registerFactory** (const std::string &name, **WireFormatFactory** *factory)
*Registers a new **WireFormatFactory** (p. 2888) with this Registry.*
- void **unregisterFactory** (const std::string &name)
Unregisters the Factory with the given name and deletes that instance of the Factory.
- void **unregisterAllFactories** ()
Removes all Factories and deletes the instances of the Factory objects.
- std::vector< std::string > **getWireFormatNames** () const
*Retrieves a list of the names of all the Registered **WireFormat** (p. 2884)'s in this - Registry.*

Static Public Member Functions

- static **WireFormatRegistry** & **getInstance** ()
*Gets the single instance of the **WireFormatRegistry** (p. 2905).*

6.596.1 Detailed Description

Registry of all **WireFormat** (p. 2884) Factories that are available to the client at runtime.

New **WireFormat** (p. 2884)'s must have a factory registered here before a connection attempt is made.

Since

3.0

6.596.2 Constructor & Destructor Documentation

6.596.2.1 virtual **activemq::wireformat::WireFormatRegistry::~~WireFormatRegistry**() [virtual]

6.596.3 Member Function Documentation

6.596.3.1 **WireFormatFactory*** **activemq::wireformat::WireFormatRegistry::findFactory**(const std::string & *name*) const

Gets a Registered **WireFormatFactory** (p. 2888) from the Registry and returns it if there is not a registered format factory with the given name an exception is thrown.

Parameters

<i>name</i>	The name of the Factory to find in the Registry.
-------------	--

Returns

the Factory registered under the given name.

Exceptions

<i>NoSuchElementException</i>	if no factory is registered with that name.
-------------------------------	---

6.596.3.2 static **WireFormatRegistry&** **activemq::wireformat::WireFormatRegistry::getInstance**() [static]

Gets the single instance of the **WireFormatRegistry** (p. 2905).

Returns

reference to the single instance of this Registry

6.596.3.3 **std::vector<std::string>** **activemq::wireformat::WireFormatRegistry::getWireFormatNames**() const

Retrieves a list of the names of all the Registered **WireFormat** (p. 2884)'s in this - Registry.

Returns

stl vector of strings with all the **WireFormat** (p. 2884) names registered.

6.596.3.4 `void activemq::wireformat::WireFormatRegistry::registerFactory (const std::string & name, WireFormatFactory * factory)`

Registers a new **WireFormatFactory** (p. 2888) with this Registry.

If a Factory with the given name is already registered it is overwritten with the new one. Once a factory is added to the Registry its lifetime is controlled by the Registry, it will be deleted once the Registry has been deleted.

Parameters

<i>name</i>	The name of the new Factory to register.
<i>factory</i>	The new Factory to add to the Registry.

Exceptions

<i>IllegalArgument-Exception</i>	is name is the empty string.
<i>NullPointerException</i>	if the Factory is Null.

6.596.3.5 `void activemq::wireformat::WireFormatRegistry::unregisterAllFactories ()`

Removes all Factories and deletes the instances of the Factory objects.

6.596.3.6 `void activemq::wireformat::WireFormatRegistry::unregisterFactory (const std::string & name)`

Unregisters the Factory with the given name and deletes that instance of the Factory.

Parameters

<i>name</i>	Name of the Factory to unregister and destroy
-------------	---

The documentation for this class was generated from the following file:

- `src/main/activemq/wireformat/WireFormatRegistry.h`

6.597 activemq::transport::inactivity::WriteChecker Class Reference

Runnable class used by the {.

```
#include <src/main/activemq/transport/inactivity/Write-Checker.h>
```

Inheritance diagram for `activemq::transport::inactivity::WriteChecker`:

Public Member Functions

- **WriteChecker** (**InactivityMonitor** *parent)
- virtual ~**WriteChecker** ()
- virtual void **run** ()

Run method - called by the Thread class in the context of the thread.

6.597.1 Detailed Description

Runnable class used by the {}.

See also

InactivityMonitor (p. 1425)} to make periodic writes to the underlying **transport** (p. 71) if no other write activity is going on in order to more quickly detect failures of the connection to the broker.

Since

3.1.0

6.597.2 Constructor & Destructor Documentation

6.597.2.1 `activemq::transport::inactivity::WriteChecker::WriteChecker (InactivityMonitor *parent)`

6.597.2.2 `virtual activemq::transport::inactivity::WriteChecker::~~WriteChecker ()`
[virtual]

6.597.3 Member Function Documentation

6.597.3.1 `virtual void activemq::transport::inactivity::WriteChecker::run ()`
[virtual]

Run method - called by the Thread class in the context of the thread.

Implements **decaf::lang::Runnable** (p. 2312).

The documentation for this class was generated from the following file:

- src/main/activemq/transport/inactivity/**WriteChecker.h**

6.598 decaf::io::Writer Class Reference

```
#include <src/main/decaf/io/Writer.h>
```

Inheritance diagram for decaf::io::Writer:

Public Member Functions

- **Writer** ()
- virtual **~Writer** ()
- virtual void **write** (char v)
Writes an single byte char value.
- virtual void **write** (const std::vector< char > &buffer)
Writes an array of Chars.
- virtual void **write** (const char *buffer, int size)
Writes a byte array to the output stream.
- virtual void **write** (const char *buffer, int size, int offset, int length)
Writes a byte array to the output stream.
- virtual void **write** (const std::string &str)
Writes a string.
- virtual void **write** (const std::string &str, int offset, int length)
Writes a string.
- virtual **decaf::lang::Appendable** & **append** (char value)
Appends the specified character to this Appendable.
- virtual **decaf::lang::Appendable** & **append** (const **decaf::lang::CharSequence** *csq)
Appends the specified character sequence to this Appendable.
- virtual **decaf::lang::Appendable** & **append** (const **decaf::lang::CharSequence** *csq, int start, int end)
Appends a subsequence of the specified character sequence to this Appendable.

Protected Member Functions

- virtual void **doWriteArrayBounded** (const char *buffer, int size, int offset, int length)=0
Override this method to customize the functionality of the method write(char buffer, int size, int offset, int length).*
- virtual void **doWriteChar** (char v)
- virtual void **doWriteVector** (const std::vector< char > &buffer)
- virtual void **doWriteArray** (const char *buffer, int size)
- virtual void **doWriteString** (const std::string &str)
- virtual void **doWriteStringBounded** (const std::string &str, int offset, int length)
- virtual **decaf::lang::Appendable** & **doAppendChar** (char value)
- virtual **decaf::lang::Appendable** & **doAppendCharSequence** (const **decaf::lang::CharSequence** *csq)
- virtual **decaf::lang::Appendable** & **doAppendCharSequenceStartEnd** (const **decaf::lang::CharSequence** *csq, int start, int end)

6.598.1 Constructor & Destructor Documentation

6.598.1.1 `decaf::io::Writer::Writer ()`

6.598.1.2 `virtual decaf::io::Writer::~~Writer ()` [virtual]

6.598.2 Member Function Documentation

6.598.2.1 `virtual decaf::lang::Appendable& decaf::io::Writer::append (char value)`
[virtual]

Appends the specified character to this Appendable.

Parameters

<i>value</i>	The character to append.
--------------	--------------------------

Returns

a Reference to this Appendable

Exceptions

<i>Exception</i>	if an error occurs.
------------------	---------------------

Implements **decaf::lang::Appendable** (p. 443).

6.598.2.2 `virtual decaf::lang::Appendable& decaf::io::Writer::append (const decaf::lang::CharSequence * csq)` [virtual]

Appends the specified character sequence to this Appendable.

Parameters

<i>csq</i>	The character sequence from which a subsequence will be appended. If <i>csq</i> is NULL, then characters will be appended as if <i>csq</i> contained the string "null".
------------	---

Returns

a Reference to this Appendable.

Exceptions

<i>Exception</i>	if an error occurs.
------------------	---------------------

Implements **decaf::lang::Appendable** (p. 443).

6.598.2.3 **virtual decaf::lang::Appendable& decaf::io::Writer::append (const decaf::lang::CharSequence * *csq*, int *start*, int *end*)** [virtual]

Appends a subsequence of the specified character sequence to this Appendable.

Parameters

<i>csq</i>	- The character sequence from which a subsequence will be appended. If <i>csq</i> is NULL, then characters will be appended as if <i>csq</i> contained the string "null".
<i>start</i>	The index of the first character in the subsequence.
<i>end</i>	The index of the character following the last character in the subsequence.

Returns

a Reference to this Appendable

Exceptions

<i>Exception</i>	if an error occurs.
<i>IndexOutOfBoundsException</i>	<i>start</i> is greater than <i>end</i> , or <i>end</i> is greater than <i>csq.length()</i>

Implements **decaf::lang::Appendable** (p. 443).

6.598.2.4 **virtual decaf::lang::Appendable& decaf::io::Writer::doAppendChar (char *value*)** [protected, virtual]

6.598.2.5 **virtual decaf::lang::Appendable& decaf::io::Writer::doAppendCharSequence (const decaf::lang::CharSequence * *csq*)** [protected, virtual]

6.598.2.6 **virtual decaf::lang::Appendable& decaf::io::Writer::doAppendCharSequenceStartEnd (const decaf::lang::CharSequence * *csq*, int *start*, int *end*)** [protected, virtual]

6.598.2.7 **virtual void decaf::io::Writer::doWriteArray (const char * *buffer*, int *size*)** [protected, virtual]

6.598.2.8 **virtual void decaf::io::Writer::doWriteArrayBounded (const char * *buffer*, int *size*, int *offset*, int *length*)** [protected, pure virtual]

Override this method to customize the functionality of the method `write(char* buffer, int size, int offset, int length)`.

All subclasses must override this method to provide the basic **Writer** (p. 2908) functionality.

Implemented in **decaf::io::OutputStreamWriter** (p. 2075).

6.598.2.9 virtual void **decaf::io::Writer::doWriteChar** (char *v*) [protected, virtual]

6.598.2.10 virtual void **decaf::io::Writer::doWriteString** (const std::string & *str*) [protected, virtual]

6.598.2.11 virtual void **decaf::io::Writer::doWriteStringBounded** (const std::string & *str*, int *offset*, int *length*) [protected, virtual]

6.598.2.12 virtual void **decaf::io::Writer::doWriteVector** (const std::vector< char > & *buffer*) [protected, virtual]

6.598.2.13 virtual void **decaf::io::Writer::write** (char *v*) [virtual]

Writes an single byte char value.

Parameters

<i>v</i>	The value to be written.
----------	--------------------------

Exceptions

<i>IOException</i> (p. 1545)	thrown if an error occurs.
--	----------------------------

6.598.2.14 virtual void **decaf::io::Writer::write** (const std::vector< char > & *buffer*) [virtual]

Writes an array of Chars.

Parameters

<i>buffer</i>	The array to be written.
---------------	--------------------------

Exceptions

<i>IOException</i> (p. 1545)	thrown if an error occurs.
--	----------------------------

6.598.2.15 virtual void **decaf::io::Writer::write** (const char * *buffer*, int *size*) [virtual]

Writes a byte array to the output stream.

Parameters

<i>buffer</i>	The byte array to write (cannot be NULL).
<i>size</i>	The size in bytes of the buffer passed.

Exceptions

<i>IOException</i> (p. 1545)	if an I/O error occurs.
<i>NullPointerException</i>	if buffer is NULL.

6.598.2.16 virtual void **decaf::io::Writer::write** (const char * *buffer*, int *size*, int *offset*, int *length*) [virtual]

Writes a byte array to the output stream.

Parameters

<i>buffer</i>	The byte array to write (cannot be NULL).
<i>size</i>	The size in bytes of the buffer passed.
<i>offset</i>	The position in the array to start writing from.
<i>length</i>	The number of bytes in the array to write.

Exceptions

<i>IOException</i> (p. 1545)	if an I/O error occurs.
<i>NullPointerException</i>	if buffer is NULL.
<i>IndexOutOfBoundsException</i>	if offset + length > size of the buffer.

6.598.2.17 virtual void **decaf::io::Writer::write** (const std::string & *str*) [virtual]

Writes a string.

Parameters

<i>str</i>	The string to be written.
------------	---------------------------

Exceptions

<i>IOException</i> (p. 1545)	thrown if an error occurs.
--	----------------------------

6.598.2.18 virtual void decaf::io::Writer::write (const std::string & *str*, int *offset*, int *length*) [virtual]

Writes a string.

Parameters

<i>str</i>	The string to be written.
<i>offset</i>	The position in the array to start writing from.
<i>length</i>	The number of bytes in the array to write.

Exceptions

<i>IOException</i> (p. 1545)	thrown if an error occurs.
<i>IndexOutOfBounds-Exception</i>	if offset+length is greater than the string length.

The documentation for this class was generated from the following file:

- src/main/decaf/io/**Writer.h**

6.599 decaf::security::auth::x500::X500Principal Class Reference

```
#include <src/main/decaf/security/auth/x500/X500Principal.h>
```

Inheritance diagram for decaf::security::auth::x500::X500Principal:

Public Member Functions

- virtual ~**X500Principal** ()
- virtual std::string **getName** () const =0
Provides the name of this principal.
- virtual void **getEncoded** (std::vector< unsigned char > &output) const =0
- virtual int **hashCode** () const =0

6.599.1 Constructor & Destructor Documentation

6.599.1.1 virtual decaf::security::auth::x500::X500Principal::~X500Principal ()
[inline, virtual]

6.599.2 Member Function Documentation

- 6.599.2.1 `virtual void decaf::security::auth::x500::X500Principal::getEncoded (std::vector< unsigned char > & output) const` [pure virtual]
- 6.599.2.2 `virtual std::string decaf::security::auth::x500::X500Principal::getName () const` [pure virtual]

Provides the name of this principal.

Returns

the name of this principal.

Implements **decaf::security::Principal** (p. 2160).

- 6.599.2.3 `virtual int decaf::security::auth::x500::X500Principal::hashCode () const` [pure virtual]

The documentation for this class was generated from the following file:

- `src/main/decaf/security/auth/x500/X500Principal.h`

6.600 decaf::security::cert::X509Certificate Class Reference

Base interface for all identity certificates.

```
#include <src/main/decaf/security/cert/X509Certificate.h>
```

Inheritance diagram for `decaf::security::cert::X509Certificate`:

Public Member Functions

- `virtual ~X509Certificate ()`
- `virtual void checkValidity () const =0`
- `virtual void checkValidity (const decaf::util::Date &date) const =0`
- `virtual int getBasicConstraints () const =0`
- `virtual void getIssuerUniqueID (std::vector< bool > &output) const =0`
- `virtual const X500Principal * getIssuerX500Principal () const =0`
- `virtual void getKeyUsage (std::vector< unsigned char > &output) const =0`
- `virtual Date getNotAfter () const =0`
- `virtual Date getNotBefore () const =0`
- `virtual std::string getSigAlgName () const =0`
- `virtual std::string getSigAlgOID () const =0`
- `virtual void getSigAlgParams (std::vector< unsigned char > &output) const =0`

- virtual void **getSignature** (std::vector< unsigned char > &output) const =0
- virtual void **getSubjectUniqueID** (std::vector< bool > &output) const =0
- virtual const X500Principal * **getSubjectX500Principal** () const =0
- virtual void **getTBSCertificate** (std::vector< unsigned char > &output) const =0
- virtual int **getVersion** () const =0

6.600.1 Detailed Description

Base interface for all identity certificates.

6.600.2 Constructor & Destructor Documentation

6.600.2.1 virtual decaf::security::cert::X509Certificate::~X509Certificate ()
[inline, virtual]

6.600.3 Member Function Documentation

6.600.3.1 virtual void decaf::security::cert::X509Certificate::checkValidity () const
[pure virtual]

6.600.3.2 virtual void decaf::security::cert::X509Certificate::checkValidity (const
decaf::util::Date & date) const [pure virtual]

6.600.3.3 virtual int decaf::security::cert::X509Certificate::getBasicConstraints ()
const [pure virtual]

6.600.3.4 virtual void decaf::security::cert::X509Certificate::getIssuerUniqueID (
std::vector< bool > & output) const [pure virtual]

6.600.3.5 virtual const X500Principal* decaf::security::cert::X509-
Certificate::getIssuerX500Principal () const [pure
virtual]

6.600.3.6 virtual void decaf::security::cert::X509Certificate::getKeyUsage (
std::vector< unsigned char > & output) const [pure virtual]

6.600.3.7 virtual Date decaf::security::cert::X509Certificate::getNotAfter () const
[pure virtual]

6.600.3.8 virtual Date decaf::security::cert::X509Certificate::getNotBefore () const
[pure virtual]

6.600.3.9 virtual std::string decaf::security::cert::X509Certificate::getSigAlgName (
) const [pure virtual]

- 6.600.3.10 `virtual std::string decaf::security::cert::X509Certificate::getSigAlgOID () const [pure virtual]`
- 6.600.3.11 `virtual void decaf::security::cert::X509Certificate::getSigAlgParams (std::vector< unsigned char > & output) const [pure virtual]`
- 6.600.3.12 `virtual void decaf::security::cert::X509Certificate::getSignature (std::vector< unsigned char > & output) const [pure virtual]`
- 6.600.3.13 `virtual void decaf::security::cert::X509Certificate::getSubjectUniqueID (std::vector< bool > & output) const [pure virtual]`
- 6.600.3.14 `virtual const X500Principal* decaf::security::cert::X509Certificate::getSubjectX500Principal () const [pure virtual]`
- 6.600.3.15 `virtual void decaf::security::cert::X509Certificate::getTBSCertificate (std::vector< unsigned char > & output) const [pure virtual]`
- 6.600.3.16 `virtual int decaf::security::cert::X509Certificate::getVersion () const [pure virtual]`

The documentation for this class was generated from the following file:

- `src/main/decaf/security/cert/X509Certificate.h`

6.601 cms::XAConnection Class Reference

The **XAConnection** (p. 2917) interface defines an extended **Connection** (p. 933) type that is used to create **XASession** (p. 2935) objects.

```
#include <src/main/cms/XAConnection.h>
```

Inheritance diagram for cms::XAConnection:

Public Member Functions

- `virtual ~XAConnection () throw ()`
- `virtual XASession * createXASession ()=0`
Creates an XASession (p. 2935) object.

6.601.1 Detailed Description

The **XAConnection** (p. 2917) interface defines an extended **Connection** (p. 933) type that is used to create **XASession** (p. 2935) objects.

This is an optional interface and CMS providers are allowed to omit an implementation and instead throw an exception from an **XAConnectionFactory** (p. 2918) stub to indicate that XA is not supported.

Since

2.3

6.601.2 Constructor & Destructor Documentation

6.601.2.1 virtual cms::XAConnection::~XAConnection () throw () [virtual]

6.601.3 Member Function Documentation

6.601.3.1 virtual XASession* cms::XAConnection::createXASession () [pure virtual]

Creates an **XASession** (p. 2935) object.

Returns

a newly created **XASession** (p. 2935) instance, caller owns the pointer.

Exceptions

CMSException (p. 826)	If the XAConnection (p. 2917) object fails to create the XASession (p. 2935) instance due to an internal error.
---------------------------------	---

Implemented in **activemq::core::ActiveMQXAConnection** (p. 433).

The documentation for this class was generated from the following file:

- src/main/cms/**XAConnectionFactory.h**

6.602 cms::XAConnectionFactory Class Reference

The **XAConnectionFactory** (p. 2918) interface is specialized interface that defines an **ConnectionFactory** (p. 955) that creates **Connection** (p. 933) instance that will participate in XA Transactions.

```
#include <src/main/cms/XAConnectionFactory.h>
```

Inheritance diagram for cms::XAConnectionFactory:

Public Member Functions

- virtual **~XAConnectionFactory** ()
- virtual **XAConnection * createXAConnection** ()=0
*Creates an **XAConnection** (p. 2917) with the default user name and password.*
- virtual **XAConnection * createXAConnection** (const std::string &userName, const std::string &password)=0
Creates an XA connection with the specified user name and password.

Static Public Member Functions

- static **XAConnectionFactory * createCMSXAConnectionFactory** (const std::string &brokerURI)
*Static method that is used to create a provider specific XA **Connection** (p. 933) factory.*

6.602.1 Detailed Description

The **XAConnectionFactory** (p. 2918) interface is specialized interface that defines an **ConnectionFactory** (p. 955) that creates **Connection** (p. 933) instance that will participate in XA Transactions.

Some application provide support for grouping XA capable resource use into a distributed transaction (optional). To include CMS API transactions in a XA transaction, an application requires a XA aware library. A CMS provider exposes its XA support using an **XAConnectionFactory** (p. 2918) object, which an application uses to create **XAConnection** (p. 2917) objects.

The **XAConnectionFactory** (p. 2918) interface is optional. CMS providers are not required to support this interface. This interface is for use by CMS providers to support transactional environments. Client programs are strongly encouraged to use the transactional support available in their environment, rather than use these XA interfaces directly.

Since

2.3

6.602.2 Constructor & Destructor Documentation

- 6.602.2.1 virtual **cms::XAConnectionFactory::~XAConnectionFactory** ()
 [virtual]

6.602.3 Member Function Documentation

6.602.3.1 static **XAConnectionFactory*** cms::XAConnectionFactory-
::createCMSXAConnectionFactory (const std::string & brokerURI)
[static]

Static method that is used to create a provider specific XA **Connection** (p. 933) factory.

The provider implements this method in their library and returns an instance of a **XA-ConnectionFactory** (p. 2918) derived object. Clients can use this method to remain abstracted from the specific CMS implementation being used.

The XA interfaces are optional in CMS however if a provider chooses to omit them it should still override this method and throw an **UnsupportedOperationException** (p. 2827) to indicate that it doesn't provide this functionality.

Parameters

<i>brokerURI</i>	The remote address to use to connect to the Provider.
------------------	---

Returns

A pointer to a provider specific implementation of the **XAConnectionFactory** (p. 2918) interface, the caller is responsible for deleting this resource.

Exceptions

CMSException (p. 826)	if an internal error occurs while creating the XAConnectionFactory (p. 2918).
Unsupported-OperationException (p. 2827)	if the provider does not support the XA API.

6.602.3.2 virtual **XAConnection*** cms::XAConnectionFactory::createXA-
Connection () [pure virtual]

Creates an **XAConnection** (p. 2917) with the default user name and password.

The connection is created in stopped mode just as the standard **Connection** (p. 933) object is created from the **ConnectionFactory** (p. 955). No messages will be delivered until the **Connection.start** (p. 2534) method is explicitly called.

Returns

a new **XAConnectionFactory** (p. 2918) instance, the caller owns the returned pointer.

Exceptions

CMSException (p. 826)	if an internal error occurs while creating the Connection (p. 933).
---------------------------------	--

<i>CMSSecurity-Exception</i> (p. 835)	if the client authentication fails because the user name or password are invalid.
---	---

Implemented in **activemq::core::ActiveMQXAConnectionFactory** (p. 435).

6.602.3.3 **virtual XAConnection* cms::XAConnectionFactory::createXA-Connection (const std::string & *userName*, const std::string & *password*)**
[pure virtual]

Creates an XA connection with the specified user name and password.

The connection is created in stopped mode just as the standard **ConnectionFactory** (p. 955) creates a new **Connection** (p. 933). No messages will be delivered until the **Connection.start** (p. 2534) method is explicitly called.

Returns

a new **XAConnectionFactory** (p. 2918) instance, the caller owns the returned pointer.

Exceptions

<i>CMSException</i> (p. 826)	if an internal error occurs while creating the Connection (p. 933).
<i>CMSSecurity-Exception</i> (p. 835)	if the client authentication fails because the user name or password are invalid.

Implemented in **activemq::core::ActiveMQXAConnectionFactory** (p. 436).

The documentation for this class was generated from the following file:

- src/main/cms/**XAConnectionFactory.h**

6.603 cms::XAException Class Reference

The **XAException** (p. 2921) is thrown by the Resource Manager (RM) to inform the Transaction Manager of an error encountered by the involved transaction.

```
#include <src/main/cms/XAException.h>
```

Inheritance diagram for cms::XAException:

Public Member Functions

- **XAException** ()
- **XAException** (int errorCode)
- **XAException** (const **XAException** &ex)
- **XAException** (const std::string &message)
- **XAException** (const std::string &message, const std::exception *cause)
- **XAException** (const std::string &message, const std::exception *cause, const std::vector< std::pair< std::string, int > > &stackTrace)
- virtual ~**XAException** () throw ()
- void **setErrorCode** (int errorCode)
*Sets the error code for this **XAException** (p. 2921).*
- int **getErrorCode** () const
*Gets the error code that was assigned to this **XAException** (p. 2921).*

Static Public Attributes

- static const int **XA_RBBASE**
Code which contains the inclusive lower bound of the rollback error codes.
- static const int **XA_RBROLLBACK**
Code which means that the rollback occurred for an unspecified reason.
- static const int **XA_RBCOMMFAIL**
Code which means that rollback was caused by a communication failure.
- static const int **XA_RBDEADLOCK**
Code which means that a failure occurred because a deadlock was detected.
- static const int **XA_RBINTEGRITY**
Code which means that a condition was detected than implies a violation of the integrity of the resource.
- static const int **XA_RBOTHER**
Code which means that the Resource Manager rolled back the transaction branch for a reason not separately listed.
- static const int **XA_RBPROTO**
Code which means that a protocol error occurred in the Resource Manager.
- static const int **XA_RBTIMEOUT**
Code which means that a transaction branch took too long.
- static const int **XA_RBTRANSIENT**
Code which means that the caller may retry the transaction branch.
- static const int **XA_RBEND**
Code which contains the inclusive upper bound of the rollback error codes.
- static const int **XA_NOMIGRATE**
Code which means that resumption must occur where the suspension occurred.
- static const int **XA_HEURHAZ**
Code which means that the transaction branch may have been heuristically completed.
- static const int **XA_HEURCOM**

Code which means that the transaction branch has been heuristically committed.

- static const int **XA_HEURRB**

Code which means that the transaction branch has been heuristically rolled back.

- static const int **XA_HEURMIX**

Code which means that the transaction branch has been heuristically committed and rolled back.

- static const int **XA_RETRY**

Code which means that the method returned with no effect and can be reissued.

- static const int **XA_RDONLY**

Code which means that the transaction branch was read only and has been committed.

- static const int **XAER_ASYNC**

Code which means that there is already an asynchronous operation outstanding.

- static const int **XAER_RMERR**

Code which means that a Resource Manager error has occurred for the transaction branch.

- static const int **XAER_NOTA**

Code which means that the XID is not valid.

- static const int **XAER_INVALID**

Code which means that invalid arguments were supplied.

- static const int **XAER_PROTO**

Code which means that the method was invoked in an improper context.

- static const int **XAER_RMFAIL**

Code which means that the Resource Manager is unavailable.

- static const int **XAER_DUPID**

Code which means that the XID already exists.

- static const int **XAER_OUTSIDE**

Work is being done by the Resource Manager outside the boundaries of a global transaction.

6.603.1 Detailed Description

The **XAException** (p.2921) is thrown by the Resource Manager (RM) to inform the Transaction Manager of an error encountered by the involved transaction.

Since

2.3

6.603.2 Constructor & Destructor Documentation

6.603.2.1 **cms::XAException::XAException ()**

6.603.2.2 **cms::XAException::XAException (int errorCode)**

6.603.2.3 `cms::XAException::XAException (const XAException & ex)`

6.603.2.4 `cms::XAException::XAException (const std::string & message)`

6.603.2.5 `cms::XAException::XAException (const std::string & message, const std::exception * cause)`

6.603.2.6 `cms::XAException::XAException (const std::string & message, const std::exception * cause, const std::vector< std::pair< std::string, int > > & stackTrace)`

6.603.2.7 `virtual cms::XAException::~XAException () throw ()` `[virtual]`

6.603.3 Member Function Documentation

6.603.3.1 `int cms::XAException::getErrorCode () const` `[inline]`

Gets the error code that was assigned to this **XAException** (p. 2921).

Returns

the assigned error code.

6.603.3.2 `void cms::XAException::setErrorCode (int errorCode)` `[inline]`

Sets the error code for this **XAException** (p. 2921).

Parameters

<code>errorCode</code>	The error code to assign to this XAException (p. 2921).
------------------------	--

6.603.4 Field Documentation

6.603.4.1 `const int cms::XAException::XA_HEURCOM` `[static]`

Code which means that the transaction branch has been heuristically committed.

6.603.4.2 `const int cms::XAException::XA_HEURHAZ` `[static]`

Code which means that the transaction branch may have been heuristically completed.

6.603.4.3 `const int cms::XAException::XA_HEURMIX` `[static]`

Code which means that the transaction branch has been heuristically committed and rolled back.

6.603.4.4 `const int cms::XAException::XA_HEURRB` `[static]`

Code which means that the transaction branch has been heuristically rolled back.

6.603.4.5 `const int cms::XAException::XA_NOMIGRATE` `[static]`

Code which means that resumption must occur where the suspension occurred.

6.603.4.6 `const int cms::XAException::XA_RBBASE` `[static]`

Code which contains the inclusive lower bound of the rollback error codes.

6.603.4.7 `const int cms::XAException::XA_RBCOMMFAIL` `[static]`

Code which means that rollback was caused by a communication failure.

6.603.4.8 `const int cms::XAException::XA_RBDEADLOCK` `[static]`

Code which means that a failure occurred because a deadlock was detected.

6.603.4.9 `const int cms::XAException::XA_RBEND` `[static]`

Code which contains the inclusive upper bound of the rollback error codes.

6.603.4.10 `const int cms::XAException::XA_RBINTEGRITY` `[static]`

Code which means that a condition was detected than implies a violation of the integrity of the resource.

6.603.4.11 `const int cms::XAException::XA_RBOTHER` `[static]`

Code which means that the Resource Manager rolled back the transaction branch for a reason not separately listed.

6.603.4.12 `const int cms::XAException::XA_RBPROTO` `[static]`

Code which means that a protocol error occurred in the Resource Manager.

6.603.4.13 `const int cms::XAException::XA_RBROLLBACK` `[static]`

Code which means that the rollback occurred for an unspecified reason.

6.603.4.14 `const int cms::XAException::XA_RBTIMEOUT` `[static]`

Code which means that a transaction branch took too long.

6.603.4.15 `const int cms::XAException::XA_RBTRANSIENT` `[static]`

Code which means that the caller may retry the transaction branch.

6.603.4.16 `const int cms::XAException::XA_RDONLY` `[static]`

Code which means that the transaction branch was read only and has been committed.

6.603.4.17 `const int cms::XAException::XA_RETRY` `[static]`

Code which means that the method returned with no effect and can be reissued.

6.603.4.18 `const int cms::XAException::XAER_ASYNC` `[static]`

Code which means that there is already an asynchronous operation outstanding.

6.603.4.19 `const int cms::XAException::XAER_DUPID` `[static]`

Code which means that the XID already exists.

6.603.4.20 `const int cms::XAException::XAER_INVALID` `[static]`

Code which means that invalid arguments were supplied.

6.603.4.21 `const int cms::XAException::XAER_NOTA` `[static]`

Code which means that the XID is not valid.

6.603.4.22 `const int cms::XAException::XAER_OUTSIDE` `[static]`

Work is being done by the Resource Manager outside the boundaries of a global transaction.

6.603.4.23 `const int cms::XAException::XAER_PROTO` `[static]`

Code which means that the method was invoked in an improper context.

6.603.4.24 `const int cms::XAException::XAER_RMERR` `[static]`

Code which means that a Resource Manager error has occurred for the transaction branch.

6.603.4.25 `const int cms::XAException::XAER_RMFAIL` `[static]`

Code which means that the Resource Manager is unavailable.

The documentation for this class was generated from the following file:

- `src/main/cms/XAException.h`

6.604 `cms::XAResource` Class Reference

The **XAResource** (p. 2926) interface is a CMS mapping of the industry standard XA interface based on the X/Open CAE Specification (Distributed Transaction Processing: The XA Specification).

```
#include <src/main/cms/XAResource.h>
```

Inheritance diagram for `cms::XAResource`:

Public Member Functions

- virtual `~XAResource` ()
- virtual void **commit** (const **Xid** *xid, bool onePhase)=0
Commits a global transaction.
- virtual void **end** (const **Xid** *xid, int flags)=0
Ends the work done for a transaction branch.
- virtual void **forget** (const **Xid** *xid)=0
Informs the Resource Manager that it can forget about a specified transaction branch.
- virtual int **getTransactionTimeout** () const =0
*Gets the transaction timeout value for this **XAResource** (p. 2926).*
- virtual bool **isSameRM** (const **XAResource** *theXAResource)=0
*Returns true if the ResourceManager for this **XAResource** (p. 2926) is the same as the Resource Manager for a supplied **XAResource** (p. 2926).*
- virtual int **prepare** (const **Xid** *xid)=0
Requests the Resource manager to prepare to commit a specified transaction.
- virtual int **recover** (int flag, **Xid** **recovered)=0
Get a list of prepared transaction branches.
- virtual void **rollback** (const **Xid** *xid)=0
Requests the Resource Manager to rollback a specified transaction branch.

- virtual bool **setTransactionTimeout** (int seconds)=0
*Sets the transaction timeout value for this **XAResource** (p. 2926).*
- virtual void **start** (const **Xid** *xid, int flags)=0
Starts work for a specified transaction branch.

Static Public Attributes

- static const int **TMENDRSCAN**
Flag to end a recovery scan.
- static const int **TMFAIL**
Flag to indicate that the caller is dissociation from a transaction branch and that it should be marked rollback only.
- static const int **TMJOIN**
Flag to indicate that the caller is joining an existing transaction branch.
- static const int **TMNOFLAGS**
Flag that indicates that no flags options are selected.
- static const int **TMONEPHASE**
Flag that indicates the caller is using one-phase commit optimization.
- static const int **TMRESUME**
Flag that indicates the caller is resuming association with a suspended transaction branch.
- static const int **TMSTARTRSCAN**
Flag that indicates the start of a recovery scan.
- static const int **TMSUCCESS**
Flag that indicates the caller is dissociating from a transaction branch.
- static const int **TMSUSPEND**
Flag that indicates that the caller is suspending (not terminating) its association with a transaction branch.
- static const int **XA_RDONLY**
Flag that indicates that transaction work has been read only and has been committed normally.
- static const int **XA_OK**
Flag that indicates that transaction work has been Prepared normally.

6.604.1 Detailed Description

The **XAResource** (p. 2926) interface is a CMS mapping of the industry standard XA interface based on the X/Open CAE Specification (Distributed Transaction Processing: The XA Specification).

The XA interface defines the contract between a Resource Manager and a Transaction Manager in a distributed transaction processing (DTP) environment. A CMS provider implements this interface to support the association between a global transaction and a message broker connection.

The **XAResource** (p.2926) is exposed to CMS client so that they can proxy calls from the Transaction Manager API of their choosing to the CMS provider. The CMS provider should behave and a standard XA Resource Manager its up to the client however to transmit the Transaction Manager's calls to the CMS provider through this interface.

Since

2.3

6.604.2 Constructor & Destructor Documentation

6.604.2.1 `virtual cms::XAResource::~XAResource () [virtual]`

6.604.3 Member Function Documentation

6.604.3.1 `virtual void cms::XAResource::commit (const Xid * xid, bool onePhase) [pure virtual]`

Commits a global transaction.

Parameters

<i>xid</i>	the XID which identifies the global transaction.
<i>onePhase</i>	true if the resource manager should use a one-phase commit protocol to commit the transaction.

Exceptions

XAException (p. 2921)	if an error occurred.
---------------------------------	-----------------------

Possible errors are identified by the errorcode in the **XAException** (p.2921) and include: XA_HEURHAZ, XA_HEURCOM, XA_HEURRB, XA_HEURMIX, XAER_RMERR, XAER_RMFAIL, XAER_NOTA, XAER_INVAL, or XAER_PROTO. In addition, one of the XA_RB* errors can occur if the transaction was not committed and onePhase was set to true. On completion of this method, the Resource Manager has rolled back the transaction and released resources held by the transaction.

Implemented in **activemq::core::ActiveMQTransactionContext** (p. 426).

6.604.3.2 `virtual void cms::XAResource::end (const Xid * xid, int flags) [pure virtual]`

Ends the work done for a transaction branch.

The Resource Manager disconnects the XA resource from the transaction branch and allows the transaction to complete.

Parameters

<i>xid</i>	the XID which identifies the global transaction. Should have previously been used as the parameter to a start. method.
<i>flags</i>	a flags integer - one of: XAResource::TMSUCCESS (p. 2934), - XAResource::TMFAIL (p. 2933), or XAResource::TMSUSPEND (p. 2934).

TMSUCCESS means that this section of work completed successfully.

TMFAIL means that this section of work failed. The Resource Manager can mark the transaction for rollback only.

TMSUSPEND means that this section of work is suspended and not yet complete. The associated transaction context is also suspended and must be restarted with a call to **start(Xid, int)** (p. ??) with the TMRESUME flag.

Exceptions

XAException (p. 2921)	if an error occurs. Possible error identified in the errorcode include: XAER_RMERR, XAER_RMFAIL, XAER_NOTA, XAER_INVAL, XAER_PROTO, or XA_RB*.
---------------------------------	--

Implemented in **activemq::core::ActiveMQTransactionContext** (p. 426).

6.604.3.3 `virtual void cms::XAResource::forget (const Xid * xid) [pure virtual]`

Informs the Resource Manager that it can forget about a specified transaction branch.

Parameters

<i>xid</i>	the XID which identifies the global transaction.
------------	--

Exceptions

XAException (p. 2921)	if an error occurs. Possible error identified in the errorcode include: XAER_RMERR, XAER_RMFAIL, XAER_NOTA, XAER_INVAL, or XAER_PROTO.
---------------------------------	--

Implemented in **activemq::core::ActiveMQTransactionContext** (p. 427).

6.604.3.4 `virtual int cms::XAResource::getTransactionTimeout () const [pure virtual]`

Gets the transaction timeout value for this **XAResource** (p. 2926).

The default timeout value is the default timeout value set for the Resource Manager.

Returns

the transaction timeout value for this **XAResource** (p. 2926) in seconds.

Exceptions

XAException (p. 2921)	if an error occurs. Possible error identified in the errorcode include: XAER_RMERR and XAER_RMFAIL.
---------------------------------	--

Implemented in **activemq::core::ActiveMQTransactionContext** (p. 427).

6.604.3.5 `virtual bool cms::XAResource::isSameRM (const XAResource *
theXAResource) [pure virtual]`

Returns true if the ResourceManager for this **XAResource** (p. 2926) is the same as the Resource Manager for a supplied **XAResource** (p. 2926).

Parameters

<i>theXA-Resource</i>	an XAResource (p. 2926) object
-----------------------	---------------------------------------

Returns

true if the Resource Manager for this **XAResource** (p. 2926) is the same as the Resource Manager for *theXAResource*.

Exceptions

XAException (p. 2921)	if an error occurs. Possible error identified in the errorcode include: XAER_RMERR and XAER_RMFAIL.
---------------------------------	--

Implemented in **activemq::core::ActiveMQTransactionContext** (p. 428).

6.604.3.6 `virtual int cms::XAResource::prepare (const Xid * xid) [pure
virtual]`

Requests the Resource manager to prepare to commit a specified transaction.

Parameters

<i>xid</i>	the XID which identifies the global transaction.
------------	--

Returns

an integer: XA_RDONLY or XA_OK. XA_OK implies that the transaction work has been prepared normally, XA_RDONLY implies that the transaction branch is read only and has been committed. If there is a failure which requires a rollback, an **XAException** (p. 2921) is raised.

Exceptions

XAException (p. 2921)	if an error occurs. Possible error identified in the errorcode include: XA_RB*, XAER_RMERR, XAER_RMFAIL, XAER_NOTA, XAER_INVALID, or XAER_PROTO.
---------------------------------	--

Implemented in **activemq::core::ActiveMQTransactionContext** (p. 429).

6.604.3.7 `virtual int cms::XAResource::recover (int flag, Xid ** recovered)` [pure virtual]

Get a list of prepared transaction branches.

Typically used by a transaction manager during recovery to find transaction branches that are in prepared or heuristically completed states.

Parameters

<i>flag</i>	an integer. Must be one of: XAResource::TMSTARTRSCAN (p. 2934), XAResource::TMENDRSCAN (p. 2933), XAResource::TMNOFLAGS (p. 2934).
-------------	---

Returns

an array of zero or more XIDs identifying the transaction branches in the prepared or heuristically completed states.

Exceptions

XAException (p. 2921)	if an error occurs. Possible error identified in the errorcode include: XAER_RMERR, XAER_RMFAIL, XAER_INVALID, and XAER_PROTO.
---------------------------------	--

Implemented in **activemq::core::ActiveMQTransactionContext** (p. 429).

6.604.3.8 `virtual void cms::XAResource::rollback (const Xid * xid)` [pure virtual]

Requests the Resource Manager to rollback a specified transaction branch.

Parameters

<i>xid</i>	the XID which identifies the transaction branch.
------------	--

Exceptions

<i>XAException</i> (p. 2921)	if an error occurs.
--	---------------------

Implemented in **activemq::core::ActiveMQTransactionContext** (p. 430).

6.604.3.9 **virtual bool cms::XAResource::setTransactionTimeout (int *seconds*)**
[pure virtual]

Sets the transaction timeout value for this **XAResource** (p. 2926).

If the value is set to 0, the default timeout value for the Resource Manager is used.

Parameters

<i>seconds</i>	the new Timeout value in seconds.
----------------	-----------------------------------

Returns

true if the transaction timeout value has been updated, false otherwise.

Exceptions

<i>XAException</i> (p. 2921)	if an error occurs. Possible error identified in the errorcode include: XAER_RMERR, XAER_RMFAIL, or XAER_INVALID.
--	--

Implemented in **activemq::core::ActiveMQTransactionContext** (p. 431).

6.604.3.10 **virtual void cms::XAResource::start (const Xid * *xid*, int *flags*)** [pure virtual]

Starts work for a specified transaction branch.

Parameters

<i>xid</i>	the XID which identifies the transaction branch.
<i>flags</i>	an integer. Must be one of XAResource::TMNOFLAGS (p. 2934), XAResource::TMJOIN (p. 2933), or XAResource::TMRESUME (p. 2934).

TMJOIN implies that the start applies to joining a transaction previously passed to the Resource Manager.

TMRESUME implies that the start applies to a suspended transaction that should be restarted.

If TMNOFLAGS is specified, then if the transaction has been previously seen by the Resource Manager, an **XAException** (p. 2921) is raised with the code XAER_DUPID.

Exceptions

XAException (p. 2921)	if an error occurs. Possible error identified in the errorcode include: XA_RB*, XAER_RMERR, XAER_RMFAIL, XAER_DUPID, XAER_OUTSIDE, XAER_NOTA, XAER_INVALID, or XAER_PROTO.
---------------------------------	--

Implemented in **activemq::core::ActiveMQTransactionContext** (p. 431).

6.604.4 Field Documentation

6.604.4.1 **const int cms::XAResource::TMENDRSCAN** [static]

Flag to end a recovery scan.

6.604.4.2 **const int cms::XAResource::TMFAIL** [static]

Flag to indicate that the caller is dissociation from a transaction branch and that it should be marked rollback only.

6.604.4.3 **const int cms::XAResource::TMJOIN** [static]

Flag to indicate that the caller is joining an existing transaction branch.

6.604.4.4 **const int cms::XAResource::TMNOFLAGS** [static]

Flag that indicates that no flags options are selected.
(ie a null flag)

6.604.4.5 **const int cms::XAResource::TMONEPHASE** [static]

Flag that indicates the caller is using one-phase commit optimization.

6.604.4.6 **const int cms::XAResource::TMRESUME** [static]

Flag that indicates the caller is resuming association with a suspended transaction branch.

6.604.4.7 `const int cms::XAResource::TMSTARTRSCAN` `[static]`

Flag that indicates the start of a recovery scan.

6.604.4.8 `const int cms::XAResource::TMSUCCESS` `[static]`

Flag that indicates the caller is dissociating from a transaction branch.

6.604.4.9 `const int cms::XAResource::TMSUSPEND` `[static]`

Flag that indicates that the caller is suspending (not terminating) its association with a transaction branch.

6.604.4.10 `const int cms::XAResource::XA_OK` `[static]`

Flag that indicates that transaction work has been Prepared normally.

6.604.4.11 `const int cms::XAResource::XA_RDONLY` `[static]`

Flag that indicates that transaction work has been read only and has been committed normally.

The documentation for this class was generated from the following file:

- `src/main/cms/XAResource.h`

6.605 cms::XASession Class Reference

The **XASession** (p.2935) interface extends the capability of **Session** (p.2361) by adding access to a CMS provider's support for the operating inside an XA Transaction (optional).

```
#include <src/main/cms/XASession.h>
```

Inheritance diagram for cms::XASession:

Public Member Functions

- virtual `~XASession ()` throw `()`
- virtual `XAResource * getXAResource ()` const =0

*Returns the XA resource associated with this **Session** (p. 2361) to the caller.*

6.605.1 Detailed Description

The **XASession** (p.2935) interface extends the capability of **Session** (p.2361) by adding access to a CMS provider's support for the operating inside an XA Transaction (optional).

This support takes the form of a **cms::XAResource** (p.2926) object. The functionality of this object closely resembles that defined by the standard X/Open XA Resource interface.

An application controls the transactional assignment of an **XASession** (p.2935) by obtaining its **XAResource** (p.2926). It uses the **XAResource** (p.2926) to assign the session to a transaction, prepare and commit work on the transaction, and so on.

An **XAResource** (p.2926) provides some fairly sophisticated facilities for interleaving work on multiple transactions, recovering a list of transactions in progress, and so on. A XA aware CMS provider must fully implement this functionality.

The **XASession** (p.2935) instance will behave much like a normal **cms::Session** (p.2361) however some methods will not operate as normal, any call to **Session::commit** (p.2366), or **Session::rollback** (p.2376) will result in a **CMSException** (p.826) being thrown. Also when not inside an XA transaction the **MessageConsumer** (p.1877) will operate as if it were in the AutoAcknowledge mode.

The **XASession** (p.2935) interface is optional. CMS providers are not required to support this interface. This interface is for use by CMS providers to support transactional environments. Client programs are strongly encouraged to use the transactional support available in their environment, rather than use these XA interfaces directly.

Since

2.3

6.605.2 Constructor & Destructor Documentation

6.605.2.1 `virtual cms::XASession::~~XASession () throw () [virtual]`

6.605.3 Member Function Documentation

6.605.3.1 `virtual XAResource* cms::XASession::getXAResource () const [pure virtual]`

Returns the XA resource associated with this **Session** (p.2361) to the caller.

The client can use the provided XA resource to interact with the XA Transaction Manager in use in the client application.

Returns

an XAResource instance to the caller, the caller does not own this pointer and should not delete it.

Implemented in **activemq::core::ActiveMQXASession** (p.438).

The documentation for this class was generated from the following file:

- `src/main/cms/XASession.h`

6.606 activemq::commands::XATransactionId Class Reference

```
#include <src/main/activemq/commands/XATransactionId.h>
```

Inheritance diagram for `activemq::commands::XATransactionId`:

Public Types

- typedef `decaf::lang::PointerComparator < XATransactionId > COMPART-OR`

Public Member Functions

- `XATransactionId ()`
- `XATransactionId (const XATransactionId &other)`
- `XATransactionId (const cms::Xid *xid)`
- `virtual ~XATransactionId ()`
- `virtual unsigned char getDataStructureType () const`
*Get the **DataStructure** (p. 1133) Type as defined in `CommandTypes.h`.*
- `virtual XATransactionId * cloneDataStructure () const`
Clone this object and return a new instance that the caller now owns, this will be an exact copy of this one.
- `virtual void copyDataStructure (const DataStructure *src)`
Copy the contents of the passed object into this objects members, overwriting any existing data.
- `virtual std::string toString () const`
*Returns a string containing the information for this **DataStructure** (p. 1133) such as its type and value of its elements.*
- `virtual bool equals (const DataStructure *value) const`
*Compares the **DataStructure** (p. 1133) passed in to this one, and returns if they are equivalent.*
- `virtual bool isXATransactionId () const`
- `virtual Xid * clone () const`
Creates a Copy of this Xid instance that contains the same id values.
- `virtual bool equals (const Xid *other) const`
- `virtual int getBranchQualifier (unsigned char *buffer, int size) const`
Gets the transaction branch qualifier component of the XID.
- `virtual int getGlobalTransactionId (unsigned char *buffer, int size) const`

Gets the global transaction id component of the XID.

- virtual int **getFormatId** () const

Gets the format identifier component of the XID.

- virtual void **setFormatId** (int **formatId**)
- virtual const std::vector< unsigned char > & **getGlobalTransactionId** () const
- virtual std::vector< unsigned char > & **getGlobalTransactionId** ()
- virtual void **setGlobalTransactionId** (const std::vector< unsigned char > &**globalTransactionId**)
- virtual const std::vector< unsigned char > & **getBranchQualifier** () const
- virtual std::vector< unsigned char > & **getBranchQualifier** ()
- virtual void **setBranchQualifier** (const std::vector< unsigned char > &**branchQualifier**)
- virtual int **compareTo** (const **XATransactionId** &value) const
- virtual bool **equals** (const **XATransactionId** &value) const
- virtual bool **operator==** (const **XATransactionId** &value) const
- virtual bool **operator<** (const **XATransactionId** &value) const
- **XATransactionId** & **operator=** (const **XATransactionId** &other)

Static Public Attributes

- static const unsigned char **ID_XATRANSACTIONID** = 112

Protected Attributes

- int **formatId**
- std::vector< unsigned char > **globalTransactionId**
- std::vector< unsigned char > **branchQualifier**

6.606.1 Member Typedef Documentation

- 6.606.1.1 **typedef decaf::lang::PointerComparator<XATransactionId>**
activemq::commands::XATransactionId::COMPARATOR

Reimplemented from **activemq::commands::TransactionId** (p. 2767).

6.606.2 Constructor & Destructor Documentation

- 6.606.2.1 **activemq::commands::XATransactionId::XATransactionId ()**
- 6.606.2.2 **activemq::commands::XATransactionId::XATransactionId (const XATransactionId & other)**
- 6.606.2.3 **activemq::commands::XATransactionId::XATransactionId (const cms::Xid * xid)**

6.606.2.4 `virtual activemq::commands::XATransactionId::~~XATransactionId ()`
`[virtual]`

6.606.3 Member Function Documentation

6.606.3.1 `virtual Xid* activemq::commands::XATransactionId::clone () const`
`[virtual]`

Creates a Copy of this Xid instance that contains the same id values.

Returns

a new Xid instance that is equal to this one when compared.

Implements **cms::Xid** (p. 2948).

6.606.3.2 `virtual XATransactionId* activemq::commands::XATransactionId::cloneDataStructure () const`
`[virtual]`

Clone this object and return a new instance that the caller now owns, this will be an exact copy of this one.

Returns

new copy of this object.

Reimplemented from **activemq::commands::TransactionId** (p. 2768).

6.606.3.3 `virtual int activemq::commands::XATransactionId::compareTo (const XATransactionId & value) const` `[virtual]`

6.606.3.4 `virtual void activemq::commands::XATransactionId::copyDataStructure (const DataStructure * src)` `[virtual]`

Copy the contents of the passed object into this objects members, overwriting any existing data.

Returns

src - Source Object

Reimplemented from **activemq::commands::TransactionId** (p. 2768).

6.606.3.5 `virtual bool activemq::commands::XATransactionId::equals (const DataStructure * value) const` `[virtual]`

Compares the **DataStructure** (p. 1133) passed in to this one, and returns if they are equivalent.

Equivalent here means that they are of the same type, and that each element of the objects are the same.

Returns

true if **DataStructure** (p. 1133)'s are Equal.

Reimplemented from **activemq::commands::TransactionId** (p. 2768).

6.606.3.6 `virtual bool activemq::commands::XATransactionId::equals (const Xid * other) const` [virtual]

6.606.3.7 `virtual bool activemq::commands::XATransactionId::equals (const XATransactionId & value) const` [virtual]

6.606.3.8 `virtual int activemq::commands::XATransactionId::getBranchQualifier (unsigned char * buffer, int size) const` [virtual]

Gets the transaction branch qualifier component of the XID.

The value of this Xid's branch qualifier is copied into the buffer provided. If the size of the provided buffer is not large enough to accommodate the branch qualifier then no copy is performed and the method returns -1.

Parameters

<i>buffer</i>	The location in memory to copy the qualifier bytes to.
<i>size</i>	The size of the buffer provided.

Returns

the number of bytes copied into the buffer, or -1 if the buffer provided was not large enough.

Exceptions

<i>XAException</i>	if the size parameter is less than zero or buffer is NULL.
--------------------	--

Implements **cms::Xid** (p. 2948).

6.606.3.9 `virtual const std::vector<unsigned char>& activemq::commands::XATransactionId::getBranchQualifier () const` [virtual]

6.606.3.10 `virtual std::vector<unsigned char>& activemq::commands::XATransactionId::getBranchQualifier ()` [virtual]

6.606.3.11 `virtual unsigned char activemq::commands::XA-
TransactionId::getDataStructureType () const`
[virtual]

Get the **DataStructure** (p. 1133) Type as defined in CommandTypes.h.

Returns

The type of the data structure

Reimplemented from **activemq::commands::TransactionId** (p. 2769).

6.606.3.12 `virtual int activemq::commands::XATransactionId::getFormatId () const`
[virtual]

Gets the format identifier component of the XID.

Returns

an integer containing the format identifier. 0 means the OSI CCR format.

Implements **cms::Xid** (p. 2949).

6.606.3.13 `virtual int activemq::commands::XATransactionId::get-
GlobalTransactionId (unsigned char * buffer, int size) const`
[virtual]

Gets the global transaction id component of the XID.

The value of this Xid's transaction id is copied into the buffer provided. If the size of the provided buffer is not large enough to accommodate the transaction id then no copy is performed and the method returns -1.

Parameters

<i>buffer</i>	The location in memory to copy the transaction id bytes to.
<i>size</i>	The size of the buffer provided.

Returns

the number of bytes copied into the buffer, or -1 if the buffer provided was not large enough.

Exceptions

<i>XAException</i>	if the size parameter is less than zero or buffer is NULL.
--------------------	--

Implements **cms::Xid** (p. 2949).

- 6.606.3.14 `virtual const std::vector<unsigned char>& activemq::commands::XATransactionId::getGlobalTransactionId () const` [virtual]
 - 6.606.3.15 `virtual std::vector<unsigned char>& activemq::commands::XATransactionId::getGlobalTransactionId ()` [virtual]
 - 6.606.3.16 `virtual bool activemq::commands::XATransactionId::isXATransactionId () const` [inline, virtual]
- Reimplemented from **activemq::commands::TransactionId** (p. 2769).
- 6.606.3.17 `virtual bool activemq::commands::XATransactionId::operator< (const XATransactionId & value) const` [virtual]
 - 6.606.3.18 `XATransactionId& activemq::commands::XATransactionId::operator= (const XATransactionId & other)`
 - 6.606.3.19 `virtual bool activemq::commands::XATransactionId::operator== (const XATransactionId & value) const` [virtual]
 - 6.606.3.20 `virtual void activemq::commands::XATransactionId::setBranchQualifier (const std::vector< unsigned char > & branchQualifier)` [virtual]
 - 6.606.3.21 `virtual void activemq::commands::XATransactionId::setFormatId (int formatId)` [virtual]
 - 6.606.3.22 `virtual void activemq::commands::XATransactionId::setGlobalTransactionId (const std::vector< unsigned char > & globalTransactionId)` [virtual]
 - 6.606.3.23 `virtual std::string activemq::commands::XATransactionId::toString () const` [virtual]

Returns a string containing the information for this **DataStructure** (p. 1133) such as its type and value of its elements.

Returns

formatted string useful for debugging.

Reimplemented from **activemq::commands::TransactionId** (p. 2770).

6.606.4 Field Documentation

- 6.606.4.1 `std::vector<unsigned char> activemq::commands::XATransactionId::branchQualifier` [protected]

- 6.606.4.2 `int activemq::commands::XATransactionId::formatId` `[protected]`
- 6.606.4.3 `std::vector<unsigned char> activemq::commands::XATransactionId::globalTransactionId` `[protected]`
- 6.606.4.4 `const unsigned char activemq::commands::XATransactionId::ID_XATRANSACTIONID = 112` `[static]`

The documentation for this class was generated from the following file:

- `src/main/activemq/commands/XATransactionId.h`

6.607 `activemq::wireformat::openwire::marshal::generated::XATransactionIdMarshaller` Class Reference

Marshaling code for Open Wire Format for `XATransactionIdMarshaller` (p. 2942).

```
#include <src/main/activemq/wireformat/openwire/marshal/generated/XATransactionIdMarshaller.h>
```

Inheritance diagram for `activemq::wireformat::openwire::marshal::generated::XATransactionIdMarshaller`:

Public Member Functions

- `XATransactionIdMarshaller ()`
- `virtual ~XATransactionIdMarshaller ()`
- `virtual commands::DataStructure * createObject () const`
Creates a new instance of the class that this class is a marshaling director for.
- `virtual unsigned char getDataStructureType () const`
Gets the DataStructureType that this class marshals/unmarshals.
- `virtual void tightUnmarshal (OpenWireFormat *wireFormat, commands::DataStructure *dataStructure, decaf::io::DataInputStream *dataIn, utils::BooleanStream *bs)`
Tight Un-marhsal to the given stream.
- `virtual int tightMarshal1 (OpenWireFormat *wireFormat, commands::DataStructure *dataStructure, utils::BooleanStream *bs)`
Tight Marhsal to the given stream.
- `virtual void tightMarshal2 (OpenWireFormat *wireFormat, commands::DataStructure *dataStructure, decaf::io::DataOutputStream *dataOut, utils::BooleanStream *bs)`
Tight Marhsal to the given stream.
- `virtual void looseUnmarshal (OpenWireFormat *wireFormat, commands::DataStructure *dataStructure, decaf::io::DataInputStream *dataIn)`

6.607 activemq::wireformat::openwire::marshal::generated::XATransactionId-Marshaller Class

Reference

2951

Loose Un-marhsal to the given stream.

- virtual void **looseMarshal** (**OpenWireFormat** *wireFormat, **commands::Data-Structure** *dataStructure, **decaf::io::DataOutputStream** *dataOut)

Tight Marhsal to the given stream.

6.607.1 Detailed Description

Marshaling code for Open Wire Format for **XATransactionIdMarshaller** (p. 2942).

NOTE!: This file is auto generated - do not modify! if you need to make a change, please see the Java Classes in the activemq-openwire-generator module

6.607.2 Constructor & Destructor Documentation

6.607.2.1 **activemq::wireformat::openwire::marshal::generated::XA-TransactionIdMarshaller::XATransactionIdMarshaller** ()
[inline]

6.607.2.2 **virtual activemq::wireformat::openwire::marshal::generated::XA-TransactionIdMarshaller::~~XATransactionIdMarshaller** () [inline, virtual]

6.607.3 Member Function Documentation

6.607.3.1 **virtual commands::DataStructure* activemq::wireformat::openwire-::marshal::generated::XATransactionIdMarshaller::createObject** ()
const [virtual]

Creates a new instance of the class that this class is a marshaling director for.

Returns

newly allocated Command

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1120).

6.607.3.2 **virtual unsigned char activemq::wireformat::openwire::marshal::generated-::XATransactionIdMarshaller::getDataStructureType** () const
[virtual]

Gets the DataStructureType that this class marshals/unmarshals.

Returns

byte Id of this classes DataStructureType

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1122).

6.607.3.3 `virtual void activemq::wireformat::openwire::marshal::generated::XA-TransactionIdMarshaller::looseMarshal (OpenWireFormat * format, commands::DataStructure * command, decaf::io::DataOutputStream * ds) [virtual]`

Tight Marshal to the given stream.

Parameters

<i>format</i>	- The OpenWireFormat properties
<i>command</i>	- the object to Marshal
<i>ds</i>	- DataOutputStream to marshal to

Exceptions

<i>IOException</i>	if an error occurs.
--------------------	---------------------

Reimplemented from `activemq::wireformat::openwire::marshal::generated::TransactionIdMarshaller` (p. 2771).

6.607.3.4 `virtual void activemq::wireformat::openwire::marshal::generated::XA-TransactionIdMarshaller::looseUnmarshal (OpenWireFormat * format, commands::DataStructure * command, decaf::io::DataInputStream * dis) [virtual]`

Loose Un-marshal to the given stream.

Parameters

<i>format</i>	- The OpenWireFormat properties
<i>command</i>	- the object to Un-Marshal
<i>dis</i>	- the DataInputStream to Un-Marshal from

Exceptions

<i>IOException</i>	if an error occurs.
--------------------	---------------------

Reimplemented from `activemq::wireformat::openwire::marshal::generated::TransactionIdMarshaller` (p. 2772).

6.607.3.5 `virtual int activemq::wireformat::openwire::marshal::generated::XA-TransactionIdMarshaller::tightMarshal1 (OpenWireFormat * format, commands::DataStructure * command, utils::BooleanStream * bs) [virtual]`

Tight Marshal to the given stream.

6.607 activemq::wireformat::openwire::marshal::generated::XATransactionId-Marshaller Class

Reference

2953

Parameters

<i>format</i>	The OpenwireFormat properties
<i>command</i>	The object to Marshal
<i>bs</i>	The boolean stream to marshal to.

Exceptions

<i>IOException</i>	if an error occurs.
--------------------	---------------------

Reimplemented from **activemq::wireformat::openwire::marshal::generated::XATransactionIdMarshaller** (p. 2772).

6.607.3.6 virtual void **activemq::wireformat::openwire::marshal::generated::XATransactionIdMarshaller::tightMarshal2** (**OpenWireFormat** * *format*, **commands::DataStructure** * *command*, **decaf::io::DataOutputStream** * *ds*, **utils::BooleanStream** * *bs*) [virtual]

Tight Marhsal to the given stream.

Parameters

<i>format</i>	- The OpenwireFormat properties
<i>command</i>	- the object to Marshal
<i>ds</i>	- the DataOutputStream to Marshal to
<i>bs</i>	- boolean stream to marshal to.

Exceptions

<i>IOException</i>	if an error occurs.
--------------------	---------------------

Reimplemented from **activemq::wireformat::openwire::marshal::generated::XATransactionIdMarshaller** (p. 2773).

6.607.3.7 virtual void **activemq::wireformat::openwire::marshal::generated::XATransactionIdMarshaller::tightUnmarshal** (**OpenWireFormat** * *format*, **commands::DataStructure** * *command*, **decaf::io::DataInputStream** * *dis*, **utils::BooleanStream** * *bs*) [virtual]

Tight Un-marhsal to the given stream.

Parameters

<i>format</i>	- The OpenwireFormat properties
<i>command</i>	- the object to Un-Marshal
<i>dis</i>	- the DataInputStream to Un-Marshal from
<i>bs</i>	- boolean stream to unmarshal from.

Exceptions

<i>IOException</i> if an error occurs.
--

Reimplemented from **activemq::wireformat::openwire::marshal::generated::TransactionIdMarshaller** (p. 2773).

The documentation for this class was generated from the following file:

- `src/main/activemq/wireformat/openwire/marshal/generated/XATransactionIdMarshaller.h`

6.608 cms::Xid Class Reference

An interface which provides a mapping for the X/Open XID transaction identifier structure.

```
#include <src/main/cms/Xid.h>
```

Inheritance diagram for cms::Xid:

Public Member Functions

- **Xid** ()
- virtual **~Xid** ()
- virtual **Xid * clone** () const =0
*Creates a Copy of this **Xid** (p. 2946) instance that contains the same id values.*
- virtual bool **equals** (const **Xid** *other) const =0
*Compares this **Xid** (p. 2946) to another and returns true if they are the same.*
- virtual int **getBranchQualifier** (unsigned char *buffer, int size) const =0
Gets the transaction branch qualifier component of the XID.
- virtual int **getFormatId** () const =0
Gets the format identifier component of the XID.
- virtual int **getGlobalTransactionId** (unsigned char *buffer, int size) const =0
Gets the global transaction id component of the XID.

Static Public Attributes

- static const int **MAXGTRIDSIZE**
The maximum number of bytes which will be copied into the array passed to `getGlobaltransactionId()`.
- static const int **MAXBQUALSIZE**
The maximum number of bytes which will be copied into the array that is passed to `getBranchQualifier()` (p. 2948).

6.608.1 Detailed Description

An interface which provides a mapping for the X/Open XID transaction identifier structure.

The **Xid** (p. 2946) interface is used by the Transaction Manager and the Resource managers. It is not typically used by application programs directly but the application developer must define a mechanism to map the calls and structures used by the Transaction Manager API in use into the format used by the CMS XA interfaces.

Since

2.3

6.608.2 Constructor & Destructor Documentation

6.608.2.1 `cms::Xid::Xid ()`

6.608.2.2 `virtual cms::Xid::~Xid () [virtual]`

6.608.3 Member Function Documentation

6.608.3.1 `virtual Xid* cms::Xid::clone () const [pure virtual]`

Creates a Copy of this **Xid** (p. 2946) instance that contains the same id values.

Returns

a new **Xid** (p. 2946) instance that is equal to this one when compared.

Implemented in **activemq::commands::XATransactionId** (p. 2938).

6.608.3.2 `virtual bool cms::Xid::equals (const Xid * other) const [pure virtual]`

Compares this **Xid** (p. 2946) to another and returns true if they are the same.

Returns

true if both **Xid** (p. 2946)'s represent that same id value.

6.608.3.3 `virtual int cms::Xid::getBranchQualifier (unsigned char * buffer, int size) const [pure virtual]`

Gets the transaction branch qualifier component of the XID.

The value of this **Xid** (p. 2946)'s branch qualifier is copied into the buffer provided. If the size of the provided buffer is not large enough to accommodate the branch qualifier then no copy is performed and the method returns -1.

Parameters

<i>buffer</i>	The location in memory to copy the qualifier bytes to.
<i>size</i>	The size of the buffer provided.

Returns

the number of bytes copied into the buffer, or -1 if the buffer provided was not large enough.

Exceptions

<i>XAException</i> (p. 2921)	if the size parameter is less than zero or buffer is NULL.
--	--

Implemented in **activemq::commands::XATransactionId** (p. 2939).

6.608.3.4 `virtual int cms::Xid::getFormatId () const` [pure virtual]

Gets the format identifier component of the XID.

Returns

an integer containing the format identifier. 0 means the OSI CCR format.

Implemented in **activemq::commands::XATransactionId** (p. 2940).

6.608.3.5 `virtual int cms::Xid::getGlobalTransactionId (unsigned char * buffer, int size) const` [pure virtual]

Gets the global transaction id component of the XID.

The value of this **Xid** (p. 2946)'s transaction id is copied into the buffer provided. If the size of the provided buffer is not large enough to accommodate the transaction id then no copy is performed and the method returns -1.

Parameters

<i>buffer</i>	The location in memory to copy the transaction id bytes to.
<i>size</i>	The size of the buffer provided.

Returns

the number of bytes copied into the buffer, or -1 if the buffer provided was not large enough.

Exceptions

<i>XAException</i> (p. 2921)	if the size parameter is less than zero or buffer is NULL.
--	--

Implemented in **activemq::commands::XATransactionId** (p. 2940).

6.608.4 Field Documentation

6.608.4.1 const int cms::Xid::MAXBQUALSIZE [static]

The maximum number of bytes which will be copied into the array that is passed to **getBranchQualifier()** (p. 2948).

6.608.4.2 const int cms::Xid::MAXGTRIDSIZE [static]

The maximum number of bytes which will be copied into the array passed to **getGlobaltransactionId()**.

The documentation for this class was generated from the following file:

- src/main/cms/Xid.h

6.609 decaf::util::logging::XMLFormatter Class Reference

Format a **LogRecord** (p. 1719) into a standard XML format.

```
#include <src/main/decaf/util/logging/XMLFormatter.h>
```

Inheritance diagram for decaf::util::logging::XMLFormatter:

Public Member Functions

- **XMLFormatter** ()
- virtual **~XMLFormatter** ()
- virtual std::string **format** (const **LogRecord** &record) const
*Converts a **LogRecord** (p. 1719) into an XML string.*
- virtual std::string **getHead** (const **Handler** *handler)
Returns the header string for a set of log records formatted as XML strings, using the output handler's encoding if it is defined, otherwise using the default platform encoding.
- virtual std::string **getTail** (const **Handler** *handler)
Returns the tail string for a set of log records formatted as XML strings.

6.609.1 Detailed Description

Format a **LogRecord** (p. 1719) into a standard XML format.

TODO - Currently only outputs UTF-8 The **XMLFormatter** (p. 2950) can be used with arbitrary character encodings, but it is recommended that it normally be used with UTF-8. The character encoding can be set on the output **Handler** (p. 1401).

Since

1.0

6.609.2 Constructor & Destructor Documentation

6.609.2.1 **decaf::util::logging::XMLFormatter::XMLFormatter ()**

6.609.2.2 **virtual decaf::util::logging::XMLFormatter::~~XMLFormatter ()**
[virtual]

6.609.3 Member Function Documentation

6.609.3.1 **virtual std::string decaf::util::logging::XMLFormatter::format (const LogRecord & record) const** [virtual]

Converts a **LogRecord** (p. 1719) into an XML string.

Parameters

<i>record</i>	The log record to be formatted.
---------------	---------------------------------

Returns

the log record formatted as an XML string.

Implements **decaf::util::logging::Formatter** (p. 1388).

6.609.3.2 **virtual std::string decaf::util::logging::XMLFormatter::getHead (const Handler * handler)** [virtual]

Returns the header string for a set of log records formatted as XML strings, using the output handler's encoding if it is defined, otherwise using the default platform encoding.

Parameters

<i>handler</i>	The output handler, may be NULL.
----------------	----------------------------------

Returns

the header string for log records formatted as XML strings.

6.609.3.3 `virtual std::string decaf::util::logging::XMLFormatter::getTail (const Handler * handler) [virtual]`

Returns the tail string for a set of log records formatted as XML strings.

Parameters

<i>handler</i>	The output handler, may be NULL.
----------------	----------------------------------

Returns

the tail string for log records formatted as XML strings.

The documentation for this class was generated from the following file:

- `src/main/decaf/util/logging/XMLFormatter.h`

6.610 z_stream_s Struct Reference

```
#include <src/main/decaf/internal/util/zip/zlib.h>
```

Data Fields

- **Bytef * next_in**
- **uInt avail_in**
- **uLong total_in**
- **Bytef * next_out**
- **uInt avail_out**
- **uLong total_out**
- **char * msg**
- **struct internal_state FAR * state**
- **alloc_func zalloc**
- **free_func zfree**
- **voidpf opaque**
- **int data_type**
- **uLong Adler**
- **uLong reserved**

6.610.1 Field Documentation

- 6.610.1.1 `uLong z_stream_s::adler`
- 6.610.1.2 `uInt z_stream_s::avail_in`
- 6.610.1.3 `uInt z_stream_s::avail_out`
- 6.610.1.4 `int z_stream_s::data_type`
- 6.610.1.5 `char* z_stream_s::msg`
- 6.610.1.6 `Bytef* z_stream_s::next_in`
- 6.610.1.7 `Bytef* z_stream_s::next_out`
- 6.610.1.8 `voidpf z_stream_s::opaque`
- 6.610.1.9 `uLong z_stream_s::reserved`
- 6.610.1.10 `struct internal_state FAR* z_stream_s::state`
- 6.610.1.11 `uLong z_stream_s::total_in`
- 6.610.1.12 `uLong z_stream_s::total_out`
- 6.610.1.13 `alloc_func z_stream_s::zalloc`
- 6.610.1.14 `free_func z_stream_s::zfree`

The documentation for this struct was generated from the following file:

- `src/main/decaf/internal/util/zip/zlib.h`

6.611 decaf::util::zip::ZipException Class Reference

```
#include <src/main/decaf/util/zip/ZipException.h>
```

Inheritance diagram for `decaf::util::zip::ZipException`:

Public Member Functions

- **`ZipException () throw ()`**
Default Constructor.

- **ZipException** (const lang::Exception &ex) throw ()
Copy Constructor.
- **ZipException** (const ZipException &ex) throw ()
Copy Constructor.
- **ZipException** (const char *file, const int lineNumber, const std::exception *cause, const char *msg,...) throw ()
Constructor - Initializes the file name and line number where this message occurred.
- **ZipException** (const std::exception *cause) throw ()
Constructor.
- **ZipException** (const char *file, const int lineNumber, const char *msg,...) throw ()
Constructor.
- virtual **ZipException** * clone () const
Clones this exception.
- virtual ~**ZipException** () throw ()

6.611.1 Constructor & Destructor Documentation

6.611.1.1 decaf::util::zip::ZipException::ZipException () throw ()

Default Constructor.

6.611.1.2 decaf::util::zip::ZipException::ZipException (const lang::Exception & ex) throw () [inline]

Copy Constructor.

Parameters

ex	the exception to copy
-----------	-----------------------

6.611.1.3 decaf::util::zip::ZipException::ZipException (const ZipException & ex) throw () [inline]

Copy Constructor.

Parameters

ex	the exception to copy, which is an instance of this type
-----------	--

6.611.1.4 **decaf::util::zip::ZipException::ZipException** (*const char * file*, *const int lineNumber*, *const std::exception * cause*, *const char * msg*, ...) *throw ()*
 [inline]

Constructor - Initializes the file name and line number where this message occurred.

Sets the message to report, using an optional list of arguments to parse into the message

Parameters

<i>file</i>	The file name where exception occurs
<i>lineNumber</i>	The line number where the exception occurred.
<i>cause</i>	The exception that was the cause for this one to be thrown.
<i>msg</i>	The message to report
...	list of primitives that are formatted into the message

6.611.1.5 **decaf::util::zip::ZipException::ZipException** (*const std::exception * cause*)
 throw () [inline]

Constructor.

Parameters

<i>cause</i>	Pointer to the exception that caused this one to be thrown, the object is cloned caller retains ownership.
--------------	--

6.611.1.6 **decaf::util::zip::ZipException::ZipException** (*const char * file*, *const int lineNumber*, *const char * msg*, ...) *throw ()* [inline]

Constructor.

Parameters

<i>file</i>	The file name where exception occurs
<i>lineNumber</i>	The line number where the exception occurred.
<i>msg</i>	The message to report
...	list of primitives that are formatted into the message

6.611.1.7 **virtual decaf::util::zip::ZipException::~ZipException** () *throw ()*
 [virtual]

6.611.2 Member Function Documentation

6.611.2.1 `virtual ZipException* decaf::util::zip::ZipException::clone () const`
`[inline, virtual]`

Clones this exception.

This is useful for cases where you need to preserve the type of the original exception as well as the message. All subclasses should override.

Returns

a new instance of an Exception that is a copy of this one.

Reimplemented from **decaf::io::IOException** (p. 1547).

The documentation for this class was generated from the following file:

- `src/main/decaf/util/zip/ZipException.h`

Chapter 7

File Documentation

7.1 src/main/activemq/cmsutil/CachedConsumer.h File Reference

```
#include <cms/MessageConsumer.h> #include <activemq/util/-  
Config.h>
```

Data Structures

- class **activemq::cmsutil::CachedConsumer**
A cached message consumer contained within a pooled session.

Namespaces

- namespace **activemq**
Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.
- namespace **activemq::cmsutil**

7.2 src/main/activemq/cmsutil/CachedProducer.h File Reference

```
#include <cms/MessageProducer.h> #include <activemq/util/-  
Config.h>
```

Data Structures

- class **activemq::cmsutil::CachedProducer**
A cached message producer contained within a pooled session.

Namespaces

- namespace **activemq**

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

- namespace **activemq::cmsutil**

7.3 src/main/activemq/cmsutil/CmsAccessor.h File Reference

```
#include <cms/ConnectionFactory.h> #include <activemq/cmsutil/-  
ResourceLifecycleManager.h>      #include <activemq/util/-  
Config.h>
```

Data Structures

- class **activemq::cmsutil::CmsAccessor**

*Base class for **activemq.cmsutil.CmsTemplate** (p. 836) and other CMS-accessing gateway helpers, defining common properties such as the CMS **cms.ConnectionFactory** (p. 955) to operate on.*

Namespaces

- namespace **activemq**

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

- namespace **activemq::cmsutil**

7.4 src/main/activemq/cmsutil/CmsDestinationAccessor.h File - Reference

```
#include <activemq/util/Config.h> #include <activemq/cmsutil/-  
CmsAccessor.h> #include <activemq/cmsutil/DynamicDestination-  
Resolver.h>
```

Data Structures

- class **activemq::cmsutil::CmsDestinationAccessor**

*Extends the **CmsAccessor** (p. 819) to add support for resolving destination names.*

Namespaces

- namespace **activemq**
Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.
- namespace **activemq::cmsutil**

7.5 src/main/activemq/cmsutil/CmsTemplate.h File Reference

```
#include <activemq/util/Config.h> #include <activemq/cmsutil/-
CmsDestinationAccessor.h> #include <activemq/cmsutil/-
SessionCallback.h> #include <activemq/cmsutil/Producer-
Callback.h> #include <activemq/cmsutil/SessionPool.h>
#include <cms/ConnectionFactory.h> #include <cms/Delivery-
Mode.h> #include <string>
```

Data Structures

- class **activemq::cmsutil::CmsTemplate**
***CmsTemplate** (p. 836) simplifies performing synchronous CMS operations.*
- class **activemq::cmsutil::CmsTemplate::ProducerExecutor**
- class **activemq::cmsutil::CmsTemplate::ResolveProducerExecutor**
- class **activemq::cmsutil::CmsTemplate::SendExecutor**
- class **activemq::cmsutil::CmsTemplate::ReceiveExecutor**
- class **activemq::cmsutil::CmsTemplate::ResolveReceiveExecutor**

Namespaces

- namespace **activemq**
Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.
- namespace **activemq::cmsutil**

7.6 src/main/activemq/cmsutil/DestinationResolver.h File Reference

```
#include <cms/Session.h> #include <activemq/util/Config.-
h>
```

Data Structures

- class **activemq::cmsutil::DestinationResolver**
*Resolves a CMS destination name to a *Destination*.*

Namespaces

- namespace **activemq**

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

- namespace **activemq::cmsutil**

7.7 src/main/activemq/cmsutil/DynamicDestinationResolver.h File Reference

```
#include <activemq/cmsutil/DestinationResolver.h> #include  
<cms/Session.h> #include <decaf/util/StlMap.h> #include  
<activemq/util/Config.h>
```

Data Structures

- class **activemq::cmsutil::DynamicDestinationResolver**

Resolves a CMS destination name to a `Destination`.

- class **activemq::cmsutil::DynamicDestinationResolver::SessionResolver**

Manages maps of names to topics and queues for a single session.

Namespaces

- namespace **activemq**

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

- namespace **activemq::cmsutil**

7.8 src/main/activemq/cmsutil/MessageCreator.h File Reference

```
#include <cms/Session.h> #include <cms/Message.h> #include  
<activemq/util/Config.h>
```

Data Structures

- class **activemq::cmsutil::MessageCreator**

Creates the user-defined message to be sent by the `CmsTemplate` (p. 836).

Namespaces

- namespace **activemq**
Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.
- namespace **activemq::cmsutil**

7.9 src/main/activemq/cmsutil/PooledSession.h File Reference

```
#include <cms/Session.h>    #include <decaf/util/StlMap.-  
h> #include <activemq/cmsutil/CachedProducer.h> #include  
<activemq/cmsutil/CachedConsumer.h> #include <activemq/util/-  
Config.h>
```

Data Structures

- class **activemq::cmsutil::PooledSession**
A pooled session object that wraps around a delegate session.

Namespaces

- namespace **activemq**
Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.
- namespace **activemq::cmsutil**

7.10 src/main/activemq/cmsutil/ProducerCallback.h File Reference

```
#include <cms/Session.h> #include <cms/MessageProducer.-  
h> #include <activemq/util/Config.h>
```

Data Structures

- class **activemq::cmsutil::ProducerCallback**
Callback for sending a message to a CMS destination.

Namespaces

- namespace **activemq**
Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.
- namespace **activemq::cmsutil**

7.11 src/main/activemq/cmsutil/ResourceLifecycleManager.h File - Reference

```
#include <cms/Connection.h>      #include <cms/Session.h>
#include <cms/Destination.h> #include <cms/MessageProducer.h>
#include <cms/MessageConsumer.h> #include <activemq/util/Config.h>
#include <decaf/util/LinkedList.h>
```

Data Structures

- class **activemq::cmsutil::ResourceLifecycleManager**
Manages the lifecycle of a set of CMS resources.

Namespaces

- namespace **activemq**
Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.
- namespace **activemq::cmsutil**

7.12 src/main/decaf/internal/util/ResourceLifecycleManager.h File - Reference

```
#include <decaf/util/Config.h> #include <decaf/util/StlSet.h>
#include <decaf/internal/util/Resource.h>
```

Data Structures

- class **decaf::internal::util::ResourceLifecycleManager**

Namespaces

- namespace **decaf**
Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.
- namespace **decaf::internal**
- namespace **decaf::internal::util**

7.13 src/main/activemq/cmsutil/SessionCallback.h File Reference

```
#include <cms/Session.h> #include <activemq/util/Config.h>
```


Data Structures

- class **activemq::cmsutil::SessionCallback**

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

Namespaces

- namespace **activemq**

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

- namespace **activemq::cmsutil**

7.14 src/main/activemq/cmsutil/SessionPool.h File Reference

```
#include <activemq/cmsutil/PooledSession.h> #include <decaf/util/concurrent/-  
Mutex.h> #include <cms/Connection.h> #include <list> x  
#include <activemq/util/Config.h>
```

Data Structures

- class **activemq::cmsutil::SessionPool**

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

Namespaces

- namespace **activemq**

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

- namespace **activemq::cmsutil**

7.15 src/main/activemq/commands/ActiveMQBlobMessage.h File - Reference

```
#include <activemq/util/Config.h> #include <activemq/commands/-  
ActiveMQMessageTemplate.h> #include <cms/Message.h> x  
#include <string> #include <memory>
```

Data Structures

- class **activemq::commands::ActiveMQBlobMessage**

Namespaces

- namespace **activemq**

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

- namespace **activemq::commands**

7.16 src/main/activemq/commands/ActiveMQBytesMessage.h File Reference

```
#include <activemq/util/Config.h> #include <activemq/commands/-  
ActiveMQMessageTemplate.h> #include <decaf/io/ByteArray-  
OutputStream.h> #include <decaf/io/DataInputStream.h> ×  
#include <decaf/io/DataOutputStream.h> #include <cms/-  
BytesMessage.h> #include <vector> #include <string> ×  
#include <memory>
```

Data Structures

- class **activemq::commands::ActiveMQBytesMessage**

Namespaces

- namespace **activemq**

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

- namespace **activemq::commands**

7.17 src/main/activemq/commands/ActiveMQDestination.h File - Reference

```
#include <activemq/util/Config.h> #include <activemq/commands/-  
BaseDataStructure.h> #include <activemq/util/ActiveM-  
QProperties.h> #include <cms/Destination.h> #include  
<decaf/lang/Pointer.h> #include <vector> #include <string> ×  
#include <map>
```

Data Structures

- class **activemq::commands::ActiveMQDestination**
- struct **activemq::commands::ActiveMQDestination::DestinationFilter**

7.18 src/main/activemq/commands/ActiveMQMapMessage.h File Reference 2973

Namespaces

- namespace **activemq**
Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.
- namespace **activemq::commands**

7.18 src/main/activemq/commands/ActiveMQMapMessage.h File - Reference

```
#include <activemq/util/Config.h> #include <activemq/commands/-  
ActiveMQMessageTemplate.h> #include <activemq/util/Primitive-  
Map.h> #include <decaf/lang/exceptions/NullPointerException.-  
h> #include <cms/MapMessage.h> #include <vector> #include  
<string> #include <memory>
```

Data Structures

- class **activemq::commands::ActiveMQMapMessage**

Namespaces

- namespace **activemq**
Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.
- namespace **activemq::commands**

7.19 src/main/activemq/commands/ActiveMQMessage.h File - Reference

```
#include <cms/Message.h> #include <activemq/util/Config.-  
h> #include <activemq/commands/ActiveMQMessageTemplate.-  
h>
```

Data Structures

- class **activemq::commands::ActiveMQMessage**

Namespaces

- namespace **activemq**

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

- namespace **activemq::commands**

7.20 src/main/activemq/commands/ActiveMQMessageTemplate.h - File Reference

```
#include <cms/DeliveryMode.h>    #include <activemq/util/-
Config.h> #include <activemq/commands/Message.h> #include
<activemq/core/ActiveMQAckHandler.h> #include <activemq/core/-
ActiveMQConnection.h> #include <activemq/wireformat/openwire/utis/-
MessagePropertyInterceptor.h> #include <activemq/wireformat/openwire/marsha
BaseDataStreamMarshaller.h>    #include <activemq/util/C-
MSExceptionSupport.h>    #include <decaf/lang/exceptions/-
UnsupportedOperationException.h>    #include <cms/Illegal-
StateException.h>    #include <cms/MessageFormatException.-
h> #include <cms/MessageNotReadableException.h> #include
<cms/MessageNotWriteableException.h>
```

Data Structures

- class **activemq::commands::ActiveMQMessageTemplate**< T >

Namespaces

- namespace **activemq**

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

- namespace **activemq::commands**

7.21 src/main/activemq/commands/ActiveMQObjectMessage.h - File Reference

```
#include <activemq/commands/ActiveMQMessageTemplate.h> ×
#include <cms/ObjectMessage.h>    #include <activemq/util/-
Config.h> #include <memory>
```

Data Structures

- class **activemq::commands::ActiveMQObjectMessage**

Namespaces

- namespace **activemq**
Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.
- namespace **activemq::commands**

7.22 src/main/activemq/commands/ActiveMQQueue.h File Reference

```
#include <activemq/util/Config.h> #include <activemq/commands/-
ActiveMQDestination.h> #include <activemq/exceptions/-
ActiveMQException.h> #include <decaf/lang/Exception.-
h> #include <cms/Queue.h> #include <vector> #include
<string> #include <memory>
```

Data Structures

- class **activemq::commands::ActiveMQQueue**

Namespaces

- namespace **activemq**
Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.
- namespace **activemq::commands**

7.23 src/main/activemq/commands/ActiveMQStreamMessage.h - File Reference

```
#include <activemq/util/Config.h> #include <activemq/commands/-
ActiveMQMessage.h> #include <activemq/commands/Active-
MQMessageTemplate.h> #include <cms/StreamMessage.h> ×
#include <cms/MessageNotWriteableException.h> #include
<cms/MessageNotReadableException.h> #include <cms/Message-
FormatException.h> #include <cms/MessageEOFException.h> ×
#include <decaf/lang/exceptions/NullPointerException.h>
#include <decaf/io/DataInputStream.h> #include <decaf/io/-
DataOutputStream.h> #include <decaf/io/ByteArrayOutput-
Stream.h> #include <string> #include <memory>
```

Data Structures

- class **activemq::commands::ActiveMQStreamMessage**

Namespaces

- namespace **activemq**

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

- namespace **activemq::commands**

7.24 src/main/activemq/commands/ActiveMQTempDestination.h File Reference

```
#include <activemq/util/Config.h> #include <activemq/commands/-  
ActiveMQDestination.h> #include <activemq/exceptions/-  
ActiveMQException.h> #include <cms/Closeable.h> #include  
<vector> #include <string>
```

Data Structures

- class **activemq::commands::ActiveMQTempDestination**

Namespaces

- namespace **activemq**

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

- namespace **activemq::core**
- namespace **activemq::commands**

7.25 src/main/activemq/commands/ActiveMQTempQueue.h File - Reference

```
#include <activemq/util/Config.h> #include <activemq/commands/-  
ActiveMQTempDestination.h> #include <cms/TemporaryQueue.-  
h> #include <vector> #include <string> #include <memory> x
```

Data Structures

- class **activemq::commands::ActiveMQTempQueue**

Namespaces

- namespace **activemq**
Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.
- namespace **activemq::commands**

7.26 src/main/activemq/commands/ActiveMQTempTopic.h File - Reference

```
#include <activemq/util/Config.h> #include <activemq/commands/-  
ActiveMQTempDestination.h> #include <cms/TemporaryTopic.-  
h> #include <vector> #include <string> #include <memory> ×
```

Data Structures

- class **activemq::commands::ActiveMQTempTopic**

Namespaces

- namespace **activemq**
Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.
- namespace **activemq::commands**

7.27 src/main/activemq/commands/ActiveMQTextMessage.h File - Reference

```
#include <activemq/util/Config.h> #include <activemq/commands/-  
ActiveMQMessageTemplate.h> #include <cms/TextMessage.h>  
#include <vector> #include <string> #include <memory>
```

Data Structures

- class **activemq::commands::ActiveMQTextMessage**

Namespaces

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

- namespace **activemq::commands**

7.28 src/main/activemq/commands/ActiveMQTopic.h File Reference

```
#include <activemq/util/Config.h> #include <activemq/commands/-  
ActiveMQDestination.h> #include <activemq/exceptions/-  
ActiveMQException.h> #include <decaf/lang/Exception.-  
h> #include <cms/Topic.h> #include <vector> #include  
<string> #include <memory>
```

Data Structures

- class **activemq::commands::ActiveMQTopic**

Namespaces

- namespace **activemq**

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

- namespace **activemq::commands**

7.29 src/main/activemq/commands/BaseCommand.h File Reference

```
#include <activemq/util/Config.h> #include <activemq/commands/-  
Command.h>
```

Data Structures

- class **activemq::commands::BaseCommand**

Namespaces

- namespace **activemq**

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

- namespace **activemq::commands**

7.30 src/main/activemq/commands/BaseDataStructure.h File Reference

```
#include <activemq/util/Config.h> #include <activemq/commands/-
DataStructure.h> #include <string> #include <sstream>
```

Data Structures

- class **activemq::commands::BaseDataStructure**

Namespaces

- namespace **activemq**
Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.
- namespace **activemq::wireformat**
- namespace **activemq::commands**

7.31 src/main/activemq/commands/BooleanExpression.h File Reference

```
#include <activemq/commands/BaseDataStructure.h> #include
<activemq/util/Config.h>
```

Data Structures

- class **activemq::commands::BooleanExpression**

Namespaces

- namespace **activemq**
Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.
- namespace **activemq::commands**

7.32 src/main/activemq/commands/BrokerError.h File Reference

```
#include <activemq/util/Config.h> #include <activemq/commands/-
BaseCommand.h> #include <decaf/lang/exceptions/NullPointerException-
Exception.h> #include <decaf/lang/Pointer.h> #include
<string> #include <vector>
```

Data Structures

- class **activemq::commands::BrokerError**
This class represents an Exception sent from the Broker.
- struct **activemq::commands::BrokerError::StackTraceElement**

Namespaces

- namespace **activemq**
Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.
- namespace **activemq::commands**

7.33 src/main/activemq/commands/BrokerId.h File Reference

```
#include <activemq/commands/BaseDataStructure.h> #include  
<activemq/util/Config.h> #include <decaf/lang/Comparable.h>  
#include <decaf/lang/Pointer.h> #include <string> ×  
#include <vector>
```

Data Structures

- class **activemq::commands::BrokerId**

Namespaces

- namespace **activemq**
Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.
- namespace **activemq::commands**

7.34 src/main/activemq/commands/BrokerInfo.h File Reference

```
#include <activemq/commands/BaseCommand.h> #include <activemq/commands/-  
BrokerId.h> #include <activemq/commands/BrokerInfo.h> ×  
#include <activemq/util/Config.h> #include <decaf/lang/-  
Pointer.h> #include <string> #include <vector>
```

Data Structures

- class **activemq::commands::BrokerInfo**

Namespaces

- namespace **activemq**
Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.
- namespace **activemq::commands**

7.35 src/main/activemq/commands/Command.h File Reference

```
#include <string>          #include <activemq/util/Config.h> ×  
#include <activemq/commands/BaseDataStructure.h> #include  
<activemq/exceptions/ActiveMQException.h> #include <decaf/lang/-  
Pointer.h>
```

Data Structures

- class **activemq::commands::Command**

Namespaces

- namespace **activemq**
Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.
- namespace **activemq::state**
- namespace **activemq::commands**

7.36 src/main/activemq/commands/ConnectionControl.h File - Reference

```
#include <activemq/commands/BaseCommand.h> #include <activemq/util/-  
Config.h> #include <decaf/lang/Pointer.h> #include <string> ×  
#include <vector>
```

Data Structures

- class **activemq::commands::ConnectionControl**

Namespaces

- namespace **activemq**
Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.
- namespace **activemq::commands**

7.37 src/main/activemq/commands/ConnectionError.h File Reference

```
#include <activemq/commands/BaseCommand.h> #include <activemq/commands/-
BrokerError.h> #include <activemq/commands/ConnectionId.-
h> #include <activemq/util/Config.h> #include <decaf/lang/-
Pointer.h> #include <string> #include <vector>
```

Data Structures

- class **activemq::commands::ConnectionError**

Namespaces

- namespace **activemq**
Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.
- namespace **activemq::commands**

7.38 src/main/activemq/commands/ConnectionId.h File Reference

```
#include <activemq/commands/BaseDataStructure.h> #include
<activemq/util/Config.h> #include <decaf/lang/Comparable.-
h> #include <decaf/lang/Pointer.h> #include <string> x
#include <vector>
```

Data Structures

- class **activemq::commands::ConnectionId**

Namespaces

- namespace **activemq**
Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.
- namespace **activemq::commands**

7.39 src/main/activemq/commands/ConnectionInfo.h File Reference

```
#include <activemq/commands/BaseCommand.h> #include <activemq/commands/-
BrokerId.h> #include <activemq/commands/ConnectionId.-
h> #include <activemq/commands/RemoveInfo.h> #include
```

```
<activemq/util/Config.h>    #include <decaf/lang/Pointer.h>
#include <string> #include <vector>
```

Data Structures

- class **activemq::commands::ConnectionInfo**

Namespaces

- namespace **activemq**
Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.
- namespace **activemq::commands**

7.40 src/main/activemq/commands/ConsumerControl.h File Reference

```
#include <activemq/commands/ActiveMQDestination.h> #include
<activemq/commands/BaseCommand.h> #include <activemq/commands/-
ConsumerId.h> #include <activemq/util/Config.h> #include
<decaf/lang/Pointer.h> #include <string> #include <vector> x
```

Data Structures

- class **activemq::commands::ConsumerControl**

Namespaces

- namespace **activemq**
Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.
- namespace **activemq::commands**

7.41 src/main/activemq/commands/ConsumerId.h File Reference

```
#include <activemq/commands/BaseDataStructure.h> #include
<activemq/commands/SessionId.h> #include <activemq/util/-
Config.h> #include <decaf/lang/Comparable.h> #include
<decaf/lang/Pointer.h> #include <string> #include <vector> x
```

Data Structures

- class **activemq::commands::ConsumerId**

Namespaces

- namespace **activemq**
Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.
- namespace **activemq::commands**

7.42 src/main/activemq/commands/ConsumerInfo.h File Reference

```
#include <activemq/commands/ActiveMQDestination.h> #include
<activemq/commands/BaseCommand.h> #include <activemq/commands/-
BooleanExpression.h> #include <activemq/commands/Broker-
Id.h> #include <activemq/commands/ConsumerId.h> #include
<activemq/commands/RemoveInfo.h> #include <activemq/util/-
Config.h> #include <decaf/lang/Pointer.h> #include <string> x
#include <vector>
```

Data Structures

- class **activemq::commands::ConsumerInfo**

Namespaces

- namespace **activemq**
Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.
- namespace **activemq::commands**

7.43 src/main/activemq/commands/ControlCommand.h File Reference

```
#include <activemq/commands/BaseCommand.h> #include <activemq/util/-
Config.h> #include <decaf/lang/Pointer.h> #include <string> x
#include <vector>
```

Data Structures

- class **activemq::commands::ControlCommand**

7.44 src/main/activemq/commands/DataArrayResponse.h File Reference 2985

Namespaces

- namespace **activemq**
Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.
- namespace **activemq::commands**

7.44 src/main/activemq/commands/DataArrayResponse.h File - Reference

```
#include <activemq/commands/DataStructure.h>      #include  
<activemq/commands/Response.h> #include <activemq/util/-  
Config.h> #include <decaf/lang/Pointer.h> #include <string> x  
#include <vector>
```

Data Structures

- class **activemq::commands::DataArrayResponse**

Namespaces

- namespace **activemq**
Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.
- namespace **activemq::commands**

7.45 src/main/activemq/commands/DataResponse.h File Reference

```
#include <activemq/commands/DataStructure.h>      #include  
<activemq/commands/Response.h> #include <activemq/util/-  
Config.h> #include <decaf/lang/Pointer.h> #include <string> x  
#include <vector>
```

Data Structures

- class **activemq::commands::DataResponse**

Namespaces

- namespace **activemq**
Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.
- namespace **activemq::commands**

7.46 src/main/activemq/commands/DataSeture.h File Reference

```
#include <activemq/util/Config.h> #include <activemq/wireformat/-  
MarshalAware.h>
```

Data Structures

- class **activemq::commands::DataSeture**

Namespaces

- namespace **activemq**
Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.
- namespace **activemq::commands**

7.47 src/main/activemq/commands/DestinationInfo.h File Reference

```
#include <activemq/commands/ActiveMQDestination.h> #include  
<activemq/commands/BaseCommand.h> #include <activemq/commands/-  
BrokerId.h> #include <activemq/commands/ConnectionId.h>  
#include <activemq/util/Config.h> #include <decaf/lang/-  
Pointer.h> #include <string> #include <vector>
```

Data Structures

- class **activemq::commands::DestinationInfo**

Namespaces

- namespace **activemq**
Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.
- namespace **activemq::commands**

7.48 src/main/activemq/commands/DiscoveryEvent.h File Reference

```
#include <activemq/commands/BaseDataSeture.h> #include  
<activemq/util/Config.h> #include <decaf/lang/Pointer.-  
h> #include <string> #include <vector>
```


Data Structures

- class **activemq::commands::DiscoveryEvent**

Namespaces

- namespace **activemq**
Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.
- namespace **activemq::commands**

7.49 src/main/activemq/commands/ExceptionResponse.h File - Reference

```
#include <activemq/commands/BrokerError.h> #include <activemq/commands/-  
Response.h> #include <activemq/util/Config.h> #include  
<decaf/lang/Pointer.h> #include <string> #include <vector> x
```

Data Structures

- class **activemq::commands::ExceptionResponse**

Namespaces

- namespace **activemq**
Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.
- namespace **activemq::commands**

7.50 src/main/activemq/commands/FlushCommand.h File Reference

```
#include <activemq/commands/BaseCommand.h> #include <activemq/util/-  
Config.h> #include <decaf/lang/Pointer.h> #include <string> x  
#include <vector>
```

Data Structures

- class **activemq::commands::FlushCommand**

Namespaces

- namespace **activemq**
Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.
- namespace **activemq::commands**

7.51 src/main/activemq/commands/IntegerResponse.h File - Reference

```
#include <activemq/commands/Response.h> #include <activemq/util/Config.h> #include <decaf/lang/Pointer.h> #include <string> ×  
#include <vector>
```

Data Structures

- class **activemq::commands::IntegerResponse**

Namespaces

- namespace **activemq**
Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.
- namespace **activemq::commands**

7.52 src/main/activemq/commands/JournalQueueAck.h File - Reference

```
#include <activemq/commands/ActiveMQDestination.h> #include  
<activemq/commands/BaseDataStructure.h> #include <activemq/commands/MessageAck.h> #include <activemq/util/Config.h> #include  
<decaf/lang/Pointer.h> #include <string> #include <vector> ×
```

Data Structures

- class **activemq::commands::JournalQueueAck**

Namespaces

- namespace **activemq**

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

- namespace **activemq::commands**

7.53 src/main/activemq/commands/JournalTopicAck.h File Reference

```
#include <activemq/commands/ActiveMQDestination.h> #include
<activemq/commands/BaseDataStructure.h> #include <activemq/commands/-
MessageId.h> #include <activemq/commands/TransactionId.-
h> #include <activemq/util/Config.h> #include <decaf/lang/-
Pointer.h> #include <string> #include <vector>
```

Data Structures

- class **activemq::commands::JournalTopicAck**

Namespaces

- namespace **activemq**

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

- namespace **activemq::commands**

7.54 src/main/activemq/commands/JournalTrace.h File Reference

```
#include <activemq/commands/BaseDataStructure.h> #include
<activemq/util/Config.h> #include <decaf/lang/Pointer.-
h> #include <string> #include <vector>
```

Data Structures

- class **activemq::commands::JournalTrace**

Namespaces

- namespace **activemq**

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

- namespace **activemq::commands**

7.55 src/main/activemq/commands/JournalTransaction.h File - Reference

```
#include <activemq/commands/BaseDataStructure.h> #include  
<activemq/commands/TransactionId.h> #include <activemq/util/-  
Config.h> #include <decaf/lang/Pointer.h> #include <string> ×  
#include <vector>
```

Data Structures

- class **activemq::commands::JournalTransaction**

Namespaces

- namespace **activemq**
Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.
- namespace **activemq::commands**

7.56 src/main/activemq/commands/KeepAliveInfo.h File Reference

```
#include <activemq/commands/BaseCommand.h> #include <activemq/util/-  
Config.h> #include <decaf/lang/Pointer.h> #include <string> ×  
#include <vector>
```

Data Structures

- class **activemq::commands::KeepAliveInfo**

Namespaces

- namespace **activemq**
Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.
- namespace **activemq::commands**

7.57 src/main/activemq/commands/LastPartialCommand.h File - Reference

```
#include <activemq/commands/PartialCommand.h> #include  
<activemq/util/Config.h> #include <decaf/lang/Pointer.-  
h> #include <string> #include <vector>
```

Data Structures

- class **activemq::commands::LastPartialCommand**

Namespaces

- namespace **activemq**
Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.
- namespace **activemq::commands**

7.58 src/main/activemq/commands/LocalTransactionId.h File Reference

```
#include <activemq/commands/ConnectionId.h> #include <activemq/commands/-
TransactionId.h> #include <activemq/util/Config.h> #include
<decaf/lang/Comparable.h> #include <decaf/lang/Pointer.-
h> #include <string> #include <vector>
```

Data Structures

- class **activemq::commands::LocalTransactionId**

Namespaces

- namespace **activemq**
Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.
- namespace **activemq::commands**

7.59 src/main/activemq/commands/Message.h File Reference

```
#include <activemq/commands/ActiveMQDestination.h> #include
<activemq/commands/BaseCommand.h> #include <activemq/commands/-
BrokerId.h> #include <activemq/commands/ConsumerId.h> ×
#include <activemq/commands/DataStructure.h> #include
<activemq/commands/MessageId.h> #include <activemq/commands/-
ProducerId.h> #include <activemq/commands/TransactionId.-
h> #include <activemq/core/ActiveMQAckHandler.h> #include
<activemq/util/Config.h> #include <activemq/util/Primitive-
Map.h> #include <decaf/lang/Pointer.h> #include <string> ×
#include <vector>
```

Data Structures

- class **activemq::commands::Message**

Namespaces

- namespace **activemq**
Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.
- namespace **activemq::core**
- namespace **activemq::commands**

7.60 src/main/cms/Message.h File Reference

```
#include <cms/Config.h>          #include <cms/Destination.h>
#include <cms/DeliveryMode.h> #include <cms/CMSException.-
h> #include <cms/IllegalStateException.h> #include <cms/-
MessageFormatException.h> #include <cms/MessageNotWriteable-
Exception.h>
```

Data Structures

- class **cms::Message**
Root of all messages.

Namespaces

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

7.61 src/main/activemq/commands/MessageAck.h File Reference

```
#include <activemq/commands/ActiveMQDestination.h> #include
<activemq/commands/BaseCommand.h> #include <activemq/commands/-
ConsumerId.h>          #include <activemq/commands/MessageId.-
h> #include <activemq/commands/TransactionId.h> #include
<activemq/util/Config.h>  #include <decaf/lang/Pointer.-
h> #include <string> #include <vector>
```

Data Structures

- class **activemq::commands::MessageAck**

Namespaces

- namespace **activemq**

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

- namespace **activemq::commands**

7.62 src/main/activemq/commands/MessageDispatch.h File Reference

```
#include <activemq/commands/ActiveMQDestination.h> #include
<activemq/commands/BaseCommand.h> #include <activemq/commands/-
ConsumerId.h> #include <activemq/commands/Message.h> ×
#include <activemq/util/Config.h> #include <decaf/lang/-
Pointer.h> #include <string> #include <vector>
```

Data Structures

- class **activemq::commands::MessageDispatch**

Namespaces

- namespace **activemq**

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

- namespace **activemq::commands**

7.63 src/main/activemq/commands/MessageDispatchNotification.h File Reference

```
#include <activemq/commands/ActiveMQDestination.h> #include
<activemq/commands/BaseCommand.h> #include <activemq/commands/-
ConsumerId.h> #include <activemq/commands/MessageId.h> ×
#include <activemq/util/Config.h> #include <decaf/lang/-
Pointer.h> #include <string> #include <vector>
```

Data Structures

- class **activemq::commands::MessageDispatchNotification**

Namespaces

- namespace **activemq**
Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.
- namespace **activemq::commands**

7.64 src/main/activemq/commands/MessageId.h File Reference

```
#include <activemq/commands/BaseDataStructure.h> #include  
<activemq/commands/ProducerId.h> #include <activemq/commands/-  
ProducerInfo.h> #include <activemq/util/Config.h> #include  
<decaf/lang/Comparable.h> #include <decaf/lang/Pointer.-  
h> #include <string> #include <vector>
```

Data Structures

- class **activemq::commands::MessageId**

Namespaces

- namespace **activemq**
Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.
- namespace **activemq::commands**

7.65 src/main/activemq/commands/MessagePull.h File Reference

```
#include <activemq/commands/ActiveMQDestination.h> #include  
<activemq/commands/BaseCommand.h> #include <activemq/commands/-  
ConsumerId.h> #include <activemq/commands/MessageId.h> x  
#include <activemq/util/Config.h> #include <decaf/lang/-  
Pointer.h> #include <string> #include <vector>
```

Data Structures

- class **activemq::commands::MessagePull**

Namespaces

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

- namespace **activemq::commands**

7.66 src/main/activemq/commands/NetworkBridgeFilter.h File - Reference

```
#include <activemq/commands/BaseDataStructure.h> #include
<activemq/commands/BrokerId.h> #include <activemq/util/-
Config.h> #include <decaf/lang/Pointer.h> #include <string> x
#include <vector>
```

Data Structures

- class **activemq::commands::NetworkBridgeFilter**

Namespaces

- namespace **activemq**
Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.
- namespace **activemq::commands**

7.67 src/main/activemq/commands/PartialCommand.h File - Reference

```
#include <activemq/commands/BaseDataStructure.h> #include
<activemq/util/Config.h> #include <decaf/lang/Pointer.-
h> #include <string> #include <vector>
```

Data Structures

- class **activemq::commands::PartialCommand**

Namespaces

- namespace **activemq**
Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.
- namespace **activemq::commands**

7.68 src/main/activemq/commands/ProducerAck.h File Reference

```
#include <activemq/commands/BaseCommand.h> #include <activemq/commands/-  
ProducerId.h> #include <activemq/util/Config.h> #include  
<decaf/lang/Pointer.h> #include <string> #include <vector> x
```

Data Structures

- class **activemq::commands::ProducerAck**

Namespaces

- namespace **activemq**
*Licensed to the Apache Software Foundation (ASF) under one or more contributor
license agreements.*
- namespace **activemq::commands**

7.69 src/main/activemq/commands/ProducerId.h File Reference

```
#include <activemq/commands/BaseDataStructure.h> #include  
<activemq/commands/SessionId.h> #include <activemq/util/-  
Config.h> #include <decaf/lang/Comparable.h> #include  
<decaf/lang/Pointer.h> #include <string> #include <vector> x
```

Data Structures

- class **activemq::commands::ProducerId**

Namespaces

- namespace **activemq**
*Licensed to the Apache Software Foundation (ASF) under one or more contributor
license agreements.*
- namespace **activemq::commands**

7.70 src/main/activemq/commands/ProducerInfo.h File Reference

```
#include <activemq/commands/ActiveMQDestination.h> #include  
<activemq/commands/BaseCommand.h> #include <activemq/commands/-  
BrokerId.h> #include <activemq/commands/ProducerId.h>
```

```
#include <activemq/commands/RemoveInfo.h> #include <activemq/util/-  
Config.h> #include <decaf/lang/Pointer.h> #include <string> x  
#include <vector>
```

Data Structures

- class **activemq::commands::ProducerInfo**

Namespaces

- namespace **activemq**
*Licensed to the Apache Software Foundation (ASF) under one or more contributor
license agreements.*
- namespace **activemq::commands**

7.71 src/main/activemq/commands/RemoveInfo.h File Reference

```
#include <activemq/commands/BaseCommand.h> #include <activemq/commands/-  
DataStructure.h> #include <activemq/util/Config.h> #include  
<decaf/lang/Pointer.h> #include <string> #include <vector> x
```

Data Structures

- class **activemq::commands::RemoveInfo**

Namespaces

- namespace **activemq**
*Licensed to the Apache Software Foundation (ASF) under one or more contributor
license agreements.*
- namespace **activemq::commands**

7.72 src/main/activemq/commands/RemoveSubscriptionInfo.h File Reference

```
#include <activemq/commands/BaseCommand.h> #include <activemq/commands/-  
ConnectionId.h> #include <activemq/util/Config.h> #include  
<decaf/lang/Pointer.h> #include <string> #include <vector> x
```

Data Structures

- class **activemq::commands::RemoveSubscriptionInfo**

Namespaces

- namespace **activemq**

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

- namespace **activemq::commands**

7.73 src/main/activemq/commands/ReplayCommand.h File Reference

```
#include <activemq/commands/BaseCommand.h> #include <activemq/util/-  
Config.h> #include <decaf/lang/Pointer.h> #include <string> x  
#include <vector>
```

Data Structures

- class **activemq::commands::ReplayCommand**

Namespaces

- namespace **activemq**

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

- namespace **activemq::commands**

7.74 src/main/activemq/commands/Response.h File Reference

```
#include <activemq/commands/BaseCommand.h> #include <activemq/util/-  
Config.h> #include <decaf/lang/Pointer.h> #include <string> x  
#include <vector>
```

Data Structures

- class **activemq::commands::Response**

Namespaces

- namespace **activemq**
Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.
- namespace **activemq::commands**

7.75 src/main/activemq/commands/SessionId.h File Reference

```
#include <activemq/commands/BaseDataStructure.h> #include  
<activemq/commands/ConnectionId.h> #include <activemq/util/-  
Config.h> #include <decaf/lang/Comparable.h> #include  
<decaf/lang/Pointer.h> #include <string> #include <vector> x
```

Data Structures

- class **activemq::commands::SessionId**

Namespaces

- namespace **activemq**
Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.
- namespace **activemq::commands**

7.76 src/main/activemq/commands/SessionInfo.h File Reference

```
#include <activemq/commands/BaseCommand.h> #include <activemq/commands/-  
RemoveInfo.h> #include <activemq/commands/SessionId.h> x  
#include <activemq/util/Config.h> #include <decaf/lang/-  
Pointer.h> #include <string> #include <vector>
```

Data Structures

- class **activemq::commands::SessionInfo**

Namespaces

- namespace **activemq**
Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.
- namespace **activemq::commands**

7.77 src/main/activemq/commands/ShutdownInfo.h File Reference

```
#include <activemq/commands/BaseCommand.h> #include <activemq/util/-  
Config.h> #include <decaf/lang/Pointer.h> #include <string> x  
#include <vector>
```

Data Structures

- class **activemq::commands::ShutdownInfo**

Namespaces

- namespace **activemq**
Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.
- namespace **activemq::commands**

7.78 src/main/activemq/commands/SubscriptionInfo.h File Reference

```
#include <activemq/commands/ActiveMQDestination.h> #include  
<activemq/commands/BaseDataStructure.h> #include <activemq/util/-  
Config.h> #include <decaf/lang/Pointer.h> #include <string> x  
#include <vector>
```

Data Structures

- class **activemq::commands::SubscriptionInfo**

Namespaces

- namespace **activemq**
Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.
- namespace **activemq::commands**

7.79 src/main/activemq/commands/TransactionId.h File Reference

```
#include <activemq/commands/BaseDataStructure.h> #include  
<activemq/util/Config.h> #include <decaf/lang/Comparable.-  
h> #include <decaf/lang/Pointer.h> #include <string> x  
#include <vector>
```

Data Structures

- class **activemq::commands::TransactionId**

Namespaces

- namespace **activemq**

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

- namespace **activemq::commands**

7.80 src/main/activemq/commands/TransactionInfo.h File Reference

```
#include <activemq/commands/BaseCommand.h> #include <activemq/commands/-  
ConnectionId.h> #include <activemq/commands/Transaction-  
Id.h> #include <activemq/util/Config.h> #include <decaf/lang/-  
Pointer.h> #include <string> #include <vector>
```

Data Structures

- class **activemq::commands::TransactionInfo**

Namespaces

- namespace **activemq**

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

- namespace **activemq::commands**

7.81 src/main/activemq/commands/WireFormatInfo.h File Reference

```
#include <activemq/util/Config.h> #include <activemq/commands/-  
BaseCommand.h> #include <activemq/util/PrimitiveMap.-  
h> #include <activemq/exceptions/ActiveMQException.h> ×  
#include <vector>
```

Data Structures

- class **activemq::commands::WireFormatInfo**

Namespaces

- namespace **activemq**
Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.
- namespace **activemq::commands**

7.82 src/main/activemq/commands/XATransactionId.h File Reference

```
#include <activemq/commands/TransactionId.h>    #include
<activemq/util/Config.h>    #include <cms/Xid.h>    #include
<decaf/lang/Comparable.h>    #include <decaf/lang/Pointer.-
h> #include <string> #include <vector>
```

Data Structures

- class **activemq::commands::XATransactionId**

Namespaces

- namespace **activemq**
Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.
- namespace **activemq::commands**

7.83 src/main/activemq/core/ActiveMQAckHandler.h File Reference

```
#include <cms/CMSException.h>    #include <activemq/util/-
Config.h>
```

Data Structures

- class **activemq::core::ActiveMQAckHandler**
Interface class that is used to give CMS Messages an interface to Ack themselves with.

Namespaces

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

- namespace **activemq::commands**
- namespace **activemq::core**

7.84 src/main/activemq/core/ActiveMQConnection.h File Reference

```
#include <cms/Connection.h>      #include <activemq/util/-
Config.h> #include <activemq/core/ActiveMQConnectionMeta-
Data.h> #include <activemq/core/Dispatcher.h> #include
<activemq/commands/ActiveMQTempDestination.h> #include
<activemq/commands/ConnectionInfo.h> #include <activemq/commands/-
ConsumerInfo.h> #include <activemq/commands/SessionId.-
h> #include <activemq/exceptions/ActiveMQException.h>
#include <activemq/transport/Transport.h> #include <activemq/transport/-
TransportListener.h> #include <activemq/threads/Scheduler.-
h> #include <decaf/util/Properties.h> #include <decaf/util/concurrent/atomic/-
AtomicBoolean.h> #include <decaf/util/concurrent/Copy-
OnWriteArrayList.h> #include <decaf/lang/exceptions/-
UnsupportedOperationException.h> #include <decaf/lang/exceptions/-
NullPointerException.h> #include <decaf/lang/exceptions/-
IllegalStateException.h> #include <string> #include <memory> x
```

Data Structures

- class **activemq::core::ActiveMQConnection**
Concrete connection used for all connectors to the ActiveMQ broker.

Namespaces

- namespace **activemq**
Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.
- namespace **activemq::core**

7.85 src/main/activemq/core/ActiveMQConnectionFactory.h File Reference

```
#include <activemq/util/Config.h> #include <cms/Connection-
Factory.h> #include <cms/Connection.h> #include <activemq/transport/-
Transport.h> #include <decaf/net/URI.h> #include <decaf/util/-
Properties.h>
```

Data Structures

- class **activemq::core::ActiveMQConnectionFactory**

Namespaces

- namespace **activemq**
Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.
- namespace **activemq::core**

7.86 src/main/activemq/core/ActiveMQConnectionMetaData.h File Reference

```
#include <activemq/util/Config.h> #include <cms/Connection-  
MetaData.h>
```

Data Structures

- class **activemq::core::ActiveMQConnectionMetaData**
*This class houses all the various settings and information that is used by an instance of an **ActiveMQConnection** (p. 187) class.*

Namespaces

- namespace **activemq**
Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.
- namespace **activemq::core**

7.87 src/main/activemq/core/ActiveMQConstants.h File Reference

```
#include <string> #include <map> #include <activemq/util/-  
Config.h>
```

Data Structures

- class **activemq::core::ActiveMQConstants**
Class holding constant values for various ActiveMQ specific things Each constant is defined as an enumeration and has functions that convert back and forth between string and enum values.
- class **activemq::core::ActiveMQConstants::StaticInitializer**

Namespaces

- namespace **activemq**

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

- namespace **activemq::core**

7.88 src/main/activemq/core/ActiveMQConsumer.h File Reference

```
#include <cms/MessageConsumer.h> #include <cms/Message-
Listener.h> #include <cms/Message.h> #include <cms/CMS-
Exception.h> #include <activemq/util/Config.h> #include
<activemq/exceptions/ActiveMQException.h> #include <activemq/commands/-
ConsumerInfo.h> #include <activemq/commands/MessageAck.-
h> #include <activemq/commands/MessageDispatch.h> #include
<activemq/core/Dispatcher.h> #include <activemq/core/-
RedeliveryPolicy.h> #include <activemq/core/MessageDispatch-
Channel.h> #include <decaf/util/concurrent/atomic/Atomic-
Boolean.h> #include <decaf/lang/Pointer.h> #include <decaf/util/concurrent/-
Mutex.h>
```

Data Structures

- class **activemq::core::ActiveMQConsumer**

Namespaces

- namespace **activemq**

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

- namespace **activemq::core**

7.89 src/main/activemq/core/ActiveMQProducer.h File Reference

```
#include <cms/MessageProducer.h> #include <cms/Message.-
h> #include <cms/Destination.h> #include <cms/Delivery-
Mode.h> #include <activemq/util/Config.h> #include <activemq/util/-
MemoryUsage.h> #include <activemq/commands/ProducerInfo.-
h> #include <activemq/commands/ProducerAck.h> #include
<activemq/exceptions/ActiveMQException.h> #include <memory> x
```

Data Structures

- class **activemq::core::ActiveMQProducer**

Namespaces

- namespace **activemq**
Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.
- namespace **activemq::core**

7.90 src/main/activemq/core/ActiveMQQueueBrowser.h File Reference

```
#include <activemq/util/Config.h>    #include <cms/Queue.-
h> #include <cms/QueueBrowser.h>    #include <cms/Message-
Enumeration.h>    #include <activemq/commands/ConsumerId.-
h>    #include <activemq/commands/ActiveMQDestination.h>
#include <decaf/lang/Pointer.h> #include <decaf/util/concurrent/-
Mutex.h>    #include <decaf/util/concurrent/atomic/Atomic-
Boolean.h> #include <string>
```

Data Structures

- class **activemq::core::ActiveMQQueueBrowser**

Namespaces

- namespace **activemq**
Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.
- namespace **activemq::core**

7.91 src/main/activemq/core/ActiveMQSession.h File Reference

```
#include <cms/Session.h> #include <cms/ExceptionListener.-
h> #include <activemq/util/Config.h> #include <activemq/util/-
Usage.h> #include <activemq/exceptions/ActiveMQException.-
h>    #include <activemq/core/ActiveMQTransactionContext.-
h>    #include <activemq/commands/ActiveMQTempDestination.-
h> #include <activemq/commands/Response.h> #include <activemq/commands/-
SessionInfo.h> #include <activemq/commands/ConsumerInfo.-
h>    #include <activemq/commands/ConsumerId.h>    #include
```

7.92 src/main/activemq/core/ActiveMQSessionExecutor.h File Reference 3007

```
<activemq/commands/ProducerId.h> #include <activemq/commands/-  
TransactionId.h> #include <activemq/core/Dispatcher.-  
h> #include <activemq/core/MessageDispatchChannel.h> ×  
#include <activemq/util/LongSequenceGenerator.h> #include  
<activemq/threads/Scheduler.h> #include <decaf/lang/-  
Pointer.h> #include <decaf/util/StlMap.h> #include <decaf/util/-  
Properties.h> #include <decaf/util/concurrent/atomic/-  
AtomicBoolean.h> #include <decaf/util/concurrent/CopyOn-  
WriteArrayList.h> #include <string> #include <memory>
```

Data Structures

- class **activemq::core::ActiveMQSession**

Namespaces

- namespace **activemq**
*Licensed to the Apache Software Foundation (ASF) under one or more contributor
license agreements.*
- namespace **activemq::core**

7.92 src/main/activemq/core/ActiveMQSessionExecutor.h File - Reference

```
#include <activemq/util/Config.h> #include <activemq/core/-  
MessageDispatchChannel.h> #include <activemq/commands/-  
ConsumerId.h> #include <activemq/commands/MessageDispatch.-  
h> #include <activemq/threads/Task.h> #include <activemq/threads/-  
TaskRunner.h> #include <decaf/lang/Pointer.h>
```

Data Structures

- class **activemq::core::ActiveMQSessionExecutor**
Delegate dispatcher for a single session.

Namespaces

- namespace **activemq**
*Licensed to the Apache Software Foundation (ASF) under one or more contributor
license agreements.*
- namespace **activemq::core**

7.93 src/main/activemq/core/ActiveMQTransactionContext.h File - Reference

```
#include <memory> #include <cms/Message.h> #include <cms/-
XAResource.h> #include <cms/CMSException.h> #include
<cms/XAException.h> #include <activemq/util/Config.-
h> #include <activemq/exceptions/ActiveMQException.h>
#include <activemq/commands/LocalTransactionId.h> #include
<activemq/core/Synchronization.h> #include <activemq/util/-
LongSequenceGenerator.h> #include <decaf/lang/exceptions/-
InvalidStateException.h> #include <decaf/util/StlSet.h>
#include <decaf/util/Properties.h> #include <decaf/util/concurrent/-
Mutex.h>
```

Data Structures

- class **activemq::core::ActiveMQTransactionContext**

Transaction Management class, hold messages that are to be redelivered upon a request to roll-back.

Namespaces

- namespace **activemq**

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

- namespace **activemq::core**

7.94 src/main/activemq/core/ActiveMQXAConnection.h File - Reference

```
#include <activemq/util/Config.h> #include <cms/XAConnection.-
h> #include <activemq/core/ActiveMQConnection.h>
```

Data Structures

- class **activemq::core::ActiveMQXAConnection**

Namespaces

- namespace **activemq**

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

- namespace **activemq::core**

7.95 src/main/activemq/core/ActiveMQXAConnectionFactory.h File Reference 8009

7.95 src/main/activemq/core/ActiveMQXAConnectionFactory.h File Reference

```
#include <activemq/util/Config.h> #include <cms/XAConnectionFactory.h>
#include <activemq/core/ActiveMQConnectionFactory.h> #include <decaf/net/URI.h>
#include <decaf/util/Properties.h>
```

Data Structures

- class **activemq::core::ActiveMQXAConnectionFactory**

Namespaces

- namespace **activemq**
Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.
- namespace **activemq::core**

7.96 src/main/activemq/core/ActiveMQXASession.h File Reference

```
#include <activemq/util/Config.h> #include <cms/XASession.h>
#include <activemq/core/ActiveMQSession.h>
```

Data Structures

- class **activemq::core::ActiveMQXASession**

Namespaces

- namespace **activemq**
Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.
- namespace **activemq::core**

7.97 src/main/activemq/core/DispatchData.h File Reference

```
#include <stdlib.h> #include <memory> #include <activemq/util/Config.h>
#include <activemq/commands/ConsumerId.h> #include <activemq/commands/Message.h>
#include <decaf/lang/Pointer.h>
```

Data Structures

- class **activemq::core::DispatchData**

Simple POCO that contains the information necessary to route a message to a specified consumer.

Namespaces

- namespace **activemq**

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

- namespace **activemq::core**

7.98 src/main/activemq/core/Dispatcher.h File Reference

```
#include <activemq/commands/MessageDispatch.h>    #include  
<activemq/util/Config.h>    #include <decaf/lang/Pointer.-  
h>
```

Data Structures

- class **activemq::core::Dispatcher**

Interface for an object responsible for dispatching messages to consumers.

Namespaces

- namespace **activemq**

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

- namespace **activemq::core**

7.99 src/main/activemq/core/FifoMessageDispatchChannel.h File - Reference

```
#include <activemq/util/Config.h> #include <activemq/core/-  
MessageDispatchChannel.h>    #include <decaf/util/Linked-  
List.h> #include <decaf/lang/Pointer.h>
```

Data Structures

- class **activemq::core::FifoMessageDispatchChannel**

7.100 src/main/activemq/core/MessageDispatchChannel.h File Reference 3011

Namespaces

- namespace **activemq**
Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.
- namespace **activemq::core**

7.100 src/main/activemq/core/MessageDispatchChannel.h File - Reference

```
#include <activemq/util/Config.h> #include <activemq/commands/-  
MessageDispatch.h> #include <decaf/util/concurrent/Mutex.-  
h> #include <decaf/util/concurrent/Synchronizable.h> ×  
#include <decaf/lang/Pointer.h>
```

Data Structures

- class **activemq::core::MessageDispatchChannel**

Namespaces

- namespace **activemq**
Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.
- namespace **activemq::core**

7.101 src/main/activemq/core/policies/DefaultPrefetchPolicy.h File Reference

```
#include <activemq/util/Config.h> #include <activemq/core/-  
PrefetchPolicy.h>
```

Data Structures

- class **activemq::core::policies::DefaultPrefetchPolicy**

Namespaces

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

- namespace **activemq::core**
- namespace **activemq::core::policies**

7.102 src/main/activemq/core/policies/DefaultRedeliveryPolicy.h File Reference

```
#include <activemq/util/Config.h> #include <activemq/core/-  
RedeliveryPolicy.h>
```

Data Structures

- class **activemq::core::policies::DefaultRedeliveryPolicy**

Namespaces

- namespace **activemq**
Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.
- namespace **activemq::core**
- namespace **activemq::core::policies**

7.103 src/main/activemq/core/PrefetchPolicy.h File Reference

```
#include <activemq/util/Config.h> #include <decaf/util/-  
Properties.h>
```

Data Structures

- class **activemq::core::PrefetchPolicy**
Interface for a Policy object that controls message Prefetching on various destination types in ActiveMQ-CPP.

Namespaces

- namespace **activemq**
Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.
- namespace **activemq::core**

7.104 src/main/activemq/core/RedeliveryPolicy.h File Reference

```
#include <activemq/util/Config.h> #include <decaf/util/-  
Properties.h>
```

Data Structures

- class **activemq::core::RedeliveryPolicy**

Interface for a **RedeliveryPolicy** (p.2250) object that controls how message - Redelivery is handled in ActiveMQ-CPP when a transaction is rolled back.

Namespaces

- namespace **activemq**
Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.
- namespace **activemq::core**

7.105 src/main/activemq/core/SimplePriorityMessageDispatch- Channel.h File Reference

```
#include <activemq/util/Config.h> #include <activemq/core/-  
MessageDispatchChannel.h> #include <decaf/util/Linked-  
List.h> #include <decaf/lang/ArrayPointer.h> #include  
<decaf/util/concurrent/Mutex.h>
```

Data Structures

- class **activemq::core::SimplePriorityMessageDispatchChannel**

Namespaces

- namespace **activemq**
Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.
- namespace **activemq::core**

7.106 src/main/activemq/core/Synchronization.h File Reference

```
#include <activemq/util/Config.h> #include <activemq/exceptions/-  
ActiveMQException.h>
```

Data Structures

- class **activemq::core::Synchronization**
*Transacted Object **Synchronization** (p. 2654), used to sync the events of a - Transaction with the items in the Transaction.*

Namespaces

- namespace **activemq**
Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.
- namespace **activemq::core**

7.107 src/main/activemq/exceptions/ActiveMQException.h File - Reference

```
#include <activemq/util/Config.h>      #include <cms/CMS-Exception.h>
#include <decaf/lang/Exception.h>      #include <activemq/exceptions/ExceptionDefines.h>
#include <stdarg.h> #include <sstream>
```

Data Structures

- class **activemq::exceptions::ActiveMQException**

Namespaces

- namespace **activemq**
Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.
- namespace **activemq::exceptions**

7.108 src/main/activemq/exceptions/BrokerException.h File - Reference

```
#include <activemq/util/Config.h> #include <activemq/exceptions/-ActiveMQException.h>
#include <activemq/commands/BrokerError.h> #include <sstream>
```

Data Structures

- class **activemq::exceptions::BrokerException**

7.109 src/main/activemq/exceptions/ConnectionFailedException.h File Reference

Namespaces

- namespace **activemq**
Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.
- namespace **activemq::exceptions**

7.109 src/main/activemq/exceptions/ConnectionFailedException.h File Reference

```
#include <activemq/util/Config.h> #include <activemq/exceptions/ActiveMQException.h>
```

Data Structures

- class **activemq::exceptions::ConnectionFailedException**

Namespaces

- namespace **activemq**
Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.
- namespace **activemq::exceptions**

7.110 src/main/activemq/exceptions/ExceptionDefines.h File Reference

Defines

- #define **AMQ_CATCH_RETHROW**(type)
Macro for catching and re-throwing an exception of a given type.
- #define **AMQ_CATCH_EXCEPTION_CONVERT**(sourceType, targetType)
Macro for catching an exception of one type and then re-throwing as another type.
- #define **AMQ_CATCHALL_THROW**(type)
A catch-all that throws a known exception.
- #define **AMQ_CATCHALL_NOTHROW**()
A catch-all that does not throw an exception, one use would be to catch any exception in a destructor and mark it, but not throw so that cleanup would continue as normal.
- #define **AMQ_CATCH_NOTHROW**(type)
Macro for catching and re-throwing an exception of a given type.

7.110.1 Define Documentation

7.110.1.1 `#define AMQ_CATCH_EXCEPTION_CONVERT(sourceType, targetType)`

Value:

```
catch( sourceType& ex ){ \
    targetType target( ex ); \
    target.setMark( __FILE__, __LINE__ ); \
    throw target; \
}
```

Macro for catching an exception of one type and then re-throwing as another type.

Parameters

<i>sourceType</i>	the type of the exception to be caught.
<i>targetType</i>	the type of the exception to be thrown.

Referenced by `activemq::wireformat::openwire::marshal::BaseDataStreamMarshaller::looseMarshalObjectArray()`, `activemq::wireformat::openwire::marshal::BaseDataStreamMarshaller::tightMarshalObjectArray1()`, and `activemq::wireformat::openwire::marshal::BaseDataStreamMarshaller::tightMarshalObjectArray2()`.

7.110.1.2 `#define AMQ_CATCH_NOTHROW(type)`

Value:

```
catch( type& ex ){ \
    ex.setMark( __FILE__, __LINE__ ); \
}
```

Macro for catching and re-throwing an exception of a given type.

Parameters

<i>type</i>	The type of the exception to throw (e.g. <code>ActiveMQException</code>).
-------------	--

7.110.1.3 `#define AMQ_CATCH_RETHROW(type)`

Value:

```
catch( type& ex ){ \
    ex.setMark( __FILE__, __LINE__ ); \
    throw; \
}
```

Macro for catching and re-throwing an exception of a given type.

7.111 src/main/decaf/lang/exceptions/ExceptionDefines.h File Reference 3017

Parameters

<i>type</i>	The type of the exception to throw (e.g. <code>ActiveMQException</code>).
-------------	--

Referenced by `activemq::wireformat::openwire::marshal::BaseDataStreamMarshaller::looseMarshalObjectArray()`, `activemq::wireformat::openwire::marshal::BaseDataStreamMarshaller::tightMarshalObjectArray1()`, and `activemq::wireformat::openwire::marshal::BaseDataStreamMarshaller::tightMarshalObjectArray2()`.

7.110.1.4 #define AMQ_CATCHALL_NOTHROW()

Value:

```
catch( ... ){ \
    activemq::exceptions::ActiveMQException ex( __FILE__, __LINE__, \
        "caught unknown exception, not rethrowing" ); \
}
```

A catch-all that does not throw an exception, one use would be to catch any exception in a destructor and mark it, but not throw so that cleanup would continue as normal.

7.110.1.5 #define AMQ_CATCHALL_THROW(type)

Value:

```
catch( ... ){ \
    type ex( __FILE__, __LINE__, \
        "caught unknown exception" ); \
    throw ex; \
}
```

A catch-all that throws a known exception.

Parameters

<i>type</i>	the type of exception to be thrown.
-------------	-------------------------------------

Referenced by `activemq::wireformat::openwire::marshal::BaseDataStreamMarshaller::looseMarshalObjectArray()`, `activemq::wireformat::openwire::marshal::BaseDataStreamMarshaller::tightMarshalObjectArray1()`, and `activemq::wireformat::openwire::marshal::BaseDataStreamMarshaller::tightMarshalObjectArray2()`.

7.111 src/main/decaf/lang/exceptions/ExceptionDefines.h File - Reference

Defines

- #define **DECAF_CATCH_RETHROW**(type)

Macro for catching and rethrowing an exception of a given type.

- **#define DECAF_CATCH_EXCEPTION_CONVERT**(sourceType, targetType)

Macro for catching an exception of one type and then rethrowing as another type.

- **#define DECAF_CATCHALL_THROW**(type)

A catch-all that throws a known exception.

- **#define DECAF_CATCHALL_NOTHROW**()

A catch-all that does not throw an exception, one use would be to catch any exception in a destructor and mark it, but not throw so that cleanup would continue as normal.

- **#define DECAF_CATCH_NOTHROW**(type)

Macro for catching and rethrowing an exception of a given type.

7.111.1 Define Documentation

7.111.1.1 **#define DECAF_CATCH_EXCEPTION_CONVERT**(sourceType, targetType)

Value:

```
catch( sourceType& ex ){ \
    targetType target( &ex ); \
    target.setMark( __FILE__, __LINE__ ); \
    throw target; \
}
```

Macro for catching an exception of one type and then rethrowing as another type.

Parameters

<i>sourceType</i>	the type of the exception to be caught.
<i>targetType</i>	the type of the exception to be thrown.

Referenced by `decaf::util::PriorityQueue< E >::add()`.

7.111.1.2 **#define DECAF_CATCH_NOTHROW**(type)

Value:

```
catch( type& ex ){ \
    ex.setMark( __FILE__, __LINE__ ); \
}
```

Macro for catching and rethrowing an exception of a given type.

Parameters

<i>type</i>	The type of the exception to throw (e.g. Exception).
-------------	---

7.111.1.3 **#define DECAF_CATCH_RETHROW(type)****Value:**

```
catch( type& ex ){ \
    ex.setMark( __FILE__, __LINE__ ); \
    throw; \
}
```

Macro for catching and rethrowing an exception of a given type.

Parameters

<i>type</i>	The type of the exception to throw (e.g. Exception).
-------------	---

Referenced by `decaf::util::PriorityQueue< E >::add()`, and `decaf::util::concurrent::LinkedBlockingQueue< E >::LinkedBlockingQueue()`.

7.111.1.4 **#define DECAF_CATCHALL_NOTHROW()****Value:**

```
catch( ... ){ \
    lang::Exception ex( __FILE__, __LINE__, \
        "caught unknown exception, not rethrowing" ); \
}
```

A catch-all that does not throw an exception, one use would be to catch any exception in a destructor and mark it, but not throw so that cleanup would continue as normal.

Referenced by `decaf::util::ArrayList< E >::~~ArrayList()`.

7.111.1.5 **#define DECAF_CATCHALL_THROW(type)****Value:**

```
catch( ... ){ \
    type ex( __FILE__, __LINE__, \
        "caught unknown exception" ); \
    throw ex; \
}
```

A catch-all that throws a known exception.

Parameters

<i>type</i>	the type of exception to be thrown.
-------------	-------------------------------------

Referenced by `decaf::util::PriorityQueue< E >::add()`, and `decaf::util::concurrent::LinkedBlockingQueue< E >::LinkedBlockingQueue()`.

7.112 src/main/activemq/io/LoggingInputStream.h File Reference

```
#include <activemq/util/Config.h>    #include <decaf/io/-  
FilterInputStream.h> #include <decaf/util/logging/Logger-  
Defines.h>    #include <decaf/lang/exceptions/NullPointer-  
Exception.h>
```

Data Structures

- class **activemq::io::LoggingInputStream**

Namespaces

- namespace **activemq**
Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.
- namespace **activemq::io**

7.113 src/main/activemq/io/LoggingOutputStream.h File Reference

```
#include <activemq/util/Config.h>    #include <decaf/io/-  
FilterOutputStream.h> #include <decaf/util/logging/Logger-  
Defines.h>
```

Data Structures

- class **activemq::io::LoggingOutputStream**
OutputStream filter that just logs the data being written.

Namespaces

- namespace **activemq**
Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.
- namespace **activemq::io**

7.114 src/main/activemq/library/ActiveMQCPP.h File Reference

```
#include <activemq/util/Config.h>
```

Data Structures

- class **activemq::library::ActiveMQCPP**

Namespaces

- namespace **activemq**
Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.
- namespace **activemq::library**

7.115 src/main/activemq/state/CommandVisitor.h File Reference

```
#include <activemq/util/Config.h> #include <activemq/exceptions/-  
ActiveMQException.h> #include <decaf/lang/Pointer.h>
```

Data Structures

- class **activemq::state::CommandVisitor**
Interface for an Object that can visit the various Command Objects that are sent from and to this client.

Namespaces

- namespace **activemq**
Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.
- namespace **activemq::commands**
- namespace **activemq::state**

7.116 src/main/activemq/state/CommandVisitorAdapter.h File Reference

```
#include <activemq/util/Config.h> #include <activemq/state/-  
CommandVisitor.h> #include <activemq/core/ActiveMQConstants.-  
h> #include <activemq/commands/ConnectionInfo.h> #include  
<activemq/commands/SessionInfo.h> #include <activemq/commands/-  
ProducerInfo.h> #include <activemq/commands/Consumer-  
Info.h> #include <activemq/commands/ConnectionId.h> ×  
#include <activemq/commands/SessionId.h> #include <activemq/commands/-  
ProducerId.h> #include <activemq/commands/ConsumerId.h> ×  
#include <activemq/commands/DestinationInfo.h> #include  
<activemq/commands/RemoveSubscriptionInfo.h> #include
```

```

<activemq/commands/Message.h> #include <activemq/commands/-
MessageAck.h> #include <activemq/commands/MessagePull.-
h> #include <activemq/commands/TransactionInfo.h> #include
<activemq/commands/WireFormatInfo.h> #include <activemq/commands/-
ProducerAck.h> #include <activemq/commands/MessageDispatch.-
h> #include <activemq/commands/MessageDispatchNotification.-
h> #include <activemq/commands/ControlCommand.h> #include
<activemq/commands/ConnectionError.h> #include <activemq/commands/-
ConnectionControl.h> #include <activemq/commands/Consumer-
Control.h> #include <activemq/commands/ShutdownInfo.-
h> #include <activemq/commands/KeepAliveInfo.h> #include
<activemq/commands/FlushCommand.h> #include <activemq/commands/-
BrokerError.h> #include <activemq/commands/BrokerInfo.-
h> #include <activemq/commands/RemoveInfo.h> #include
<activemq/commands/ReplayCommand.h> #include <activemq/commands/-
Response.h>

```

Data Structures

- class **activemq::state::CommandVisitorAdapter**

*Default Implementation of a **CommandVisitor** (p. 872) that returns NULL for all calls.*

Namespaces

- namespace **activemq**

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

- namespace **activemq::state**

7.117 src/main/activemq/state/ConnectionState.h File Reference

```

#include <activemq/util/Config.h> #include <activemq/commands/-
ConnectionInfo.h> #include <activemq/commands/Destination-
Info.h> #include <activemq/commands/SessionInfo.h> #include
<activemq/commands/ConsumerId.h> #include <activemq/commands/-
ProducerId.h> #include <activemq/commands/Transaction-
Id.h> #include <activemq/commands/LocalTransactionId.-
h> #include <activemq/state/ConsumerState.h> #include
<activemq/state/ProducerState.h> #include <activemq/state/-
SessionState.h> #include <activemq/state/Transaction-
State.h> #include <decaf/util/StlMap.h> #include <decaf/util/concurrent/atc
AtomicBoolean.h> #include <decaf/util/concurrent/Concurrent-
StlMap.h> #include <decaf/util/LinkedList.h> #include
<decaf/lang/Pointer.h> #include <string> #include <memory> ×

```

Data Structures

- class **activemq::state::ConnectionState**

Namespaces

- namespace **activemq**
Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.
- namespace **activemq::state**

7.118 src/main/activemq/state/ConnectionStateTracker.h File - Reference

```
#include <activemq/util/Config.h> #include <activemq/commands/-
ConnectionId.h> #include <activemq/exceptions/ActiveM-
QException.h> #include <activemq/state/CommandVisitor-
Adapter.h> #include <activemq/state/ConnectionState.-
h> #include <activemq/state/ConsumerState.h> #include
<activemq/state/ProducerState.h> #include <activemq/state/-
SessionState.h> #include <activemq/state/Transaction-
State.h> #include <activemq/state/Tracked.h> #include
<activemq/transport/Transport.h> #include <decaf/util/concurrent/-
ConcurrentStlMap.h> #include <decaf/lang/Pointer.h>
```

Data Structures

- class **activemq::state::ConnectionStateTracker**

Namespaces

- namespace **activemq**
Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.
- namespace **activemq::state**

7.119 src/main/activemq/state/ConsumerState.h File Reference

```
#include <activemq/util/Config.h> #include <activemq/commands/-
ConsumerInfo.h> #include <decaf/lang/Pointer.h> #include
<string> #include <memory>
```

Data Structures

- class **activemq::state::ConsumerState**

Namespaces

- namespace **activemq**
Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.
- namespace **activemq::state**

7.120 src/main/activemq/state/ProducerState.h File Reference

```
#include <activemq/util/Config.h> #include <activemq/commands/-  
ProducerInfo.h> #include <decaf/lang/Pointer.h> #include  
<string> #include <memory>
```

Data Structures

- class **activemq::state::ProducerState**

Namespaces

- namespace **activemq**
Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.
- namespace **activemq::state**

7.121 src/main/activemq/state/SessionState.h File Reference

```
#include <activemq/util/Config.h> #include <activemq/commands/-  
SessionInfo.h> #include <activemq/commands/ConsumerId.-  
h> #include <activemq/commands/ProducerId.h> #include  
<activemq/state/ConsumerState.h> #include <activemq/state/-  
ProducerState.h> #include <decaf/util/concurrent/atomic/-  
AtomicBoolean.h> #include <decaf/util/concurrent/Concurrent-  
StlMap.h> #include <string> #include <memory>
```

Data Structures

- class **activemq::state::SessionState**

Namespaces

- namespace **activemq**
Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.
- namespace **activemq::state**

7.122 src/main/activemq/state/Tracked.h File Reference

```
#include <activemq/util/Config.h> #include <activemq/commands/-  
Response.h> #include <decaf/lang/Runnable.h> #include  
<decaf/lang/Pointer.h>
```

Data Structures

- class **activemq::state::Tracked**

Namespaces

- namespace **activemq**
Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.
- namespace **activemq::state**

7.123 src/main/activemq/state/TransactionState.h File Reference

```
#include <activemq/util/Config.h> #include <activemq/commands/-  
Command.h> #include <activemq/commands/ProducerId.h> ×  
#include <activemq/commands/TransactionId.h> #include  
<decaf/lang/Pointer.h> #include <decaf/util/LinkedList.-  
h> #include <decaf/util/concurrent/atomic/AtomicBoolean.-  
h> #include <decaf/util/concurrent/ConcurrentStlMap.h> ×  
#include <string> #include <memory>
```

Data Structures

- class **activemq::state::TransactionState**

Namespaces

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

- namespace **activemq::state**

7.124 src/main/activemq/threads/CompositeTask.h File Reference

```
#include <activemq/util/Config.h> #include <activemq/threads/-  
Task.h>
```

Data Structures

- class **activemq::threads::CompositeTask**
*Represents a single task that can be part of a set of Tasks that are contained in a **CompositeTaskRunner** (p. 893).*

Namespaces

- namespace **activemq**
Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.
- namespace **activemq::threads**

7.125 src/main/activemq/threads/CompositeTaskRunner.h File - Reference

```
#include <activemq/util/Config.h> #include <activemq/threads/-  
TaskRunner.h> #include <activemq/threads/CompositeTask.-  
h> #include <decaf/util/StlSet.h> #include <decaf/util/-  
LinkedList.h> #include <decaf/lang/Thread.h> #include  
<decaf/lang/Runnable.h> #include <decaf/util/concurrent/-  
Mutex.h> #include <decaf/lang/Pointer.h>
```

Data Structures

- class **activemq::threads::CompositeTaskRunner**
*A **Task** (p. 2676) Runner that can contain one or more CompositeTasks that are each checked for pending work and run if any is present in the order that the tasks were added.*

Namespaces

- namespace **activemq**
Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.
- namespace **activemq::threads**

7.126 src/main/activemq/threads/DedicatedTaskRunner.h File Reference

```
#include <activemq/util/Config.h> #include <activemq/threads/-
TaskRunner.h> #include <activemq/threads/Task.h> #include
<decaf/lang/Thread.h> #include <decaf/lang/Runnable.-
h> #include <decaf/util/concurrent/Mutex.h> #include
<decaf/lang/Pointer.h>
```

Data Structures

- class **activemq::threads::DedicatedTaskRunner**

Namespaces

- namespace **activemq**

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

- namespace **activemq::threads**

7.127 src/main/activemq/threads/Scheduler.h File Reference

```
#include <activemq/util/Config.h> #include <activemq/util/-
ServiceSupport.h> #include <decaf/lang/Runnable.h> #include
<decaf/util/Timer.h> #include <decaf/util/StlMap.h> x
#include <decaf/util/concurrent/Mutex.h> #include <string> x
```

Data Structures

- class **activemq::threads::Scheduler**

Scheduler (p. 2317) class for use in executing Runnable Tasks either periodically or one time only with optional delay.

Namespaces

- namespace **activemq**

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

- namespace **activemq::threads**

7.128 src/main/activemq/threads/SchedulerTimerTask.h File Reference

```
#include <activemq/util/Config.h> #include <decaf/util/-  
TimerTask.h> #include <decaf/lang/Runnable.h>
```

Data Structures

- class **activemq::threads::SchedulerTimerTask**

Extension of the Decaf TimerTask that adds a Runnable instance which is the target of this task.

Namespaces

- namespace **activemq**

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

- namespace **activemq::threads**

7.129 src/main/activemq/threads/Task.h File Reference

```
#include <activemq/util/Config.h>
```

Data Structures

- class **activemq::threads::Task**

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

Namespaces

- namespace **activemq**

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

- namespace **activemq::threads**

7.130 src/main/activemq/threads/TaskRunner.h File Reference

```
#include <activemq/util/Config.h> #include <activemq/threads/-  
Task.h>
```

7.131 src/main/activemq/transport/AbstractTransportFactory.h File Reference 8029

Data Structures

- class **activemq::threads::TaskRunner**

Namespaces

- namespace **activemq**
Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.
- namespace **activemq::threads**

7.131 src/main/activemq/transport/AbstractTransportFactory.h File Reference

```
#include <activemq/util/Config.h> #include <activemq/transport/-  
Transport.h> #include <activemq/transport/TransportFactory.-  
h> #include <activemq/wireformat/WireFormat.h> #include  
<decaf/util/NoSuchElementException.h> #include <decaf/lang/-  
Pointer.h> #include <decaf/util/Properties.h>
```

Data Structures

- class **activemq::transport::AbstractTransportFactory**
*Abstract implementation of the **TransportFactory** (p. 2798) interface, providing the base functionality that's common to most of the **TransportFactory** (p. 2798) instances.*

Namespaces

- namespace **activemq**
Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.
- namespace **activemq::transport**

7.132 src/main/activemq/transport/CompositeTransport.h File - Reference

```
#include <activemq/util/Config.h> #include <activemq/transport/-  
Transport.h> #include <decaf/net/URI.h> #include <decaf/util/-  
List.h>
```

Data Structures

- class **activemq::transport::CompositeTransport**

*A Composite **Transport** (p. 2790) is a **Transport** (p. 2790) implementation that is composed of several Transports.*

Namespaces

- namespace **activemq**

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

- namespace **activemq::transport**

7.133 src/main/activemq/transport/correlator/FutureResponse.h - File Reference

```
#include <activemq/util/Config.h> #include <decaf/lang/-
Pointer.h> #include <decaf/util/concurrent/Mutex.h> x
#include <decaf/util/concurrent/CountDownLatch.h> #include
<activemq/commands/Response.h> #include <activemq/exceptions/-
ActiveMQException.h>
```

Data Structures

- class **activemq::transport::correlator::FutureResponse**

A container that holds a response object.

Namespaces

- namespace **activemq**

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

- namespace **activemq::transport**
- namespace **activemq::transport::correlator**

7.134 src/main/activemq/transport/correlator/ResponseCorrelator.h File Reference

```
#include <activemq/util/Config.h> #include <activemq/transport/-
TransportFilter.h> #include <activemq/transport/correlator/-
FutureResponse.h> #include <activemq/commands/Command.-
h> #include <activemq/commands/Response.h> #include <decaf/util/concurrent/
```

7.135 src/main/activemq/transport/DefaultTransportListener.h File Reference 3031

```
Mutex.h> #include <decaf/util/concurrent/Concurrent.h> ×  
#include <decaf/util/concurrent/atomic/AtomicInteger.h>  
#include <map> #include <stdio.h>
```

Data Structures

- class **activemq::transport::correlator::ResponseCorrelator**

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

Namespaces

- namespace **activemq**

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

- namespace **activemq::transport**
- namespace **activemq::transport::correlator**

7.135 src/main/activemq/transport/DefaultTransportListener.h File Reference

```
#include <activemq/util/Config.h> #include <activemq/transport/-  
TransportListener.h> #include <activemq/commands/Command.-  
h> #include <decaf/lang/Pointer.h>
```

Data Structures

- class **activemq::transport::DefaultTransportListener**

*A Utility class that create empty implementations for the **TransportListener** (p. 2810) interface so that a subclass only needs to override the one's its interested.*

Namespaces

- namespace **activemq**

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

- namespace **activemq::transport**

7.136 src/main/activemq/transport/failover/BackupTransport.h File Reference

```
#include <activemq/util/Config.h> #include <activemq/transport/-  
Transport.h> #include <activemq/transport/DefaultTransport-  
Listener.h> #include <decaf/net/URI.h> #include <decaf/lang/-  
Pointer.h> #include <memory>
```

Data Structures

- class **activemq::transport::failover::BackupTransport**

Namespaces

- namespace **activemq**
Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.
- namespace **activemq::transport**
- namespace **activemq::transport::failover**

7.137 src/main/activemq/transport/failover/BackupTransportPool.h File Reference

```
#include <activemq/util/Config.h> #include <activemq/threads/-  
CompositeTask.h> #include <activemq/threads/Composite-  
TaskRunner.h> #include <activemq/transport/failover/-  
CloseTransportsTask.h> #include <activemq/transport/failover/-  
BackupTransport.h> #include <activemq/transport/failover/-  
URIPool.h> #include <decaf/lang/Pointer.h> #include <decaf/io/-  
IOException.h> #include <decaf/util/LinkedList.h>
```

Data Structures

- class **activemq::transport::failover::BackupTransportPool**

Namespaces

- namespace **activemq**
Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.
- namespace **activemq::transport**
- namespace **activemq::transport::failover**

7.138 src/main/activemq/transport/failover/CloseTransportsTask.h File Reference

7.138 src/main/activemq/transport/failover/CloseTransportsTask.h File Reference

```
#include <activemq/util/Config.h> #include <activemq/threads/-  
CompositeTask.h> #include <activemq/transport/Transport.-  
h> #include <decaf/util/LinkedList.h> #include <decaf/lang/-  
Pointer.h>
```

Data Structures

- class **activemq::transport::failover::CloseTransportsTask**

Namespaces

- namespace **activemq**
*Licensed to the Apache Software Foundation (ASF) under one or more contributor
license agreements.*
- namespace **activemq::transport**
- namespace **activemq::transport::failover**

7.139 src/main/activemq/transport/failover/FailoverTransport.h File Reference

```
#include <activemq/util/Config.h> #include <activemq/commands/-  
Command.h> #include <activemq/commands/ConnectionId.-  
h> #include <activemq/threads/TaskRunner.h> #include  
<activemq/threads/CompositeTaskRunner.h> #include <activemq/state/-  
ConnectionStateTracker.h> #include <activemq/transport/-  
CompositeTransport.h> #include <activemq/transport/failover/-  
BackupTransportPool.h> #include <activemq/transport/failover/-  
CloseTransportsTask.h> #include <activemq/transport/failover/-  
FailoverTransportListener.h> #include <activemq/transport/failover/-  
URIPool.h> #include <activemq/wireformat/WireFormat.h> ×  
#include <decaf/util/LinkedList.h> #include <decaf/util/-  
StlMap.h> #include <decaf/util/Properties.h> #include  
<decaf/util/concurrent/Mutex.h> #include <decaf/util/concurrent/atomic/-  
AtomicReference.h> #include <decaf/net/URI.h> #include  
<decaf/io/IOException.h>
```

Data Structures

- class **activemq::transport::failover::FailoverTransport**

Namespaces

- namespace **activemq**
Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.
- namespace **activemq::transport**
- namespace **activemq::transport::failover**

7.140 src/main/activemq/transport/failover/FailoverTransportFactory.h File Reference

```
#include <activemq/util/Config.h> #include <activemq/transport/-  
AbstractTransportFactory.h> #include <activemq/transport/-  
Transport.h> #include <activemq/exceptions/ActiveMQException.-  
h> #include <activemq/wireformat/WireFormat.h> #include  
<decaf/net/URI.h> #include <decaf/util/Properties.h>
```

Data Structures

- class **activemq::transport::failover::FailoverTransportFactory**
*Creates an instance of a **FailoverTransport** (p. 1305).*

Namespaces

- namespace **activemq**
Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.
- namespace **activemq::transport**
- namespace **activemq::transport::failover**

7.141 src/main/activemq/transport/failover/FailoverTransportListener.h File Reference

```
#include <activemq/util/Config.h> #include <activemq/transport/-  
TransportListener.h> #include <decaf/lang/Pointer.h>
```

Data Structures

- class **activemq::transport::failover::FailoverTransportListener**
*Utility class used by the **Transport** (p. 2790) to perform the work of responding to events from the active **Transport** (p. 2790).*

Namespaces

- namespace **activemq**
Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.
- namespace **activemq::transport**
- namespace **activemq::transport::failover**

7.142 src/main/activemq/transport/failover/URIPool.h File Reference

```
#include <activemq/util/Config.h> #include <decaf/net/URI-  
I.h> #include <decaf/util/LinkedList.h> #include <decaf/util/-  
NoSuchElementException.h>
```

Data Structures

- class **activemq::transport::failover::URIPool**

Namespaces

- namespace **activemq**
Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.
- namespace **activemq::transport**
- namespace **activemq::transport::failover**

7.143 src/main/activemq/transport/inactivity/InactivityMonitor.h - File Reference

```
#include <activemq/util/Config.h> #include <activemq/transport/-  
TransportFilter.h> #include <activemq/commands/Command.-  
h> #include <activemq/commands/Response.h> #include <activemq/commands/-  
WireFormatInfo.h> #include <activemq/wireformat/Wire-  
Format.h> #include <decaf/lang/Pointer.h> #include <decaf/util/-  
Timer.h> #include <decaf/util/Properties.h> #include  
<decaf/util/concurrent/atomic/AtomicBoolean.h>
```

Data Structures

- class **activemq::transport::inactivity::InactivityMonitor**

Namespaces

- namespace **activemq**

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

- namespace **activemq::transport**
- namespace **activemq::transport::inactivity**

7.144 src/main/activemq/transport/inactivity/ReadChecker.h File - Reference

```
#include <activemq/util/Config.h> #include <decaf/util/-  
TimerTask.h>
```

Data Structures

- class **activemq::transport::inactivity::ReadChecker**

Runnable class that is used by the {.

Namespaces

- namespace **activemq**

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

- namespace **activemq::transport**
- namespace **activemq::transport::inactivity**

7.145 src/main/activemq/transport/inactivity/WriteChecker.h File - Reference

```
#include <activemq/util/Config.h> #include <decaf/util/-  
TimerTask.h>
```

Data Structures

- class **activemq::transport::inactivity::WriteChecker**

Runnable class used by the {.

Namespaces

- namespace **activemq**
Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.
- namespace **activemq::transport**
- namespace **activemq::transport::inactivity**

7.146 src/main/activemq/transport/IOTransport.h File Reference

```
#include <activemq/util/Config.h> #include <activemq/transport/-
Transport.h> #include <activemq/transport/TransportListener.-
h> #include <activemq/commands/Command.h> #include <activemq/commands/-
Response.h> #include <activemq/exceptions/ActiveMQException.-
h> #include <activemq/wireformat/WireFormat.h> #include
<decaf/lang/Runnable.h> #include <decaf/lang/Thread.h>
#include <decaf/io/DataInputStream.h> #include <decaf/io/-
DataOutputStream.h> #include <decaf/util/logging/Logger-
Defines.h> #include <memory>
```

Data Structures

- class **activemq::transport::IOTransport**
*Implementation of the **Transport** (p. 2790) interface that performs marshaling of commands to IO streams.*

Namespaces

- namespace **activemq**
Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.
- namespace **activemq::transport**

7.147 src/main/activemq/transport/logging/LoggingTransport.h - File Reference

```
#include <activemq/util/Config.h> #include <activemq/transport/-
TransportFilter.h> #include <decaf/lang/Pointer.h>
```

Data Structures

- class **activemq::transport::logging::LoggingTransport**
A transport filter that logs commands as they are sent/received.

Namespaces

- namespace **activemq**
Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.
- namespace **activemq::transport**
- namespace **activemq::transport::logging**

7.148 src/main/activemq/transport/mock/InternalCommandListener.h File Reference

```
#include <activemq/util/Config.h> #include <activemq/transport/mock/-
ResponseBuilder.h> #include <activemq/transport/Default-
TransportListener.h> #include <decaf/lang/Thread.h> x
#include <decaf/lang/Pointer.h> #include <decaf/util/-
LinkedList.h> #include <decaf/util/concurrent/Concurrent.-
h> #include <decaf/util/concurrent/atomic/AtomicInteger.-
h> #include <decaf/util/concurrent/CountDownLatch.h>
```

Data Structures

- class **activemq::transport::mock::InternalCommandListener**
*Listens for Commands sent from the **MockTransport** (p. 1948).*

Namespaces

- namespace **activemq**
Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.
- namespace **activemq::transport**
- namespace **activemq::transport::mock**

7.149 src/main/activemq/transport/mock/MockTransport.h File - Reference

```
#include <activemq/util/Config.h> #include <activemq/exceptions/-
ActiveMQException.h> #include <activemq/transport/Transport.-
h> #include <activemq/transport/TransportListener.h> x
#include <activemq/transport/DefaultTransportListener.-
h> #include <activemq/transport/mock/ResponseBuilder.-
h> #include <activemq/transport/mock/InternalCommand-
Listener.h> #include <activemq/wireformat/WireFormat.-
h> #include <decaf/lang/Thread.h> #include <decaf/lang/-
```

7.150 src/main/activemq/transport/mock/MockTransportFactory.h File Reference

```
Pointer.h>    #include <decaf/util/concurrent/Concurrent.-  
h> #include <decaf/util/concurrent/atomic/AtomicInteger.-  
h>    #include <decaf/util/concurrent/CountDownLatch.h> ×  
#include <cms/Message.h> #include <map> #include <set>
```

Data Structures

- class **activemq::transport::mock::MockTransport**

*The **MockTransport** (p. 1948) defines a base level **Transport** (p. 2790) class that is intended to be used in place of an a regular protocol **Transport** (p. 2790) such as TCP.*

Namespaces

- namespace **activemq**

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

- namespace **activemq::transport**
- namespace **activemq::transport::mock**

7.150 src/main/activemq/transport/mock/MockTransportFactory.h - File Reference

```
#include <activemq/util/Config.h> #include <activemq/transport/-  
AbstractTransportFactory.h>
```

Data Structures

- class **activemq::transport::mock::MockTransportFactory**

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

Namespaces

- namespace **activemq**

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

- namespace **activemq::transport**
- namespace **activemq::transport::mock**

7.151 src/main/activemq/transport/mock/ResponseBuilder.h File - Reference

```
#include <activemq/util/Config.h> #include <activemq/commands/-  
Command.h> #include <activemq/commands/Response.h> #include  
<decaf/lang/Pointer.h> #include <decaf/util/LinkedList.-  
h>
```

Data Structures

- class **activemq::transport::mock::ResponseBuilder**

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

Namespaces

- namespace **activemq**

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

- namespace **activemq::transport**
- namespace **activemq::transport::mock**

7.152 src/main/activemq/transport/tcp/SslTransport.h File Reference

```
#include <activemq/util/Config.h> #include <activemq/transport/tcp/-  
TcpTransport.h>
```

Data Structures

- class **activemq::transport::tcp::SslTransport**

***Transport** (p. 2790) for connecting to a Broker using an SSL Socket.*

Namespaces

- namespace **activemq**

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

- namespace **activemq::transport**
- namespace **activemq::transport::tcp**

7.153 src/main/activemq/transport/tcp/SslTransportFactory.h File Reference 3041

7.153 src/main/activemq/transport/tcp/SslTransportFactory.h File - Reference

```
#include <activemq/util/Config.h> #include <activemq/transport/tcp/-  
TcpTransportFactory.h>
```

Data Structures

- class **activemq::transport::tcp::SslTransportFactory**

Namespaces

- namespace **activemq**
Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.
- namespace **activemq::transport**
- namespace **activemq::transport::tcp**

7.154 src/main/activemq/transport/tcp/TcpTransport.h File - Reference

```
#include <activemq/io/LoggingInputStream.h> #include <activemq/io/-  
LoggingOutputStream.h> #include <activemq/util/Config.-  
h> #include <activemq/transport/TransportFilter.h> #include  
<decaf/net/Socket.h> #include <decaf/net/URI.h> #include  
<decaf/util/Properties.h> #include <decaf/lang/Pointer.-  
h> #include <decaf/io/BufferedInputStream.h> #include  
<decaf/io/BufferedOutputStream.h> #include <decaf/io/-  
DataInputStream.h> #include <decaf/io/DataOutputStream.-  
h> #include <memory>
```

Data Structures

- class **activemq::transport::tcp::TcpTransport**
*Implements a TCP/IP based transport filter, this transport is meant to wrap an instance of an **IOTransport** (p. 1548).*

Namespaces

- namespace **activemq**
Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.
- namespace **activemq::transport**

- namespace **activemq::transport::tcp**

7.155 src/main/activemq/transport/tcp/TcpTransportFactory.h File - Reference

```
#include <activemq/util/Config.h> #include <activemq/transport/-
AbstractTransportFactory.h> #include <activemq/exceptions/-
ActiveMQException.h>
```

Data Structures

- class **activemq::transport::tcp::TcpTransportFactory**
*Factory Responsible for creating the **TcpTransport** (p. 2693).*

Namespaces

- namespace **activemq**
Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.
- namespace **activemq::transport**
- namespace **activemq::transport::tcp**

7.156 src/main/activemq/transport/Transport.h File Reference

```
#include <decaf/io/InputStream.h>      #include <decaf/io/-
OutputStream.h> #include <decaf/io/IOException.h> #include
<decaf/io/Closeable.h>      #include <decaf/util/List.h> ×
#include <decaf/net/URI.h> #include <decaf/lang/Pointer.-
h> #include <decaf/lang/exceptions/UnsupportedOperation-
Exception.h> #include <activemq/util/Config.h> #include
<activemq/commands/Command.h> #include <activemq/commands/-
Response.h> #include <typeinfo>
```

Data Structures

- class **activemq::transport::Transport**
Interface for a transport layer for command objects.

Namespaces

- namespace **activemq**

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

- namespace **activemq::wireformat**
- namespace **activemq::transport**

7.157 src/main/activemq/transport/TransportFactory.h File Reference

```
#include <activemq/util/Config.h> #include <activemq/exceptions/-  
ActiveMQException.h> #include <activemq/transport/Transport.-  
h> #include <decaf/net/URI.h> #include <decaf/util/-  
Properties.h> #include <decaf/lang/Pointer.h>
```

Data Structures

- class **activemq::transport::TransportFactory**
Defines the interface for Factories that create Transports or TransportFilters.

Namespaces

- namespace **activemq**
Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.
- namespace **activemq::transport**

7.158 src/main/activemq/transport/TransportFilter.h File Reference

```
#include <activemq/util/Config.h> #include <activemq/exceptions/-  
ActiveMQException.h> #include <activemq/transport/Transport.-  
h> #include <activemq/commands/Command.h> #include <activemq/transport/-  
TransportListener.h> #include <decaf/lang/Pointer.h> x  
#include <typeinfo>
```

Data Structures

- class **activemq::transport::TransportFilter**
A filter on the transport layer.

Namespaces

- namespace **activemq**
Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.
- namespace **activemq::transport**

7.159 src/main/activemq/transport/TransportListener.h File - Reference

```
#include <activemq/util/Config.h> #include <activemq/commands/-  
Command.h> #include <decaf/lang/Exception.h> #include  
<decaf/lang/Pointer.h>
```

Data Structures

- class **activemq::transport::TransportListener**
A listener of asynchronous exceptions from a command transport object.

Namespaces

- namespace **activemq**
Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.
- namespace **activemq::transport**

7.160 src/main/activemq/transport/TransportRegistry.h File - Reference

```
#include <activemq/util/Config.h> #include <string> ×  
#include <vector> #include <activemq/transport/Transport-  
Factory.h> #include <decaf/util/STLMap.h> #include <decaf/util/-  
NoSuchElementException.h> #include <decaf/lang/exceptions/-  
NullPointerException.h> #include <decaf/lang/exceptions/-  
IllegalArgumentException.h>
```

Data Structures

- class **activemq::transport::TransportRegistry**
*Registry of all **Transport** (p. 2790) Factories that are available to the client at runtime.*

Namespaces

- namespace **activemq**
Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.
- namespace **activemq::transport**

7.161 src/main/activemq/util/ActiveMQProperties.h File Reference

```
#include <map>      #include <string>      #include <sstream> ×  
#include <activemq/util/Config.h>          #include <cms/CMS-  
Properties.h> #include <decaf/util/Properties.h>
```

Data Structures

- class **activemq::util::ActiveMQProperties**

*Implementation of the CMSProperties interface that delegates to a **decaf::util::Properties** (p. 2200) object.*

Namespaces

- namespace **activemq**

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

- namespace **activemq::util**

7.162 src/main/activemq/util/CMSExceptionSupport.h File Reference

```
#include <activemq/util/Config.h>          #include <cms/CMS-  
Exception.h> #include <cms/CMSSecurityException.h> #include  
<cms/MessageEOFException.h> #include <cms/MessageFormat-  
Exception.h> #include <cms/MessageNotReadableException.-  
h> #include <cms/MessageNotWriteableException.h> #include  
<cms/InvalidClientIdException.h> #include <cms/Invalid-  
DestinationException.h> #include <cms/InvalidSelector-  
Exception.h> #include <cms/IllegalStateException.h> ×  
#include <cms/UnsupportedOperationException.h> #include  
<decaf/lang/Exception.h> #include <string>
```

Data Structures

- class **activemq::util::CMSExceptionSupport**

Namespaces

- namespace **activemq**

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

- namespace **activemq::util**

Defines

- **#define AMQ_CATCH_ALL_THROW_CMSEXCEPTION()**

Macro for catching an exception of one type and then re-throwing as a Basic CMS-Exception, good for cases where the method isn't specific about what CMS Exceptions are thrown, bad if you need to throw an exception of MessageNotReadableException for instance.

7.162.1 Define Documentation

7.162.1.1 #define AMQ_CATCH_ALL_THROW_CMSEXCEPTION()

Macro for catching an exception of one type and then re-throwing as a Basic CMS-Exception, good for cases where the method isn't specific about what CMS Exceptions are thrown, bad if you need to throw an exception of MessageNotReadableException for instance.

Referenced by `activemq::commands::ActiveMQMessageTemplate< cms::ObjectMessage >::acknowledge()`, `activemq::commands::ActiveMQMessageTemplate< cms::ObjectMessage >::clearBody()`, `activemq::commands::ActiveMQMessageTemplate< cms::ObjectMessage >::clearProperties()`, `activemq::commands::ActiveMQMessageTemplate< cms::ObjectMessage >::equals()`, `activemq::commands::ActiveMQMessageTemplate< cms::ObjectMessage >::getBooleanProperty()`, `activemq::commands::ActiveMQMessageTemplate< cms::ObjectMessage >::getByteProperty()`, `activemq::commands::ActiveMQMessageTemplate< cms::ObjectMessage >::getCMSMessageID()`, `activemq::commands::ActiveMQMessageTemplate< cms::ObjectMessage >::getDoubleProperty()`, `activemq::commands::ActiveMQMessageTemplate< cms::ObjectMessage >::getFloatProperty()`, `activemq::commands::ActiveMQMessageTemplate< cms::ObjectMessage >::getIntProperty()`, `activemq::commands::ActiveMQMessageTemplate< cms::ObjectMessage >::getLongProperty()`, `activemq::commands::ActiveMQMessageTemplate< cms::ObjectMessage >::getPropertyNames()`, `activemq::commands::ActiveMQMessageTemplate< cms::ObjectMessage >::getShortProperty()`, `activemq::commands::ActiveMQMessageTemplate< cms::ObjectMessage >::getStringProperty()`, `activemq::commands::ActiveMQMessageTemplate< cms::ObjectMessage >::propertyExists()`, `activemq::commands::ActiveMQMessageTemplate< cms::ObjectMessage >::setBooleanProperty()`, `activemq::commands::ActiveMQMessageTemplate< cms::ObjectMessage >::setByteProperty()`, `activemq::commands::ActiveMQMessageTemplate< cms::ObjectMessage >::setCMSDestination()`, `activemq::commands::ActiveMQMessageTemplate< cms::ObjectMessage >::setCMSReplyTo()`, `activemq::commands::ActiveMQMessageTemplate< cms::ObjectMessage >::setDoubleProperty()`, `activemq::commands::ActiveMQMessageTemplate< cms::ObjectMessage >::setFloatProperty()`, `activemq::commands::ActiveMQMessageTemplate< cms::ObjectMessage >::setIntProperty()`, `activemq::commands::ActiveMQMessageTemplate< cms::ObjectMessage >::setLongProperty()`, `activemq::commands::ActiveMQMessageTemplate< cms::ObjectMessage >::setShortProperty()`, and `activemq::commands::ActiveMQMessageTemplate< cms::ObjectMessage >::setStringProperty()`.

7.163 src/main/activemq/util/CompositeData.h File Reference

```
#include <activemq/util/Config.h> #include <decaf/util/-  
Properties.h> #include <decaf/util/LinkedList.h> #include  
<decaf/net/URI.h> #include <decaf/net/URISyntaxException.-  
h>
```

Data Structures

- class **activemq::util::CompositeData**

Represents a Composite URI.

Namespaces

- namespace **activemq**
*Licensed to the Apache Software Foundation (ASF) under one or more contributor
license agreements.*
- namespace **activemq::util**

7.164 src/main/activemq/util/Config.h File Reference

Defines

- #define **AMQCPP_API**
- #define **HAVE_UUID_UUID_H**
- #define **HAVE_UUID_T**
- #define **HAVE_PTHREAD_H**

7.164.1 Define Documentation

7.164.1.1 #define **AMQCPP_API**

7.164.1.2 #define **HAVE_PTHREAD_H**

7.164.1.3 #define **HAVE_UUID_T**

7.164.1.4 #define **HAVE_UUID_UUID_H**

7.165 src/main/cms/Config.h File Reference

Defines

- #define **CMS_API**

7.165.1 Define Documentation

7.165.1.1 `#define CMS_API`

7.166 `src/main/decaf/util/Config.h` File Reference

Defines

- `#define DECAF_API`
- `#define HAVE_UUID_UUID_H`
- `#define HAVE_UUID_T`
- `#define HAVE_PTHREAD_H`
- `#define DECAF_UNUSED`

7.166.1 Define Documentation

7.166.1.1 `#define DECAF_API`

7.166.1.2 `#define DECAF_UNUSED`

7.166.1.3 `#define HAVE_PTHREAD_H`

7.166.1.4 `#define HAVE_UUID_T`

7.166.1.5 `#define HAVE_UUID_UUID_H`

7.167 `src/main/activemq/util/IdGenerator.h` File Reference

```
#include <activemq/util/Config.h> #include <string>
```

Data Structures

- class `activemq::util::IdGenerator`

Namespaces

- namespace `activemq`
Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.
- namespace `activemq::library`
- namespace `activemq::util`

7.168 src/main/activemq/util/LongSequenceGenerator.h File Reference

```
#include <activemq/util/Config.h> #include <decaf/util/concurrent/-  
Mutex.h>
```

Data Structures

- class **activemq::util::LongSequenceGenerator**
This class is used to generate a sequence of long long values that are incremented each time a new value is requested.

Namespaces

- namespace **activemq**
Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.
- namespace **activemq::util**

7.169 src/main/activemq/util/MarshallingSupport.h File Reference

```
#include <activemq/util/Config.h> #include <decaf/io/I-  
OException.h> #include <decaf/io/UTFDataFormatException.-  
h> #include <decaf/io/DataOutputStream.h> #include <decaf/io/-  
DataInputStream.h> #include <string>
```

Data Structures

- class **activemq::util::MarshallingSupport**

Namespaces

- namespace **activemq**
Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.
- namespace **activemq::util**

7.170 src/main/activemq/util/MemoryUsage.h File Reference

```
#include <activemq/util/Config.h> #include <activemq/util/-  
Usage.h> #include <decaf/util/concurrent/Mutex.h>
```

Data Structures

- class **activemq::util::MemoryUsage**

Namespaces

- namespace **activemq**

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

- namespace **activemq::util**

7.171 src/main/activemq/util/PrimitiveList.h File Reference

```
#include <string> #include <vector> #include <decaf/util/-  
LinkedList.h> #include <decaf/lang/exceptions/Unsupported-  
OperationException.h> #include <decaf/lang/exceptions/-  
IndexOutOfBoundsException.h> #include <stdio.h> #include  
<activemq/util/PrimitiveValueNode.h> #include <activemq/util/-  
PrimitiveValueConverter.h>
```

Data Structures

- class **activemq::util::PrimitiveList**

List of primitives.

Namespaces

- namespace **activemq**

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

- namespace **activemq::util**

7.172 src/main/activemq/util/PrimitiveMap.h File Reference

```
#include <string> #include <vector> #include <activemq/util/-  
Config.h> #include <decaf/util/Config.h> #include <decaf/util/-  
StlMap.h> #include <decaf/util/NoSuchElementException.-  
h> #include <activemq/util/PrimitiveValueNode.h> #include  
<activemq/util/PrimitiveValueConverter.h>
```


Data Structures

- class **activemq::util::PrimitiveMap**

Map of named primitives.

Namespaces

- namespace **activemq**
Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.
- namespace **activemq::util**

7.173 src/main/activemq/util/PrimitiveValueConverter.h File Reference

```
#include <activemq/util/Config.h> #include <activemq/util/-  
PrimitiveValueNode.h> #include <decaf/lang/exceptions/-  
UnsupportedOperationException.h> #include <string>
```

Data Structures

- class **activemq::util::PrimitiveValueConverter**

*Class controls the conversion of data contained in a **PrimitiveValueNode** (p. 2145) from one type to another.*

Namespaces

- namespace **activemq**
Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.
- namespace **activemq::util**

7.174 src/main/activemq/util/PrimitiveValueNode.h File Reference

```
#include <activemq/util/Config.h> #include <decaf/util/-  
NoSuchElementException.h> #include <decaf/util/Map.h> ×  
#include <decaf/util/List.h>
```

Data Structures

- class **activemq::util::PrimitiveValueNode**

Class that wraps around a single value of one of the many types.

- union **activemq::util::PrimitiveValueNode::PrimitiveValue**

Define a union type comprised of the various types.

Namespaces

- namespace **activemq**

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

- namespace **activemq::util**

7.175 src/main/activemq/util/Service.h File Reference

```
#include <activemq/util/Config.h>
```

Data Structures

- class **activemq::util::Service**

*Base interface for all classes that run as a **Service** (p. 2355) inside the application.*

Namespaces

- namespace **activemq**

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

- namespace **activemq::util**

7.176 src/main/activemq/util/ServiceListener.h File Reference

```
#include <activemq/util/Config.h>
```

Data Structures

- class **activemq::util::ServiceListener**

*Listener interface for observers of **Service** (p. 2355) related events.*

Namespaces

- namespace **activemq**

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

- namespace **activemq::util**

7.177 src/main/activemq/util/ServiceStopper.h File Reference

```
#include <activemq/util/Config.h> #include <decaf/lang/-  
Exception.h>
```

Data Structures

- class **activemq::util::ServiceStopper**

Namespaces

- namespace **activemq**
Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.
- namespace **activemq::util**

7.178 src/main/activemq/util/ServiceSupport.h File Reference

```
#include <activemq/util/Config.h> #include <activemq/util/-  
Service.h> #include <decaf/util/concurrent/atomic/Atomic-  
Boolean.h> #include <decaf/util/concurrent/CopyOnWrite-  
ArrayList.h>
```

Data Structures

- class **activemq::util::ServiceSupport**
*Provides a base class for **Service** (p. 2355) implementations.*

Namespaces

- namespace **activemq**
Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.
- namespace **activemq::util**

Defines

- #define **SERVICESUPPORT_H_**

7.178.1 Define Documentation

7.178.1.1 `#define SERVICESUPPORT_H_`

7.179 `src/main/activemq/util/URISupport.h` File Reference

```
#include <activemq/util/Config.h> #include <activemq/util/-
CompositeData.h> #include <decaf/util/Properties.h> ×
#include <decaf/util/LinkedList.h> #include <decaf/lang/exceptions/-
IllegalArgumentException.h>
```

Data Structures

- class **activemq::util::URISupport**

Namespaces

- namespace **activemq**
Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.
- namespace **activemq::util**

7.180 `src/main/activemq/util/Usage.h` File Reference

```
#include <activemq/util/Config.h>
```

Data Structures

- class **activemq::util::Usage**

Namespaces

- namespace **activemq**
Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.
- namespace **activemq::util**

7.181 `src/main/activemq/wireformat/MarshalAware.h` File Reference

```
#include <vector> #include <decaf/io/IOException.h> ×
#include <activemq/util/Config.h>
```

7.182

src/main/activemq/wireformat/openwire/marshal/BaseDataStreamMarshaller.h

File Reference

3055

Data Structures

- class **activemq::wireformat::MarshalAware**

Namespaces

- namespace **activemq**
Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.
- namespace **activemq::wireformat**

7.182 src/main/activemq/wireformat/openwire/marshal/BaseDataStreamMarshaller.h File Reference

```
#include <activemq/wireformat/openwire/marshal/DataStream-  
Marshaller.h> #include <activemq/wireformat/openwire/Utils/-  
HexTable.h> #include <activemq/commands/MessageId.h>  
#include <activemq/commands/ProducerId.h> #include <activemq/commands/-  
TransactionId.h> #include <activemq/util/Config.h>
```

Data Structures

- class **activemq::wireformat::openwire::marshal::BaseDataStreamMarshaller**

Base class for all Marshallers that marshal DataStructures to and from the wire using the OpenWire protocol.

Namespaces

- namespace **activemq**
Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.
- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**

7.183 src/main/activemq/wireformat/openwire/marshal/DataStreamMarshaller.h File Reference

```
#include <decaf/io/DataInputStream.h> #include <decaf/io/-  
DataOutputStream.h> #include <decaf/io/IOException.h> ×  
#include <activemq/commands/DataStructure.h> #include
```

```
<activemq/wireformat/openwire/utils/BooleanStream.h> ×
#include <activemq/wireformat/openwire/OpenWireFormat.-
h> #include <activemq/util/Config.h>
```

Data Structures

- class **activemq::wireformat::openwire::marshal::DataStreamMarshaller**
Base class for all classes that marshal commands for Openwire.

Namespaces

- namespace **activemq**
Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.
- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**

7.184 src/main/activemq/wireformat/openwire/marshal/generated/-ActiveMQBlobMessageMarshaller.h File Reference

```
#include <activemq/wireformat/openwire/marshal/generated/-
MessageMarshaller.h> #include <decaf/io/DataInputStream.-
h> #include <decaf/io/DataOutputStream.h> #include <decaf/io/-
IOException.h> #include <activemq/util/Config.h> #include
<activemq/commands/DataStructure.h> #include <activemq/wireformat/openwire
OpenWireFormat.h> #include <activemq/wireformat/openwire/utils/-
BooleanStream.h>
```

Data Structures

- class **activemq::wireformat::openwire::marshal::generated::ActiveMQBlobMessageMarshaller**
*Marshaling code for Open Wire Format for **ActiveMQBlobMessageMarshaller** (p. 161).*

Namespaces

- namespace **activemq**
Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.
- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**

7.185 src/main/activemq/wireformat/openwire/marshal/generated/ActiveMQ-BytesMessageMarshaller.h File

Reference

3057

- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::generated**

7.185 src/main/activemq/wireformat/openwire/marshal/generated/-ActiveMQBytesMessageMarshaller.h File Reference

```
#include <activemq/wireformat/openwire/marshal/generated/-  
MessageMarshaller.h> #include <decaf/io/DataInputStream.-  
h> #include <decaf/io/DataOutputStream.h> #include <decaf/io/-  
IOException.h> #include <activemq/util/Config.h> #include  
<activemq/commands/DataStructure.h> #include <activemq/wireformat/openwire/-  
OpenWireFormat.h> #include <activemq/wireformat/openwire/utis/-  
BooleanStream.h>
```

Data Structures

- class **activemq::wireformat::openwire::marshal::generated::ActiveMQ-BytesMessageMarshaller**

*Marshaling code for Open Wire Format for **ActiveMQBytesMessageMarshaller** (p. 183).*

Namespaces

- namespace **activemq**
Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.
- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::generated**

7.186 src/main/activemq/wireformat/openwire/marshal/generated/-ActiveMQDestinationMarshaller.h File Reference

```
#include <activemq/wireformat/openwire/marshal/BaseData-  
StreamMarshaller.h> #include <decaf/io/DataInputStream.-  
h> #include <decaf/io/DataOutputStream.h> #include <decaf/io/-  
IOException.h> #include <activemq/util/Config.h> #include  
<activemq/commands/DataStructure.h> #include <activemq/wireformat/openwire/-  
OpenWireFormat.h> #include <activemq/wireformat/openwire/utis/-  
BooleanStream.h>
```

Data Structures

- class **activemq::wireformat::openwire::marshal::generated::ActiveMQDestinationMarshaller**

*Marshaling code for Open Wire Format for **ActiveMQDestinationMarshaller** (p. 258).*

Namespaces

- namespace **activemq**
Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.
- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::generated**

7.187 src/main/activemq/wireformat/openwire/marshal/generated/-ActiveMQMapMessageMarshaller.h File Reference

```
#include <activemq/wireformat/openwire/marshal/generated/-
MessageMarshaller.h> #include <decaf/io/DataInputStream.-
h> #include <decaf/io/DataOutputStream.h> #include <decaf/io/-
IOException.h> #include <activemq/util/Config.h> #include
<activemq/commands/DataStructure.h> #include <activemq/wireformat/openwire
OpenWireFormat.h> #include <activemq/wireformat/openwire/utis/-
BooleanStream.h>
```

Data Structures

- class **activemq::wireformat::openwire::marshal::generated::ActiveMQMapMessageMarshaller**

*Marshaling code for Open Wire Format for **ActiveMQMapMessageMarshaller** (p. 284).*

Namespaces

- namespace **activemq**
Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.
- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::generated**

7.188 src/main/activemq/wireformat/openwire/marshal/generated/ActiveMQ-MessageMarshaller.h File

Reference

3059

7.188 src/main/activemq/wireformat/openwire/marshal/generated/ActiveMQMessageMarshaller.h File Reference

```
#include <activemq/wireformat/openwire/marshal/generated/-  
MessageMarshaller.h> #include <decaf/io/DataInputStream.-  
h> #include <decaf/io/DataOutputStream.h> #include <decaf/io/-  
IOException.h> #include <activemq/util/Config.h> #include  
<activemq/commands/DataStructure.h> #include <activemq/wireformat/openwire/-  
OpenWireFormat.h> #include <activemq/wireformat/openwire/utis/-  
BooleanStream.h>
```

Data Structures

- class **activemq::wireformat::openwire::marshal::generated::ActiveMQ-MessageMarshaller**

*Marshaling code for Open Wire Format for **ActiveMQMessageMarshaller** (p. 291).*

Namespaces

- namespace **activemq**
Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.
- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::generated**

7.189 src/main/activemq/wireformat/openwire/marshal/generated/ActiveMQObjectMessageMarshaller.h File Reference

```
#include <activemq/wireformat/openwire/marshal/generated/-  
MessageMarshaller.h> #include <decaf/io/DataInputStream.-  
h> #include <decaf/io/DataOutputStream.h> #include <decaf/io/-  
IOException.h> #include <activemq/util/Config.h> #include  
<activemq/commands/DataStructure.h> #include <activemq/wireformat/openwire/-  
OpenWireFormat.h> #include <activemq/wireformat/openwire/utis/-  
BooleanStream.h>
```

Data Structures

- class **activemq::wireformat::openwire::marshal::generated::ActiveMQ-ObjectMessageMarshaller**

*Marshaling code for Open Wire Format for **ActiveMQObjectMessageMarshaller** (p. 304).*

Namespaces

- namespace **activemq**
Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.
- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::generated**

7.190 src/main/activemq/wireformat/openwire/marshal/generated/-ActiveMQQueueMarshaller.h File Reference

```
#include <activemq/wireformat/openwire/marshal/generated/-
ActiveMQDestinationMarshaller.h> #include <decaf/io/Data-
InputStream.h> #include <decaf/io/DataOutputStream.h>
#include <decaf/io/IOException.h> #include <activemq/util/-
Config.h> #include <activemq/commands/DataStructure.h> ×
#include <activemq/wireformat/openwire/OpenWireFormat.-
h> #include <activemq/wireformat/openwire/Utils/Boolean-
Stream.h>
```

Data Structures

- class **activemq::wireformat::openwire::marshal::generated::ActiveMQ-QueueMarshaller**
*Marshaling code for Open Wire Format for **ActiveMQQueueMarshaller** (p. 329).*

Namespaces

- namespace **activemq**
Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.
- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::generated**

7.191 src/main/activemq/wireformat/openwire/marshal/generated/-ActiveMQStreamMessageMarshaller.h File Reference

```
#include <activemq/wireformat/openwire/marshal/generated/-
MessageMarshaller.h> #include <decaf/io/DataInputStream.-
```

7.192 src/main/activemq/wireformat/openwire/marshal/generated/ActiveMQTempDestinationMarshaller.h File

Reference

3061

```
h> #include <decaf/io/DataOutputStream.h> #include <decaf/io/-
IOException.h> #include <activemq/util/Config.h> #include
<activemq/commands/DataStructure.h> #include <activemq/wireformat/openwire/-
OpenWireFormat.h> #include <activemq/wireformat/openwire/utis/-
BooleanStream.h>
```

Data Structures

- class **activemq::wireformat::openwire::marshal::generated::ActiveMQStreamMessageMarshaller**

*Marshaling code for Open Wire Format for **ActiveMQStreamMessageMarshaller** (p. 375).*

Namespaces

- namespace **activemq**
Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.
- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::generated**

7.192 src/main/activemq/wireformat/openwire/marshal/generated/ActiveMQTempDestinationMarshaller.h File Reference

```
#include <activemq/wireformat/openwire/marshal/generated/-
ActiveMQDestinationMarshaller.h> #include <decaf/io/Data-
InputStream.h> #include <decaf/io/DataOutputStream.h>
#include <decaf/io/IOException.h> #include <activemq/util/-
Config.h> #include <activemq/commands/DataStructure.h> ×
#include <activemq/wireformat/openwire/OpenWireFormat.-
h> #include <activemq/wireformat/openwire/utis/Boolean-
Stream.h>
```

Data Structures

- class **activemq::wireformat::openwire::marshal::generated::ActiveMQTempDestinationMarshaller**

*Marshaling code for Open Wire Format for **ActiveMQTempDestinationMarshaller** (p. 382).*

Namespaces

- namespace **activemq**
Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.
- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::generated**

7.193 src/main/activemq/wireformat/openwire/marshal/generated/-ActiveMQTempQueueMarshaller.h File Reference

```
#include <activemq/wireformat/openwire/marshal/generated/-
ActiveMQTempDestinationMarshaller.h> #include <decaf/io/-
DataInputStream.h> #include <decaf/io/DataOutputStream.-
h> #include <decaf/io/IOException.h> #include <activemq/util/-
Config.h> #include <activemq/commands/DataStructure.h> ×
#include <activemq/wireformat/openwire/OpenWireFormat.-
h> #include <activemq/wireformat/openwire/Utils/Boolean-
Stream.h>
```

Data Structures

- class **activemq::wireformat::openwire::marshal::generated::ActiveMQTempQueueMarshaller**
*Marshaling code for Open Wire Format for **ActiveMQTempQueueMarshaller** (p. 391).*

Namespaces

- namespace **activemq**
Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.
- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::generated**

7.194 src/main/activemq/wireformat/openwire/marshal/generated/-ActiveMQTempTopicMarshaller.h File Reference

```
#include <activemq/wireformat/openwire/marshal/generated/-
ActiveMQTempDestinationMarshaller.h> #include <decaf/io/-
```

7.195 src/main/activemq/wireformat/openwire/marshal/generated/ActiveMQTextMessageMarshaller.h File

Reference

3063

```
DataInputStream.h> #include <decaf/io/DataOutputStream.h>
h> #include <decaf/io/IOException.h> #include <activemq/util/Config.h>
#include <activemq/commands/DataStructure.h> ×
#include <activemq/wireformat/openwire/OpenWireFormat.h>
h> #include <activemq/wireformat/openwire/utils/BooleanStream.h>
```

Data Structures

- class **activemq::wireformat::openwire::marshal::generated::ActiveMQTempTopicMarshaller**

*Marshaling code for Open Wire Format for **ActiveMQTempTopicMarshaller** (p. 401).*

Namespaces

- namespace **activemq**

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

- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::generated**

7.195 src/main/activemq/wireformat/openwire/marshal/generated/ActiveMQTextMessageMarshaller.h File Reference

```
#include <activemq/wireformat/openwire/marshal/generated/MessageMarshaller.h>
#include <decaf/io/DataInputStream.h> #include <decaf/io/DataOutputStream.h>
#include <decaf/io/IOException.h> #include <activemq/util/Config.h> #include
<activemq/commands/DataStructure.h> #include <activemq/wireformat/openwire/OpenWireFormat.h>
#include <activemq/wireformat/openwire/utils/BooleanStream.h>
```

Data Structures

- class **activemq::wireformat::openwire::marshal::generated::ActiveMQTextMessageMarshaller**

*Marshaling code for Open Wire Format for **ActiveMQTextMessageMarshaller** (p. 410).*

Namespaces

- namespace **activemq**
Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.
- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::generated**

7.196 src/main/activemq/wireformat/openwire/marshal/generated/-ActiveMQTopicMarshaller.h File Reference

```
#include <activemq/wireformat/openwire/marshal/generated/-
ActiveMQDestinationMarshaller.h> #include <decaf/io/Data-
InputStream.h> #include <decaf/io/DataOutputStream.h>
#include <decaf/io/IOException.h> #include <activemq/util/-
Config.h> #include <activemq/commands/DataStructure.h> x
#include <activemq/wireformat/openwire/OpenWireFormat.-
h> #include <activemq/wireformat/openwire/Utils/Boolean-
Stream.h>
```

Data Structures

- class **activemq::wireformat::openwire::marshal::generated::ActiveMQ-TopicMarshaller**
*Marshaling code for Open Wire Format for **ActiveMQTopicMarshaller** (p. 419).*

Namespaces

- namespace **activemq**
Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.
- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::generated**

7.197 src/main/activemq/wireformat/openwire/marshal/generated/-BaseCommandMarshaller.h File Reference

```
#include <activemq/wireformat/openwire/marshal/BaseData-
StreamMarshaller.h> #include <decaf/io/DataInputStream.-
```

7.198

src/main/activemq/wireformat/openwire/marshal/generated/BrokerIdMarshaller.h

File Reference

3065

```
#include <decaf/io/DataOutputStream.h> #include <decaf/io/-
IOException.h> #include <activemq/util/Config.h> #include
<activemq/commands/DataStructure.h> #include <activemq/wireformat/openwire/-
OpenWireFormat.h> #include <activemq/wireformat/openwire/utis/-
BooleanStream.h>
```

Data Structures

- class **activemq::wireformat::openwire::marshal::generated::BaseCommandMarshaller**

*Marshaling code for Open Wire Format for **BaseCommandMarshaller** (p. 499).*

Namespaces

- namespace **activemq**
Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.
- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::generated**

7.198 src/main/activemq/wireformat/openwire/marshal/generated/- BrokerIdMarshaller.h File Reference

```
#include <activemq/wireformat/openwire/marshal/BaseData-
StreamMarshaller.h> #include <decaf/io/DataInputStream.-
h> #include <decaf/io/DataOutputStream.h> #include <decaf/io/-
IOException.h> #include <activemq/util/Config.h> #include
<activemq/commands/DataStructure.h> #include <activemq/wireformat/openwire/-
OpenWireFormat.h> #include <activemq/wireformat/openwire/utis/-
BooleanStream.h>
```

Data Structures

- class **activemq::wireformat::openwire::marshal::generated::BrokerIdMarshaller**

*Marshaling code for Open Wire Format for **BrokerIdMarshaller** (p. 567).*

Namespaces

- namespace **activemq**

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

- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::generated**

7.199 src/main/activemq/wireformat/openwire/marshal/generated/-BrokerInfoMarshaller.h File Reference

```
#include <activemq/wireformat/openwire/marshal/generated/-
BaseCommandMarshaller.h> #include <decaf/io/DataInput-
Stream.h> #include <decaf/io/DataOutputStream.h> #include
<decaf/io/IOException.h> #include <activemq/util/Config.-
h> #include <activemq/commands/DataStructure.h> #include
<activemq/wireformat/openwire/OpenWireFormat.h> #include
<activemq/wireformat/openwire/utils/BooleanStream.h>
```

Data Structures

- class **activemq::wireformat::openwire::marshal::generated::BrokerInfoMarshaller**

*Marshaling code for Open Wire Format for **BrokerInfoMarshaller** (p. 578).*

Namespaces

- namespace **activemq**
Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.
- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::generated**

7.200 src/main/activemq/wireformat/openwire/marshal/generated/-ConnectionControlMarshaller.h File Reference

```
#include <activemq/wireformat/openwire/marshal/generated/-
BaseCommandMarshaller.h> #include <decaf/io/DataInput-
Stream.h> #include <decaf/io/DataOutputStream.h> #include
<decaf/io/IOException.h> #include <activemq/util/Config.-
h> #include <activemq/commands/DataStructure.h> #include
<activemq/wireformat/openwire/OpenWireFormat.h> #include
<activemq/wireformat/openwire/utils/BooleanStream.h>
```


7.201 src/main/activemq/wireformat/openwire/marshal/generated/ConnectionErrorMarshaller.h File

Reference Data Structures

3067

- class **activemq::wireformat::openwire::marshal::generated::ConnectionControlMarshaller**

*Marshaling code for Open Wire Format for **ConnectionControlMarshaller** (p. 943).*

Namespaces

- namespace **activemq**
Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.
- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::generated**

7.201 src/main/activemq/wireformat/openwire/marshal/generated/ConnectionErrorMarshaller.h File Reference

```
#include <activemq/wireformat/openwire/marshal/generated/BaseCommandMarshaller.h> #include <decaf/io/DataInputStream.h> #include <decaf/io/DataOutputStream.h> #include <decaf/io/IOException.h> #include <activemq/util/Config.h> #include <activemq/commands/DataStructure.h> #include <activemq/wireformat/openwire/OpenWireFormat.h> #include <activemq/wireformat/openwire/Utils/BooleanStream.h>
```

Data Structures

- class **activemq::wireformat::openwire::marshal::generated::ConnectionErrorMarshaller**

*Marshaling code for Open Wire Format for **ConnectionErrorMarshaller** (p. 951).*

Namespaces

- namespace **activemq**
Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.
- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::generated**

7.202 src/main/activemq/wireformat/openwire/marshal/generated/- ConnectionIdMarshaller.h File Reference

```
#include <activemq/wireformat/openwire/marshal/BaseData-  
StreamMarshaller.h> #include <decaf/io/DataInputStream.-  
h> #include <decaf/io/DataOutputStream.h> #include <decaf/io/-  
IOException.h> #include <activemq/util/Config.h> #include  
<activemq/commands/DataStructure.h> #include <activemq/wireformat/openwire/  
OpenWireFormat.h> #include <activemq/wireformat/openwire/utils/-  
BooleanStream.h>
```

Data Structures

- class **activemq::wireformat::openwire::marshal::generated::ConnectionIdMarshaller**

*Marshaling code for Open Wire Format for **ConnectionIdMarshaller** (p. 963).*

Namespaces

- namespace **activemq**
Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.
- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::generated**

7.203 src/main/activemq/wireformat/openwire/marshal/generated/- ConnectionInfoMarshaller.h File Reference

```
#include <activemq/wireformat/openwire/marshal/generated/-  
BaseCommandMarshaller.h> #include <decaf/io/DataInput-  
Stream.h> #include <decaf/io/DataOutputStream.h> #include  
<decaf/io/IOException.h> #include <activemq/util/Config.-  
h> #include <activemq/commands/DataStructure.h> #include  
<activemq/wireformat/openwire/OpenWireFormat.h> #include  
<activemq/wireformat/openwire/utils/BooleanStream.h>
```

Data Structures

- class **activemq::wireformat::openwire::marshal::generated::ConnectionInfoMarshaller**

*Marshaling code for Open Wire Format for **ConnectionInfoMarshaller** (p. 974).*

7.204 src/main/activemq/wireformat/openwire/marshal/generated/ConsumerControlMarshaller.h File

Reference

3069

Namespaces

- namespace **activemq**
Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.
- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::generated**

7.204 src/main/activemq/wireformat/openwire/marshal/generated/ConsumerControlMarshaller.h File Reference

```
#include <activemq/wireformat/openwire/marshal/generated/BaseCommandMarshaller.h> #include <decaf/io/DataInputStream.h> #include <decaf/io/DataOutputStream.h> #include <decaf/io/IOException.h> #include <activemq/util/Config.h> #include <activemq/commands/DataStructure.h> #include <activemq/wireformat/openwire/OpenWireFormat.h> #include <activemq/wireformat/openwire/utils/BooleanStream.h>
```

Data Structures

- class **activemq::wireformat::openwire::marshal::generated::ConsumerControlMarshaller**
*Marshaling code for Open Wire Format for **ConsumerControlMarshaller** (p. 996).*

Namespaces

- namespace **activemq**
Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.
- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::generated**

7.205 src/main/activemq/wireformat/openwire/marshal/generated/ConsumerIdMarshaller.h File Reference

```
#include <activemq/wireformat/openwire/marshal/BaseDataStreamMarshaller.h> #include <decaf/io/DataInputStream.h> #include <decaf/io/DataOutputStream.h> #include <decaf/io/IOException.h> #include <activemq/util/Config.h> #include
```

```
<activemq/commands/DataSetStructure.h> #include <activemq/wireformat/openwire/
OpenWireFormat.h> #include <activemq/wireformat/openwire/utils/-
BooleanStream.h>
```

Data Structures

- class **activemq::wireformat::openwire::marshal::generated::ConsumerId-Marshaller**

*Marshaling code for Open Wire Format for **ConsumerIdMarshaller** (p. 1005).*

Namespaces

- namespace **activemq**
Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.
- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::generated**

7.206 src/main/activemq/wireformat/openwire/marshal/generated/-ConsumerInfoMarshaller.h File Reference

```
#include <activemq/wireformat/openwire/marshal/generated/-
BaseCommandMarshaller.h> #include <decaf/io/DataInput-
Stream.h> #include <decaf/io/DataOutputStream.h> #include
<decaf/io/IOException.h> #include <activemq/util/Config.-
h> #include <activemq/commands/DataSetStructure.h> #include
<activemq/wireformat/openwire/OpenWireFormat.h> #include
<activemq/wireformat/openwire/utils/BooleanStream.h>
```

Data Structures

- class **activemq::wireformat::openwire::marshal::generated::ConsumerInfo-Marshaller**

*Marshaling code for Open Wire Format for **ConsumerInfoMarshaller** (p. 1017).*

Namespaces

- namespace **activemq**
Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.
- namespace **activemq::wireformat**

7.207 src/main/activemq/wireformat/openwire/marshal/generated/ControlCommandMarshaller.h File

Reference

3071

- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::generated**

7.207 src/main/activemq/wireformat/openwire/marshal/generated/-ControlCommandMarshaller.h File Reference

```
#include <activemq/wireformat/openwire/marshal/generated/-  
BaseCommandMarshaller.h> #include <decaf/io/DataInput-  
Stream.h> #include <decaf/io/DataOutputStream.h> #include  
<decaf/io/IOException.h> #include <activemq/util/Config.-  
h> #include <activemq/commands/DataStructure.h> #include  
<activemq/wireformat/openwire/OpenWireFormat.h> #include  
<activemq/wireformat/openwire/utils/BooleanStream.h>
```

Data Structures

- class **activemq::wireformat::openwire::marshal::generated::ControlCommandMarshaller**

*Marshaling code for Open Wire Format for **ControlCommandMarshaller** (p. 1025).*

Namespaces

- namespace **activemq**
Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.
- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::generated**

7.208 src/main/activemq/wireformat/openwire/marshal/generated/-DataArrayResponseMarshaller.h File Reference

```
#include <activemq/wireformat/openwire/marshal/generated/-  
ResponseMarshaller.h> #include <decaf/io/DataInputStream.-  
h> #include <decaf/io/DataOutputStream.h> #include <decaf/io/-  
IOException.h> #include <activemq/util/Config.h> #include  
<activemq/commands/DataStructure.h> #include <activemq/wireformat/openwire/-  
OpenWireFormat.h> #include <activemq/wireformat/openwire/utils/-  
BooleanStream.h>
```

Data Structures

- class **activemq::wireformat::openwire::marshal::generated::DataArrayResponseMarshaller**

*Marshaling code for Open Wire Format for **DataArrayResponseMarshaller** (p. 1072).*

Namespaces

- namespace **activemq**
Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.
- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::generated**

7.209 src/main/activemq/wireformat/openwire/marshal/generated/-DataResponseMarshaller.h File Reference

```
#include <activemq/wireformat/openwire/marshal/generated/-
ResponseMarshaller.h> #include <decaf/io/DataInputStream.-
h> #include <decaf/io/DataOutputStream.h> #include <decaf/io/-
IOException.h> #include <activemq/util/Config.h> #include
<activemq/commands/DataStructure.h> #include <activemq/wireformat/openwire
OpenWireFormat.h> #include <activemq/wireformat/openwire/utis/-
BooleanStream.h>
```

Data Structures

- class **activemq::wireformat::openwire::marshal::generated::DataResponseMarshaller**

*Marshaling code for Open Wire Format for **DataResponseMarshaller** (p. 1115).*

Namespaces

- namespace **activemq**
Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.
- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::generated**

7.210 src/main/activemq/wireformat/openwire/marshal/generated/Destination-InfoMarshaller.h File

Reference

3073

7.210 src/main/activemq/wireformat/openwire/marshal/generated/DestinationInfoMarshaller.h File Reference

```
#include <activemq/wireformat/openwire/marshal/generated/-  
BaseCommandMarshaller.h> #include <decaf/io/DataInput-  
Stream.h> #include <decaf/io/DataOutputStream.h> #include  
<decaf/io/IOException.h> #include <activemq/util/Config.-  
h> #include <activemq/commands/DataStructure.h> #include  
<activemq/wireformat/openwire/OpenWireFormat.h> #include  
<activemq/wireformat/openwire/Utils/BooleanStream.h>
```

Data Structures

- class **activemq::wireformat::openwire::marshal::generated::Destination-InfoMarshaller**

*Marshaling code for Open Wire Format for **DestinationInfoMarshaller** (p. 1218).*

Namespaces

- namespace **activemq**
Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.
- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::generated**

7.211 src/main/activemq/wireformat/openwire/marshal/generated/DiscoveryEventMarshaller.h File Reference

```
#include <activemq/wireformat/openwire/marshal/BaseData-  
StreamMarshaller.h> #include <decaf/io/DataInputStream.-  
h> #include <decaf/io/DataOutputStream.h> #include <decaf/io/-  
IOException.h> #include <activemq/util/Config.h> #include  
<activemq/commands/DataStructure.h> #include <activemq/wireformat/openwire/-  
OpenWireFormat.h> #include <activemq/wireformat/openwire/Utils/-  
BooleanStream.h>
```

Data Structures

- class **activemq::wireformat::openwire::marshal::generated::Discovery-EventMarshaller**

*Marshaling code for Open Wire Format for **DiscoveryEventMarshaller** (p. 1230).*

Namespaces

- namespace **activemq**
Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.
- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::generated**

7.212 src/main/activemq/wireformat/openwire/marshal/generated/-ExceptionResponseMarshaller.h File Reference

```
#include <activemq/wireformat/openwire/marshal/generated/-
ResponseMarshaller.h> #include <decaf/io/DataInputStream.-
h> #include <decaf/io/DataOutputStream.h> #include <decaf/io/-
IOException.h> #include <activemq/util/Config.h> #include
<activemq/commands/DataStructure.h> #include <activemq/wireformat/openwire
OpenWireFormat.h> #include <activemq/wireformat/openwire/utis/-
BooleanStream.h>
```

Data Structures

- class **activemq::wireformat::openwire::marshal::generated::Exception-ResponseMarshaller**
*Marshaling code for Open Wire Format for **ExceptionResponseMarshaller** (p. 1290).*

Namespaces

- namespace **activemq**
Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.
- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::generated**

7.213 src/main/activemq/wireformat/openwire/marshal/generated/-FlushCommandMarshaller.h File Reference

```
#include <activemq/wireformat/openwire/marshal/generated/-
BaseCommandMarshaller.h> #include <decaf/io/DataInput-
Stream.h> #include <decaf/io/DataOutputStream.h> #include
```


7.214 src/main/activemq/wireformat/openwire/marshal/generated/IntegerResponseMarshaller.h File

Reference

3075

```
<decaf/io/IOException.h> #include <activemq/util/Config.h>
#include <activemq/commands/DataStructure.h> #include
<activemq/wireformat/openwire/OpenWireFormat.h> #include
<activemq/wireformat/openwire/utils/BooleanStream.h>
```

Data Structures

- class **activemq::wireformat::openwire::marshal::generated::FlushCommandMarshaller**

*Marshaling code for Open Wire Format for **FlushCommandMarshaller** (p. 1383).*

Namespaces

- namespace **activemq**
Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.
- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::generated**

7.214 src/main/activemq/wireformat/openwire/marshal/generated/-IntegerResponseMarshaller.h File Reference

```
#include <activemq/wireformat/openwire/marshal/generated/-
ResponseMarshaller.h> #include <decaf/io/DataInputStream.-
h> #include <decaf/io/DataOutputStream.h> #include <decaf/io/-
IOException.h> #include <activemq/util/Config.h> #include
<activemq/commands/DataStructure.h> #include <activemq/wireformat/openwire/-
OpenWireFormat.h> #include <activemq/wireformat/openwire/utils/-
BooleanStream.h>
```

Data Structures

- class **activemq::wireformat::openwire::marshal::generated::IntegerResponseMarshaller**

*Marshaling code for Open Wire Format for **IntegerResponseMarshaller** (p. 1519).*

Namespaces

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

- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::generated**

7.215 src/main/activemq/wireformat/openwire/marshal/generated/- JournalQueueAckMarshaller.h File Reference

```
#include <activemq/wireformat/openwire/marshal/BaseData-  
StreamMarshaller.h> #include <decaf/io/DataInputStream.-  
h> #include <decaf/io/DataOutputStream.h> #include <decaf/io/-  
IOException.h> #include <activemq/util/Config.h> #include  
<activemq/commands/DataStructure.h> #include <activemq/wireformat/openwire  
OpenWireFormat.h> #include <activemq/wireformat/openwire/utis/-  
BooleanStream.h>
```

Data Structures

- class **activemq::wireformat::openwire::marshal::generated::JournalQueue-
AckMarshaller**

*Marshaling code for Open Wire Format for **JournalQueueAckMarshaller** (p. 1564).*

Namespaces

- namespace **activemq**
*Licensed to the Apache Software Foundation (ASF) under one or more contributor
license agreements.*
- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::generated**

7.216 src/main/activemq/wireformat/openwire/marshal/generated/- JournalTopicAckMarshaller.h File Reference

```
#include <activemq/wireformat/openwire/marshal/BaseData-  
StreamMarshaller.h> #include <decaf/io/DataInputStream.-  
h> #include <decaf/io/DataOutputStream.h> #include <decaf/io/-  
IOException.h> #include <activemq/util/Config.h> #include  
<activemq/commands/DataStructure.h> #include <activemq/wireformat/openwire  
OpenWireFormat.h> #include <activemq/wireformat/openwire/utis/-  
BooleanStream.h>
```

7.217 src/main/activemq/wireformat/openwire/marshal/generated/JournalTrace-Marshaller.h File

Reference

3077

Data Structures

- class **activemq::wireformat::openwire::marshal::generated::JournalTopic-AckMarshaller**

*Marshaling code for Open Wire Format for **JournalTopicAckMarshaller** (p. 1572).*

Namespaces

- namespace **activemq**
Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.
- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::generated**

7.217 src/main/activemq/wireformat/openwire/marshal/generated/-JournalTraceMarshaller.h File Reference

```
#include <activemq/wireformat/openwire/marshal/BaseData-  
StreamMarshaller.h> #include <decaf/io/DataInputStream.-  
h> #include <decaf/io/DataOutputStream.h> #include <decaf/io/-  
IOException.h> #include <activemq/util/Config.h> #include  
<activemq/commands/DataStructure.h> #include <activemq/wireformat/openwire/-  
OpenWireFormat.h> #include <activemq/wireformat/openwire/utis/-  
BooleanStream.h>
```

Data Structures

- class **activemq::wireformat::openwire::marshal::generated::JournalTrace-Marshaller**

*Marshaling code for Open Wire Format for **JournalTraceMarshaller** (p. 1579).*

Namespaces

- namespace **activemq**
Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.
- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::generated**

7.218 src/main/activemq/wireformat/openwire/marshal/generated/-JournalTransactionMarshaller.h File Reference

```
#include <activemq/wireformat/openwire/marshal/BaseDataStreamMarshaller.h> #include <decaf/io/DataInputStream.h> #include <decaf/io/DataOutputStream.h> #include <decaf/io/IOException.h> #include <activemq/util/Config.h> #include <activemq/commands/DataStructure.h> #include <activemq/wireformat/openwire/OpenWireFormat.h> #include <activemq/wireformat/openwire/utils/BooleanStream.h>
```

Data Structures

- class **activemq::wireformat::openwire::marshal::generated::JournalTransactionMarshaller**

*Marshaling code for Open Wire Format for **JournalTransactionMarshaller** (p. 1587).*

Namespaces

- namespace **activemq**
Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.
- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::generated**

7.219 src/main/activemq/wireformat/openwire/marshal/generated/-KeepAliveInfoMarshaller.h File Reference

```
#include <activemq/wireformat/openwire/marshal/generated/BaseCommandMarshaller.h> #include <decaf/io/DataInputStream.h> #include <decaf/io/DataOutputStream.h> #include <decaf/io/IOException.h> #include <activemq/util/Config.h> #include <activemq/commands/DataStructure.h> #include <activemq/wireformat/openwire/OpenWireFormat.h> #include <activemq/wireformat/openwire/utils/BooleanStream.h>
```

Data Structures

- class **activemq::wireformat::openwire::marshal::generated::KeepAliveInfoMarshaller**

*Marshaling code for Open Wire Format for **KeepAliveInfoMarshaller** (p. 1594).*

7.220 src/main/activemq/wireformat/openwire/marshal/generated/LastPartialCommandMarshaller.h File

Reference

3079

Namespaces

- namespace **activemq**
Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.
- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::generated**

7.220 src/main/activemq/wireformat/openwire/marshal/generated/LastPartialCommandMarshaller.h File Reference

```
#include <activemq/wireformat/openwire/marshal/generated/PartialCommandMarshaller.h> #include <decaf/io/DataInputStream.h> #include <decaf/io/DataOutputStream.h> #include <decaf/io/IOException.h> #include <activemq/util/Config.h> #include <activemq/commands/DataStructure.h> #include <activemq/wireformat/openwire/OpenWireFormat.h> #include <activemq/wireformat/openwire/utils/BooleanStream.h>
```

Data Structures

- class **activemq::wireformat::openwire::marshal::generated::LastPartialCommandMarshaller**
*Marshaling code for Open Wire Format for **LastPartialCommandMarshaller** (p. 1608).*

Namespaces

- namespace **activemq**
Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.
- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::generated**

7.221 src/main/activemq/wireformat/openwire/marshal/generated/LocalTransactionIdMarshaller.h File Reference

```
#include <activemq/wireformat/openwire/marshal/generated/TransactionIdMarshaller.h> #include <decaf/io/DataInputStream.h> #include <decaf/io/DataOutputStream.h> #include
```

```
<decaf/io/IOException.h> #include <activemq/util/Config.-
h> #include <activemq/commands/DataStructure.h> #include
<activemq/wireformat/openwire/OpenWireFormat.h> #include
<activemq/wireformat/openwire/utils/BooleanStream.h>
```

Data Structures

- class **activemq::wireformat::openwire::marshal::generated::LocalTransactionIdMarshaller**

*Marshaling code for Open Wire Format for **LocalTransactionIdMarshaller** (p. 1678).*

Namespaces

- namespace **activemq**
Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.
- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::generated**

7.222 src/main/activemq/wireformat/openwire/marshal/generated/-MarshallerFactory.h File Reference

```
#include <activemq/wireformat/openwire/OpenWireFormat.-
h>
```

Data Structures

- class **activemq::wireformat::openwire::marshal::generated::MarshallerFactory**

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

Namespaces

- namespace **activemq**
Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.
- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::generated**

7.223 src/main/activemq/wireformat/openwire/marshal/generated/MessageAck-Marshaller.h File

Reference

3081

7.223 src/main/activemq/wireformat/openwire/marshal/generated/MessageAckMarshaller.h File Reference

```
#include <activemq/wireformat/openwire/marshal/generated/-  
BaseCommandMarshaller.h> #include <decaf/io/DataInput-  
Stream.h> #include <decaf/io/DataOutputStream.h> #include  
<decaf/io/IOException.h> #include <activemq/util/Config.-  
h> #include <activemq/commands/DataStructure.h> #include  
<activemq/wireformat/openwire/OpenWireFormat.h> #include  
<activemq/wireformat/openwire/Utils/BooleanStream.h>
```

Data Structures

- class **activemq::wireformat::openwire::marshal::generated::MessageAck-Marshaller**

*Marshaling code for Open Wire Format for **MessageAckMarshaller** (p. 1873).*

Namespaces

- namespace **activemq**
Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.
- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::generated**

7.224 src/main/activemq/wireformat/openwire/marshal/generated/MessageDispatchMarshaller.h File Reference

```
#include <activemq/wireformat/openwire/marshal/generated/-  
BaseCommandMarshaller.h> #include <decaf/io/DataInput-  
Stream.h> #include <decaf/io/DataOutputStream.h> #include  
<decaf/io/IOException.h> #include <activemq/util/Config.-  
h> #include <activemq/commands/DataStructure.h> #include  
<activemq/wireformat/openwire/OpenWireFormat.h> #include  
<activemq/wireformat/openwire/Utils/BooleanStream.h>
```

Data Structures

- class **activemq::wireformat::openwire::marshal::generated::Message-DispatchMarshaller**

*Marshaling code for Open Wire Format for **MessageDispatchMarshaller** (p. 1890).*

Namespaces

- namespace **activemq**
Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.
- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::generated**

7.225 src/main/activemq/wireformat/openwire/marshal/generated/- MessageDispatchNotificationMarshaller.h File Reference

```
#include <activemq/wireformat/openwire/marshal/generated/-  
BaseCommandMarshaller.h> #include <decaf/io/DataInput-  
Stream.h> #include <decaf/io/DataOutputStream.h> #include  
<decaf/io/IOException.h> #include <activemq/util/Config.-  
h> #include <activemq/commands/DataStructure.h> #include  
<activemq/wireformat/openwire/OpenWireFormat.h> #include  
<activemq/wireformat/openwire/Utils/BooleanStream.h>
```

Data Structures

- class **activemq::wireformat::openwire::marshal::generated::Message-
DispatchNotificationMarshaller**
*Marshaling code for Open Wire Format for **MessageDispatchNotificationMarshaller**
(p. 1899).*

Namespaces

- namespace **activemq**
Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.
- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::generated**

7.226 src/main/activemq/wireformat/openwire/marshal/generated/- MessageIdMarshaller.h File Reference

```
#include <activemq/wireformat/openwire/marshal/BaseData-  
StreamMarshaller.h> #include <decaf/io/DataInputStream.-
```


7.227

src/main/activemq/wireformat/openwire/marshal/generated/MessageMarshaller.h

File Reference

3083

```
h> #include <decaf/io/DataOutputStream.h> #include <decaf/io/-  
IOException.h> #include <activemq/util/Config.h> #include  
<activemq/commands/DataStructure.h> #include <activemq/wireformat/openwire/-  
OpenWireFormat.h> #include <activemq/wireformat/openwire/utis/-  
BooleanStream.h>
```

Data Structures

- class **activemq::wireformat::openwire::marshal::generated::MessageId-
Marshaller**

*Marshaling code for Open Wire Format for **MessageIdMarshaller** (p. 1912).*

Namespaces

- namespace **activemq**
*Licensed to the Apache Software Foundation (ASF) under one or more contributor
license agreements.*
- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::generated**

7.227 src/main/activemq/wireformat/openwire/marshal/generated/- MessageMarshaller.h File Reference

```
#include <activemq/wireformat/openwire/marshal/generated/-  
BaseCommandMarshaller.h> #include <decaf/io/DataInput-  
Stream.h> #include <decaf/io/DataOutputStream.h> #include  
<decaf/io/IOException.h> #include <activemq/util/Config.-  
h> #include <activemq/commands/DataStructure.h> #include  
<activemq/wireformat/openwire/OpenWireFormat.h> #include  
<activemq/wireformat/openwire/utis/BooleanStream.h>
```

Data Structures

- class **activemq::wireformat::openwire::marshal::generated::Message-
Marshaller**

*Marshaling code for Open Wire Format for **MessageMarshaller** (p. 1917).*

Namespaces

- namespace **activemq**

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

- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::generated**

7.228 src/main/activemq/wireformat/openwire/marshal/generated/-MessagePullMarshaller.h File Reference

```
#include <activemq/wireformat/openwire/marshal/generated/-
BaseCommandMarshaller.h> #include <decaf/io/DataInput-
Stream.h> #include <decaf/io/DataOutputStream.h> #include
<decaf/io/IOException.h> #include <activemq/util/Config.-
h> #include <activemq/commands/DataStructure.h> #include
<activemq/wireformat/openwire/OpenWireFormat.h> #include
<activemq/wireformat/openwire/utils/BooleanStream.h>
```

Data Structures

- class **activemq::wireformat::openwire::marshal::generated::MessagePullMarshaller**

*Marshaling code for Open Wire Format for **MessagePullMarshaller** (p. 1944).*

Namespaces

- namespace **activemq**
Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.
- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::generated**

7.229 src/main/activemq/wireformat/openwire/marshal/generated/-NetworkBridgeFilterMarshaller.h File Reference

```
#include <activemq/wireformat/openwire/marshal/BaseData-
StreamMarshaller.h> #include <decaf/io/DataInputStream.-
h> #include <decaf/io/DataOutputStream.h> #include <decaf/io/-
IOException.h> #include <activemq/util/Config.h> #include
<activemq/commands/DataStructure.h> #include <activemq/wireformat/openwire
OpenWireFormat.h> #include <activemq/wireformat/openwire/utils/-
BooleanStream.h>
```

7.230 src/main/activemq/wireformat/openwire/marshal/generated/Partial-CommandMarshaller.h File

Reference

3085

Data Structures

- class **activemq::wireformat::openwire::marshal::generated::Network-BridgeFilterMarshaller**

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

Namespaces

- namespace **activemq**
Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.
- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::generated**

7.230 src/main/activemq/wireformat/openwire/marshal/generated/-PartialCommandMarshaller.h File Reference

```
#include <activemq/wireformat/openwire/marshal/BaseData-  
StreamMarshaller.h> #include <decaf/io/DataInputStream.-  
h> #include <decaf/io/DataOutputStream.h> #include <decaf/io/-  
IOException.h> #include <activemq/util/Config.h> #include  
<activemq/commands/DataStructure.h> #include <activemq/wireformat/openwire/-  
OpenWireFormat.h> #include <activemq/wireformat/openwire/utis/-  
BooleanStream.h>
```

Data Structures

- class **activemq::wireformat::openwire::marshal::generated::PartialCommand-Marshaller**

*Marshaling code for Open Wire Format for **PartialCommandMarshaller** (p. 2079).*

Namespaces

- namespace **activemq**
Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.
- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::generated**

7.231 src/main/activemq/wireformat/openwire/marshal/generated/- ProducerAckMarshaller.h File Reference

```
#include <activemq/wireformat/openwire/marshal/generated/-  
BaseCommandMarshaller.h> #include <decaf/io/DataInput-  
Stream.h> #include <decaf/io/DataOutputStream.h> #include  
<decaf/io/IOException.h> #include <activemq/util/Config.-  
h> #include <activemq/commands/DataStructure.h> #include  
<activemq/wireformat/openwire/OpenWireFormat.h> #include  
<activemq/wireformat/openwire/utils/BooleanStream.h>
```

Data Structures

- class **activemq::wireformat::openwire::marshal::generated::ProducerAckMarshaller**

*Marshaling code for Open Wire Format for **ProducerAckMarshaller** (p. 2175).*

Namespaces

- namespace **activemq**
Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.
- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::generated**

7.232 src/main/activemq/wireformat/openwire/marshal/generated/- ProducerIdMarshaller.h File Reference

```
#include <activemq/wireformat/openwire/marshal/BaseData-  
StreamMarshaller.h> #include <decaf/io/DataInputStream.-  
h> #include <decaf/io/DataOutputStream.h> #include <decaf/io/-  
IOException.h> #include <activemq/util/Config.h> #include  
<activemq/commands/DataStructure.h> #include <activemq/wireformat/openwire  
OpenWireFormat.h> #include <activemq/wireformat/openwire/utils/-  
BooleanStream.h>
```

Data Structures

- class **activemq::wireformat::openwire::marshal::generated::ProducerIdMarshaller**

*Marshaling code for Open Wire Format for **ProducerIdMarshaller** (p. 2186).*

7.233 src/main/activemq/wireformat/openwire/marshal/generated/ProducerInfoMarshaller.h File

Reference

3087

Namespaces

- namespace **activemq**
Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.
- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::generated**

7.233 src/main/activemq/wireformat/openwire/marshal/generated/-ProducerInfoMarshaller.h File Reference

```
#include <activemq/wireformat/openwire/marshal/generated/-BaseCommandMarshaller.h> #include <decaf/io/DataInputStream.h> #include <decaf/io/DataOutputStream.h> #include <decaf/io/IOException.h> #include <activemq/util/Config.h> #include <activemq/commands/DataStructure.h> #include <activemq/wireformat/openwire/OpenWireFormat.h> #include <activemq/wireformat/openwire/utls/BooleanStream.h>
```

Data Structures

- class **activemq::wireformat::openwire::marshal::generated::ProducerInfoMarshaller**
*Marshaling code for Open Wire Format for **ProducerInfoMarshaller** (p. 2195).*

Namespaces

- namespace **activemq**
Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.
- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::generated**

7.234 src/main/activemq/wireformat/openwire/marshal/generated/-RemoveInfoMarshaller.h File Reference

```
#include <activemq/wireformat/openwire/marshal/generated/-BaseCommandMarshaller.h> #include <decaf/io/DataInputStream.h> #include <decaf/io/DataOutputStream.h> #include
```

```
<decaf/io/IOException.h> #include <activemq/util/Config.-
h> #include <activemq/commands/DataStructure.h> #include
<activemq/wireformat/openwire/OpenWireFormat.h> #include
<activemq/wireformat/openwire/utils/BooleanStream.h>
```

Data Structures

- class **activemq::wireformat::openwire::marshal::generated::RemoveInfoMarshall**

*Marshaling code for Open Wire Format for **RemoveInfoMarshall** (p. 2271).*

Namespaces

- namespace **activemq**
Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.
- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::generated**

7.235 src/main/activemq/wireformat/openwire/marshal/generated/-RemoveSubscriptionInfoMarshall.h File Reference

```
#include <activemq/wireformat/openwire/marshal/generated/-
BaseCommandMarshall.h> #include <decaf/io/DataInput-
Stream.h> #include <decaf/io/DataOutputStream.h> #include
<decaf/io/IOException.h> #include <activemq/util/Config.-
h> #include <activemq/commands/DataStructure.h> #include
<activemq/wireformat/openwire/OpenWireFormat.h> #include
<activemq/wireformat/openwire/utils/BooleanStream.h>
```

Data Structures

- class **activemq::wireformat::openwire::marshal::generated::RemoveSubscriptionInfoMarshall**

*Marshaling code for Open Wire Format for **RemoveSubscriptionInfoMarshall** (p. 2280).*

Namespaces

- namespace **activemq**

7.236 src/main/activemq/wireformat/openwire/marshal/generated/Replay-CommandMarshaller.h File

Reference

3089

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

- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::generated**

7.236 src/main/activemq/wireformat/openwire/marshal/generated/-ReplayCommandMarshaller.h File Reference

```
#include <activemq/wireformat/openwire/marshal/generated/-BaseCommandMarshaller.h> #include <decaf/io/DataInputStream.h> #include <decaf/io/DataOutputStream.h> #include <decaf/io/IOException.h> #include <activemq/util/Config.h> #include <activemq/commands/DataStructure.h> #include <activemq/wireformat/openwire/OpenWireFormat.h> #include <activemq/wireformat/openwire/utils/BooleanStream.h>
```

Data Structures

- class **activemq::wireformat::openwire::marshal::generated::ReplayCommandMarshaller**

*Marshaling code for Open Wire Format for **ReplayCommandMarshaller** (p. 2287).*

Namespaces

- namespace **activemq**
Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.
- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::generated**

7.237 src/main/activemq/wireformat/openwire/marshal/generated/-ResponseMarshaller.h File Reference

```
#include <activemq/wireformat/openwire/marshal/generated/-BaseCommandMarshaller.h> #include <decaf/io/DataInputStream.h> #include <decaf/io/DataOutputStream.h> #include <decaf/io/IOException.h> #include <activemq/util/Config.h> #include <activemq/commands/DataStructure.h> #include <activemq/wireformat/openwire/OpenWireFormat.h> #include <activemq/wireformat/openwire/utils/BooleanStream.h>
```

Data Structures

- class **activemq::wireformat::openwire::marshal::generated::ResponseMarshall**

*Marshaling code for Open Wire Format for **ResponseMarshall** (p. 2307).*

Namespaces

- namespace **activemq**
Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.
- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::generated**

7.238 src/main/activemq/wireformat/openwire/marshal/generated/-SessionIdMarshaller.h File Reference

```
#include <activemq/wireformat/openwire/marshal/BaseDataStreamMarshaller.h> #include <decaf/io/DataInputStream.h> #include <decaf/io/DataOutputStream.h> #include <decaf/io/IOException.h> #include <activemq/util/Config.h> #include <activemq/commands/DataStructure.h> #include <activemq/wireformat/openwire/OpenWireFormat.h> #include <activemq/wireformat/openwire/Utils/BooleanStream.h>
```

Data Structures

- class **activemq::wireformat::openwire::marshal::generated::SessionIdMarshaller**

*Marshaling code for Open Wire Format for **SessionIdMarshaller** (p. 2382).*

Namespaces

- namespace **activemq**
Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.
- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::generated**

7.239 src/main/activemq/wireformat/openwire/marshal/generated/SessionInfoMarshaller.h File

Reference

3091

7.239 src/main/activemq/wireformat/openwire/marshal/generated/SessionInfoMarshaller.h File Reference

```
#include <activemq/wireformat/openwire/marshal/generated/BaseCommandMarshaller.h> #include <decaf/io/DataInputStream.h> #include <decaf/io/DataOutputStream.h> #include <decaf/io/IOException.h> #include <activemq/util/Config.h> #include <activemq/commands/DataStructure.h> #include <activemq/wireformat/openwire/OpenWireFormat.h> #include <activemq/wireformat/openwire/Utils/BooleanStream.h>
```

Data Structures

- class **activemq::wireformat::openwire::marshal::generated::SessionInfoMarshaller**

*Marshaling code for Open Wire Format for **SessionInfoMarshaller** (p. 2390).*

Namespaces

- namespace **activemq**
Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.
- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::generated**

7.240 src/main/activemq/wireformat/openwire/marshal/generated/ShutdownInfoMarshaller.h File Reference

```
#include <activemq/wireformat/openwire/marshal/generated/BaseCommandMarshaller.h> #include <decaf/io/DataInputStream.h> #include <decaf/io/DataOutputStream.h> #include <decaf/io/IOException.h> #include <activemq/util/Config.h> #include <activemq/commands/DataStructure.h> #include <activemq/wireformat/openwire/OpenWireFormat.h> #include <activemq/wireformat/openwire/Utils/BooleanStream.h>
```

Data Structures

- class **activemq::wireformat::openwire::marshal::generated::ShutdownInfoMarshaller**

*Marshaling code for Open Wire Format for **ShutdownInfoMarshaller** (p. 2434).*

Namespaces

- namespace **activemq**
Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.
- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::generated**

7.241 src/main/activemq/wireformat/openwire/marshal/generated/-SubscriptionInfoMarshaller.h File Reference

```
#include <activemq/wireformat/openwire/marshal/BaseDataStreamMarshaller.h> #include <decaf/io/DataInputStream.h> #include <decaf/io/DataOutputStream.h> #include <decaf/io/IOException.h> #include <activemq/util/Config.h> #include <activemq/commands/DataStructure.h> #include <activemq/wireformat/openwire/OpenWireFormat.h> #include <activemq/wireformat/openwire/utils/BooleanStream.h>
```

Data Structures

- class **activemq::wireformat::openwire::marshal::generated::SubscriptionInfoMarshaller**
*Marshaling code for Open Wire Format for **SubscriptionInfoMarshaller** (p. 2635).*

Namespaces

- namespace **activemq**
Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.
- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::generated**

7.242 src/main/activemq/wireformat/openwire/marshal/generated/-TransactionIdMarshaller.h File Reference

```
#include <activemq/wireformat/openwire/marshal/BaseDataStreamMarshaller.h> #include <decaf/io/DataInputStream.h> #include <decaf/io/DataOutputStream.h> #include <decaf/io/IOException.h> #include <activemq/util/Config.h> #include
```

7.243 src/main/activemq/wireformat/openwire/marshal/generated/Transaction-InfoMarshaller.h File

Reference

3093

```
<activemq/commands/DataStructure.h> #include <activemq/wireformat/openwire/-  
OpenWireFormat.h> #include <activemq/wireformat/openwire/Utils/-  
BooleanStream.h>
```

Data Structures

- class **activemq::wireformat::openwire::marshal::generated::TransactionId-Marshaller**

*Marshaling code for Open Wire Format for **TransactionIdMarshaller** (p. 2770).*

Namespaces

- namespace **activemq**
Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.
- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::generated**

7.243 src/main/activemq/wireformat/openwire/marshal/generated/-TransactionInfoMarshaller.h File Reference

```
#include <activemq/wireformat/openwire/marshal/generated/-  
BaseCommandMarshaller.h> #include <decaf/io/DataInput-  
Stream.h> #include <decaf/io/DataOutputStream.h> #include  
<decaf/io/IOException.h> #include <activemq/util/Config.-  
h> #include <activemq/commands/DataStructure.h> #include  
<activemq/wireformat/openwire/OpenWireFormat.h> #include  
<activemq/wireformat/openwire/Utils/BooleanStream.h>
```

Data Structures

- class **activemq::wireformat::openwire::marshal::generated::Transaction-InfoMarshaller**

*Marshaling code for Open Wire Format for **TransactionInfoMarshaller** (p. 2778).*

Namespaces

- namespace **activemq**
Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.
- namespace **activemq::wireformat**

- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::generated**

7.244 src/main/activemq/wireformat/openwire/marshal/generated/-WireFormatInfoMarshaller.h File Reference

```
#include <activemq/wireformat/openwire/marshal/BaseDataStreamMarshaller.h> #include <decaf/io/DataInputStream.h> #include <decaf/io/DataOutputStream.h> #include <decaf/io/IOException.h> #include <activemq/util/Config.h> #include <activemq/commands/DataStructure.h> #include <activemq/wireformat/openwire/OpenWireFormat.h> #include <activemq/wireformat/openwire/utils/BooleanStream.h>
```

Data Structures

- class **activemq::wireformat::openwire::marshal::generated::WireFormatInfoMarshaller**

*Marshaling code for Open Wire Format for **WireFormatInfoMarshaller** (p. 2900).*

Namespaces

- namespace **activemq**
Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.
- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::generated**

7.245 src/main/activemq/wireformat/openwire/marshal/generated/-XATransactionIdMarshaller.h File Reference

```
#include <activemq/wireformat/openwire/marshal/generated/TransactionIdMarshaller.h> #include <decaf/io/DataInputStream.h> #include <decaf/io/DataOutputStream.h> #include <decaf/io/IOException.h> #include <activemq/util/Config.h> #include <activemq/commands/DataStructure.h> #include <activemq/wireformat/openwire/OpenWireFormat.h> #include <activemq/wireformat/openwire/utils/BooleanStream.h>
```

7.246

src/main/activemq/wireformat/openwire/marshal/PrimitiveTypesMarshaller.h File Reference

3095

Data Structures

- class **activemq::wireformat::openwire::marshal::generated::XATransactionIdMarshaller**

*Marshaling code for Open Wire Format for **XATransactionIdMarshaller** (p. 2942).*

Namespaces

- namespace **activemq**
Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.
- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::generated**

7.246 src/main/activemq/wireformat/openwire/marshal/PrimitiveTypesMarshaller.h File Reference

```
#include <cms/CMSException.h>    #include <activemq/util/-  
Config.h>    #include <activemq/util/PrimitiveValueNode.h>  
#include <activemq/util/PrimitiveMap.h> #include <activemq/util/-  
PrimitiveList.h>    #include <decaf/io/DataOutputStream.h>  
#include <decaf/io/DataInputStream.h> #include <decaf/io/-  
IOException.h> #include <string>
```

Data Structures

- class **activemq::wireformat::openwire::marshal::PrimitiveTypesMarshaller**

This class wraps the functionality needed to marshal a primitive map to the Openwire Format's expectation of what the map looks like on the wire.

Namespaces

- namespace **activemq**
Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.
- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**

7.247 src/main/activemq/wireformat/openwire/OpenWireFormat.h - File Reference

```
#include <activemq/util/Config.h> #include <activemq/commands/-  
WireFormatInfo.h> #include <activemq/commands/DataStructure.-  
h> #include <activemq/wireformat/WireFormat.h> #include  
<activemq/wireformat/openwire/Utils/BooleanStream.h> ×  
#include <decaf/lang/Pointer.h> #include <decaf/util/-  
Properties.h> #include <decaf/util/concurrent/atomic/-  
AtomicBoolean.h> #include <decaf/lang/exceptions/Illegal-  
StateException.h> #include <decaf/lang/exceptions/Illegal-  
ArgumentException.h> #include <memory>
```

Data Structures

- class **activemq::wireformat::openwire::OpenWireFormat**

Namespaces

- namespace **activemq**
Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.
- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**

7.248 src/main/activemq/wireformat/openwire/OpenWireFormatFactory.h File Reference

```
#include <activemq/util/Config.h> #include <activemq/wireformat/-  
WireFormatFactory.h> #include <activemq/commands/Wire-  
FormatInfo.h> #include <decaf/lang/exceptions/Illegal-  
StateException.h> #include <decaf/lang/Pointer.h> #include  
<decaf/util/Properties.h>
```

Data Structures

- class **activemq::wireformat::openwire::OpenWireFormatFactory**

Namespaces

- namespace **activemq**

7.249 src/main/activemq/wireformat/openwire/OpenWireFormatNegotiator.h File Reference 3097

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

- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**

7.249 src/main/activemq/wireformat/openwire/OpenWireFormatNegotiator.h File Reference

```
#include <activemq/util/Config.h> #include <activemq/transport/-  
TransportFilter.h> #include <activemq/wireformat/openwire/-  
OpenWireFormat.h> #include <activemq/wireformat/Wire-  
FormatNegotiator.h> #include <decaf/util/concurrent/-  
Mutex.h> #include <decaf/util/concurrent/CountDownLatch.-  
h> #include <decaf/util/concurrent/Concurrent.h> #include  
<decaf/util/concurrent/atomic/AtomicBoolean.h> #include  
<decaf/lang/Pointer.h>
```

Data Structures

- class **activemq::wireformat::openwire::OpenWireFormatNegotiator**

Namespaces

- namespace **activemq**
Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.
- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**

7.250 src/main/activemq/wireformat/openwire/OpenWireResponseBuilder.h File Reference

```
#include <activemq/util/Config.h> #include <activemq/transport/mock/-  
ResponseBuilder.h> #include <decaf/util/LinkedList.h> ×  
#include <decaf/lang/Pointer.h>
```

Data Structures

- class **activemq::wireformat::openwire::OpenWireResponseBuilder**
*Used to allow a MockTransport to generate response commands to OpenWire -
Commands.*

Namespaces

- namespace **activemq**
Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.
- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**

7.251 src/main/activemq/wireformat/openwire/utils/BooleanStream.h File Reference

```
#include <decaf/io/DataInputStream.h> #include <decaf/io/-
DataOutputStream.h> #include <activemq/util/Config.h>
```

Data Structures

- class **activemq::wireformat::openwire::utils::BooleanStream**
Manages the writing and reading of boolean data streams to and from a data source such as a DataInputStream or DataOutputStream.

Namespaces

- namespace **activemq**
Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.
- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::utils**

7.252 src/main/activemq/wireformat/openwire/utils/HexTable.h File Reference

```
#include <vector> #include <string> #include <activemq/util/-
Config.h> #include <decaf/lang/exceptions/IndexOutOf-
BoundsException.h>
```

Data Structures

- class **activemq::wireformat::openwire::utils::HexTable**
*The **HexTable** (p. 1407) class maps hexadecimal strings to the value of an index into the table, i.e.*

7.253

src/main/activemq/wireformat/openwire/utils/MessagePropertyInterceptor.h File Reference

3099

Namespaces

- namespace **activemq**
Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.
- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::utils**

7.253 src/main/activemq/wireformat/openwire/utils/MessageProperty-Interceptor.h File Reference

```
#include <activemq/util/Config.h> #include <activemq/commands/-  
Message.h> #include <activemq/util/PrimitiveMap.h> #include  
<activemq/exceptions/ActiveMQException.h> #include <decaf/lang/exceptions/-  
NullPointerException.h>
```

Data Structures

- class **activemq::wireformat::openwire::utils::MessagePropertyInterceptor**
Used the base ActiveMQMessage class to intercept calls to get and set properties in order to capture the calls that use the reserved JMS properties and get and set them in the OpenWire Message properties.

Namespaces

- namespace **activemq**
Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.
- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::utils**

7.254 src/main/activemq/wireformat/stomp/StompCommandConstants.h File Reference

```
#include <cms/Destination.h> #include <activemq/util/-  
Config.h> #include <decaf/lang/exceptions/IllegalArgument-  
Exception.h> #include <string> #include <map>
```

Data Structures

- class **activemq::wireformat::stomp::StompCommandConstants**

Namespaces

- namespace **activemq**

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

- namespace **activemq::wireformat**
- namespace **activemq::wireformat::stomp**

7.255 src/main/activemq/wireformat/stomp/StompFrame.h File - Reference

```
#include <string> #include <string.h> #include <map> ×
#include <decaf/util/Properties.h> #include <decaf/io/-
DataOutputStream.h> #include <decaf/io/DataInputStream.-
h> #include <activemq/util/Config.h>
```

Data Structures

- class **activemq::wireformat::stomp::StompFrame**

A Stomp-level message frame that encloses all messages to and from the broker.

Namespaces

- namespace **activemq**

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

- namespace **activemq::wireformat**
- namespace **activemq::wireformat::stomp**

7.256 src/main/activemq/wireformat/stomp/StompHelper.h File - Reference

```
#include <activemq/util/Config.h> #include <activemq/util/-
LongSequenceGenerator.h> #include <activemq/wireformat/stomp/-
StompFrame.h> #include <activemq/commands/Message.h>
#include <activemq/commands/MessageId.h> #include <activemq/commands/-
ProducerId.h> #include <activemq/commands/ConsumerId.-
h> #include <activemq/commands/TransactionId.h> #include
<activemq/commands/ActiveMQDestination.h> #include <decaf/lang/-
Pointer.h>
```

7.257 src/main/activemq/wireformat/stomp/StompWireFormat.h File Reference 3101

Data Structures

- class **activemq::wireformat::stomp::StompHelper**
*Utility Methods used when marshaling to and from **StompFrame** (p. 2587)'s.*

Namespaces

- namespace **activemq**
Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.
- namespace **activemq::wireformat**
- namespace **activemq::wireformat::stomp**

7.257 src/main/activemq/wireformat/stomp/StompWireFormat.h - File Reference

```
#include <activemq/util/Config.h> #include <activemq/wireformat/-  
WireFormat.h> #include <activemq/wireformat/stomp/Stomp-  
Frame.h> #include <activemq/wireformat/stomp/StompHelper.-  
h> #include <decaf/util/concurrent/atomic/AtomicBoolean.-  
h> #include <decaf/io/IOException.h> #include <decaf/lang/-  
Pointer.h>
```

Data Structures

- class **activemq::wireformat::stomp::StompWireFormat**

Namespaces

- namespace **activemq**
Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.
- namespace **activemq::wireformat**
- namespace **activemq::wireformat::stomp**

7.258 src/main/activemq/wireformat/stomp/StompWireFormat-Factory.h File Reference

```
#include <activemq/util/Config.h> #include <activemq/wireformat/-  
WireFormatFactory.h> #include <activemq/wireformat/stomp/-  
StompWireFormat.h> #include <decaf/lang/Pointer.h>
```

Data Structures

- class **activemq::wireformat::stomp::StompWireFormatFactory**

Factory used to create the Stomp Wire Format instance.

Namespaces

- namespace **activemq**
Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.
- namespace **activemq::wireformat**
- namespace **activemq::wireformat::stomp**

7.259 src/main/activemq/wireformat/WireFormat.h File Reference

```
#include <activemq/wireformat/WireFormatNegotiator.h>
#include <decaf/io/DataInputStream.h> #include <decaf/io/-
DataOutputStream.h> #include <decaf/io/IOException.h> ×
#include <decaf/lang/Pointer.h> #include <activemq/util/-
Config.h> #include <activemq/commands/Command.h> #include
<activemq/transport/Transport.h> #include <decaf/lang/exceptions/-
UnsupportedOperationException.h>
```

Data Structures

- class **activemq::wireformat::WireFormat**

Provides a mechanism to marshal commands into and out of packets or into and out of streams, Channels and Datagrams.

Namespaces

- namespace **activemq**
Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.
- namespace **activemq::wireformat**

7.260 src/main/activemq/wireformat/WireFormatFactory.h File - Reference

```
#include <activemq/util/Config.h> #include <activemq/wireformat/-
WireFormat.h> #include <decaf/util/Properties.h> #include
<decaf/lang/Pointer.h> #include <decaf/lang/exceptions/-
IllegalStateException.h>
```

7.261 src/main/activemq/wireformat/WireFormatNegotiator.h File Reference 3103

Data Structures

- class **activemq::wireformat::WireFormatFactory**

The **WireFormatFactory** (p.2888) is the interface that all **WireFormatFactory** (p. 2888) classes must extend.

Namespaces

- namespace **activemq**

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

- namespace **activemq::wireformat**

7.261 src/main/activemq/wireformat/WireFormatNegotiator.h File - Reference

```
#include <activemq/util/Config.h> #include <activemq/transport/-  
TransportFilter.h> #include <decaf/lang/Pointer.h>
```

Data Structures

- class **activemq::wireformat::WireFormatNegotiator**

Defines a **WireFormatNegotiator** (p. 2904) which allows a **WireFormat** (p. 2884) to.

Namespaces

- namespace **activemq**

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

- namespace **activemq::wireformat**

7.262 src/main/activemq/wireformat/WireFormatRegistry.h File - Reference

```
#include <activemq/util/Config.h>      #include <string> ×  
#include <vector>      #include <activemq/wireformat/Wire-  
FormatFactory.h> #include <decaf/util/StlMap.h> #include  
<decaf/util/NoSuchElementException.h> #include <decaf/lang/exceptions/-  
NullPointerException.h> #include <decaf/lang/exceptions/-  
IllegalArgumentException.h>
```

Data Structures

- class **activemq::wireformat::WireFormatRegistry**

*Registry of all **WireFormat** (p. 2884) Factories that are available to the client at run-time.*

Namespaces

- namespace **activemq**

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

- namespace **activemq::wireformat**

7.263 src/main/cms/BytesMessage.h File Reference

```
#include <cms/Config.h> #include <cms/Message.h> #include  
<cms/CMSException.h> #include <cms/MessageNotReadable-  
Exception.h> #include <cms/MessageNotWriteableException.-  
h> #include <cms/MessageEOFException.h> #include <cms/-  
MessageFormatException.h>
```

Data Structures

- class **cms::BytesMessage**

*A **BytesMessage** (p. 718) object is used to send a message containing a stream of unsigned bytes.*

Namespaces

- namespace **cms**

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

7.264 src/main/cms/Closeable.h File Reference

```
#include <cms/Config.h> #include <cms/CMSException.h>
```

Data Structures

- class **cms::Closeable**

Interface for a class that implements the close method.

Namespaces

- namespace **cms**

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

7.265 src/main/decaf/io/Closeable.h File Reference

```
#include <decaf/util/Config.h>      #include <decaf/io/IO-  
Exception.h>
```

Data Structures

- class **decaf::io::Closeable**

Interface for a class that implements the close method.

Namespaces

- namespace **decaf**

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

- namespace **decaf::io**

7.266 src/main/cms/CMSException.h File Reference

```
#include <string> #include <vector> #include <iostream> ×  
#include <exception> #include <cms/Config.h>
```

Data Structures

- class **cms::CMSException**

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

Namespaces

- namespace **cms**

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

7.267 src/main/cms/CMSProperties.h File Reference

```
#include <cms/Config.h> #include <map> #include <string> ×  
#include <vector>
```

Data Structures

- class **cms::CMSProperties**
Interface for a Java-like properties object.

Namespaces

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

7.268 src/main/cms/CMSSecurityException.h File Reference

```
#include <cms/Config.h> #include <cms/CMSException.h>
```

Data Structures

- class **cms::CMSSecurityException**
This exception must be thrown when a provider rejects a user name/password submitted by a client.

Namespaces

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

7.269 src/main/cms/Connection.h File Reference

```
#include <cms/Config.h> #include <cms/Startable.h> ×  
#include <cms/Stopppable.h> #include <cms/Closeable.h> ×  
#include <cms/Session.h> #include <cms/ConnectionMeta-  
Data.h>
```


Data Structures

- class **cms::Connection**

The client's connection to its provider.

Namespaces

- namespace **cms**

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

7.270 src/main/cms/ConnectionFactory.h File Reference

```
#include <cms/Config.h>      #include <cms/Connection.h> ×  
#include <cms/CMSException.h> #include <string>
```

Data Structures

- class **cms::ConnectionFactory**

*Defines the interface for a factory that creates connection objects, the **Connection** (p. 933) objects returned implement the CMS **Connection** (p. 933) interface and hide the CMS Provider specific implementation details behind that interface.*

Namespaces

- namespace **cms**

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

7.271 src/main/cms/ConnectionMetaData.h File Reference

```
#include <cms/Config.h> #include <cms/CMSException.h>
```

Data Structures

- class **cms::ConnectionMetaData**

*A **ConnectionMetaData** (p. 978) object provides information describing the - **Connection** (p. 933) object.*

Namespaces

- namespace **cms**

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

7.272 src/main/cms/DeliveryMode.h File Reference

```
#include <cms/Config.h>
```

Data Structures

- class **cms::DeliveryMode**

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

Namespaces

- namespace **cms**

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

7.273 src/main/cms/Destination.h File Reference

```
#include <cms/CMSProperties.h> #include <cms/Config.h> ×  
#include <string>
```

Data Structures

- class **cms::Destination**

*A **Destination** (p. 1210) object encapsulates a provider-specific address.*

Namespaces

- namespace **cms**

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

7.274 src/main/cms/ExceptionListener.h File Reference

```
#include <cms/Config.h> #include <cms/CMSException.h>
```

Data Structures

- class **cms::ExceptionListener**

*If a CMS provider detects a serious problem, it notifies the client application through an **ExceptionListener** (p. 1286) that is registered with the **Connection** (p. 933).*

Namespaces

- namespace **cms**

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

7.275 src/main/cms/IllegalStateException.h File Reference

```
#include <cms/Config.h> #include <cms/CMSException.h>
```

Data Structures

- class **cms::IllegalStateException**

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

Namespaces

- namespace **cms**

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

7.276 src/main/decaf/lang/exceptions/IllegalStateException.h File - Reference

```
#include <decaf/lang/Exception.h>
```

Data Structures

- class **decaf::lang::exceptions::IllegalStateException**

Namespaces

- namespace **decaf**
Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.
- namespace **decaf::lang**
- namespace **decaf::lang::exceptions**

7.277 src/main/cms/InvalidClientIdException.h File Reference

```
#include <cms/Config.h> #include <cms/CMSException.h>
```

Data Structures

- class **cms::InvalidClientIdException**
This exception must be thrown when a client attempts to set a connection's client ID to a value that is rejected by a provider.

Namespaces

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

7.278 src/main/cms/InvalidDestinationException.h File Reference

```
#include <cms/Config.h> #include <cms/CMSException.h>
```

Data Structures

- class **cms::InvalidDestinationException**
This exception must be thrown when a destination either is not understood by a provider or is no longer valid.

Namespaces

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

7.279 src/main/cms/InvalidSelectorException.h File Reference

```
#include <cms/Config.h> #include <cms/CMSException.h>
```

Data Structures

- class **cms::InvalidSelectorException**

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

Namespaces

- namespace **cms**

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

7.280 src/main/cms/MapMessage.h File Reference

```
#include <cms/Config.h> #include <cms/Message.h> #include  
<cms/CMSException.h> #include <cms/MessageFormatException.h>  
#include <cms/MessageNotWriteableException.h>
```

Data Structures

- class **cms::MapMessage**

*A **MapMessage** (p. 1779) object is used to send a set of name-value pairs.*

Namespaces

- namespace **cms**

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

7.281 src/main/cms/MessageConsumer.h File Reference

```
#include <cms/Config.h> #include <cms/MessageListener.h>  
#include <cms/Message.h> #include <cms/Closeable.h> ×  
#include <cms/Startable.h> #include <cms/Stoppable.h>
```

Data Structures

- class **cms::MessageConsumer**

*A client uses a **MessageConsumer** (p. 1877) to received messages from a destination.*

Namespaces

- namespace **cms**

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

7.282 src/main/cms/MessageEnumeration.h File Reference

```
#include <cms/Config.h> #include <cms/Message.h> #include  
<cms/CMSException.h>
```

Data Structures

- class **cms::MessageEnumeration**

Defines an object that enumerates a collection of Messages.

Namespaces

- namespace **cms**

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

7.283 src/main/cms/MessageEOFException.h File Reference

```
#include <cms/Config.h> #include <cms/CMSException.h>
```

Data Structures

- class **cms::MessageEOFException**

*This exception must be thrown when an unexpected end of stream has been reached when a **StreamMessage** (p. 2606) or **BytesMessage** (p. 718) is being read.*

Namespaces

- namespace **cms**

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

7.284 src/main/cms/MessageFormatException.h File Reference

```
#include <cms/Config.h> #include <cms/CMSException.h>
```

Data Structures

- class **cms::MessageFormatException**

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

Namespaces

- namespace **cms**

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

7.285 src/main/cms/MessageListener.h File Reference

```
#include <cms/Config.h>
```

Data Structures

- class **cms::MessageListener**

*A **MessageListener** (p. 1916) object is used to receive asynchronously delivered messages.*

Namespaces

- namespace **cms**

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

7.286 src/main/cms/MessageNotReadableException.h File Reference

```
#include <cms/Config.h> #include <cms/CMSException.h>
```

Data Structures

- class **cms::MessageNotReadableException**

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

Namespaces

- namespace **cms**

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

7.287 src/main/cms/MessageNotWriteableException.h File Reference

```
#include <cms/Config.h> #include <cms/CMSException.h>
```

Data Structures

- class **cms::MessageNotWriteableException**

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

Namespaces

- namespace **cms**

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

7.288 src/main/cms/MessageProducer.h File Reference

```
#include <cms/Config.h> #include <cms/Message.h> #include
<cms/Destination.h> #include <cms/Closeable.h> #include
<cms/CMSException.h> #include <cms/InvalidDestination-
Exception.h> #include <cms/MessageFormatException.h> ×
#include <cms/UnsupportedOperationException.h> #include
<cms/DeliveryMode.h>
```

Data Structures

- class **cms::MessageProducer**

*A client uses a **MessageProducer** (p. 1924) object to send messages to a **Destination** (p. 1210).*

Namespaces

- namespace **cms**

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

7.289 src/main/cms/ObjectMessage.h File Reference

```
#include <cms/Config.h> #include <cms/Message.h>
```

Data Structures

- class **cms::ObjectMessage**

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

Namespaces

- namespace **cms**

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

7.290 src/main/cms/Queue.h File Reference

```
#include <cms/Config.h>      #include <cms/Destination.h> ×  
#include <cms/CMSException.h>
```

Data Structures

- class **cms::Queue**

An interface encapsulating a provider-specific queue name.

Namespaces

- namespace **cms**

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

7.291 src/main/decaf/util/Queue.h File Reference

```
#include <decaf/util/Config.h>      #include <decaf/util/-  
Iterator.h>      #include <decaf/util/AbstractCollection.h>  
#include <decaf/lang/Exception.h> #include <decaf/lang/exceptions/-  
IndexOutOfBoundsException.h> #include <decaf/util/NoSuch-  
ElementException.h>
```

Data Structures

- class **decaf::util::Queue**< **E** >

A kind of collection provides advanced operations than other basic collections, such as insertion, extraction, and inspection.

Namespaces

- namespace **decaf**

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

- namespace **decaf::util**

7.292 src/main/cms/QueueBrowser.h File Reference

```
#include <string> #include <cms/Config.h> #include <cms/-  
Closeable.h> #include <cms/Queue.h> #include <cms/Message.-  
h> #include <cms/CMSException.h> #include <cms/Message-  
Enumeration.h>
```

Data Structures

- class **cms::QueueBrowser**

*This class implements in interface for browsing the messages in a **Queue** (p. 2221) without removing them.*

Namespaces

- namespace **cms**

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

7.293 src/main/cms/Session.h File Reference

```
#include <cms/Config.h>          #include <cms/Closeable.h> ×  
#include <cms/Startable.h>      #include <cms/Stoppable.h> ×  
#include <cms/Message.h>        #include <cms/TextMessage.h> ×  
#include <cms/BytesMessage.h>   #include <cms/MapMessage.-  
h> #include <cms/StreamMessage.h> #include <cms/Message-  
Producer.h> #include <cms/MessageConsumer.h> #include  
<cms/Topic.h> #include <cms/Queue.h> #include <cms/-  
QueueBrowser.h> #include <cms/TemporaryTopic.h> #include  
<cms/TemporaryQueue.h> #include <cms/CMSException.h>
```

Data Structures

- class **cms::Session**

A **Session** (p. 2361) object is a single-threaded context for producing and consuming messages.

Namespaces

- namespace **cms**

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

7.294 src/main/cms/Startable.h File Reference

```
#include <cms/Config.h> #include <cms/CMSException.h>
```

Data Structures

- class **cms::Startable**

Interface for a class that implements the start method.

Namespaces

- namespace **cms**

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

7.295 src/main/cms/Stoppable.h File Reference

```
#include <cms/Config.h> #include <cms/CMSException.h>
```

Data Structures

- class **cms::Stoppable**

Interface for a class that implements the stop method.

Namespaces

- namespace **cms**

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

7.296 src/main/cms/StreamMessage.h File Reference

```
#include <cms/Config.h> #include <cms/Message.h> #include  
<cms/CMSException.h> #include <cms/MessageNotReadable-  
Exception.h> #include <cms/MessageNotWriteableException.-  
h> #include <cms/MessageFormatException.h> #include <cms/-  
MessageEOFException.h>
```

Data Structures

- class **cms::StreamMessage**

*Interface for a **StreamMessage** (p. 2606).*

Namespaces

- namespace **cms**

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

7.297 src/main/cms/TemporaryQueue.h File Reference

```
#include <cms/Config.h> #include <cms/Destination.h> ×  
#include <cms/CMSException.h>
```

Data Structures

- class **cms::TemporaryQueue**

*Defines a Temporary **Queue** (p. 2221) based **Destination** (p. 1210).*

Namespaces

- namespace **cms**

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

7.298 src/main/cms/TemporaryTopic.h File Reference

```
#include <cms/Config.h>      #include <cms/Destination.h> ×  
#include <cms/CMSException.h>
```

Data Structures

- class **cms::TemporaryTopic**

*Defines a Temporary **Topic** (p. 2764) based **Destination** (p. 1210).*

Namespaces

- namespace **cms**

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

7.299 src/main/cms/TextMessage.h File Reference

```
#include <cms/Config.h> #include <cms/Message.h> #include  
<cms/CMSException.h>   #include <cms/MessageNotWriteable-  
Exception.h>
```

Data Structures

- class **cms::TextMessage**

Interface for a text message.

Namespaces

- namespace **cms**

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

7.300 src/main/cms/Topic.h File Reference

```
#include <cms/Config.h>      #include <cms/Destination.h> ×  
#include <cms/CMSException.h>
```

Data Structures

- class **cms::Topic**

An interface encapsulating a provider-specific topic name.

Namespaces

- namespace **cms**

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

7.301 src/main/cms/TransactionInProgressException.h File Reference

```
#include <cms/Config.h> #include <cms/CMSException.h>
```

Data Structures

- class **cms::TransactionInProgressException**

This exception is thrown when an operation is invalid because a transaction is in progress.

Namespaces

- namespace **cms**

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

7.302 src/main/cms/TransactionRolledBackException.h File Reference

```
#include <cms/Config.h> #include <cms/CMSException.h>
```

Data Structures

- class **cms::TransactionRolledBackException**

*This exception must be thrown when a call to **Session.commit** (p. 2366) results in a rollback of the current transaction.*

Namespaces

- namespace **cms**

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

7.303 src/main/cms/UnsupportedOperationException.h File Reference

```
#include <cms/Config.h> #include <cms/CMSException.h>
```

Data Structures

- class **cms::UnsupportedOperationException**

This exception must be thrown when a CMS client attempts use a CMS method that is not implemented or not supported by the CMS Provider in use.

Namespaces

- namespace **cms**

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

7.304 src/main/decaf/lang/exceptions/UnsupportedOperationException.h File Reference

```
#include <decaf/lang/Exception.h>
```

Data Structures

- class **decaf::lang::exceptions::UnsupportedOperationException**

Namespaces

- namespace **decaf**
Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.
- namespace **decaf::lang**
- namespace **decaf::lang::exceptions**

7.305 src/main/cms/XAConnection.h File Reference

```
#include <cms/Config.h>    #include <cms/CMSException.h> ×  
#include <cms/Connection.h> #include <cms/XASession.h>
```

Data Structures

- class **cms::XAConnection**
*The **XAConnection** (p. 2917) interface defines an extended **Connection** (p. 933) type that is used to create **XASession** (p. 2935) objects.*

Namespaces

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

7.306 src/main/cms/XAConnectionFactory.h File Reference

```
#include <cms/Config.h>    #include <cms/XAConnection.h> ×  
#include <cms/XAException.h>
```

Data Structures

- class **cms::XAConnectionFactory**
*The **XAConnectionFactory** (p. 2918) interface is specialized interface that defines an **ConnectionFactory** (p. 955) that creates **Connection** (p. 933) instance that will participate in XA Transactions.*

Namespaces

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

7.307 src/main/cms/XAException.h File Reference

```
#include <cms/Config.h> #include <cms/CMSException.h>
```

Data Structures

- class **cms::XAException**

*The **XAException** (p. 2921) is thrown by the Resource Manager (RM) to inform the Transaction Manager of an error encountered by the involved transaction.*

Namespaces

- namespace **cms**

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

7.308 src/main/cms/XAResource.h File Reference

```
#include <cms/Config.h> #include <cms/Xid.h> #include  
<cms/XAException.h>
```

Data Structures

- class **cms::XAResource**

*The **XAResource** (p. 2926) interface is a CMS mapping of the industry standard XA interface based on the X/Open CAE Specification (Distributed Transaction Processing: The XA Specification).*

Namespaces

- namespace **cms**

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

7.309 src/main/cms/XASession.h File Reference

```
#include <cms/Config.h> #include <cms/Session.h> #include  
<cms/XAResource.h>
```

Data Structures

- class **cms::XASession**

*The **XASession** (p. 2935) interface extends the capability of **Session** (p. 2361) by adding access to a CMS provider's support for the operating inside an XA Transaction (optional).*

Namespaces

- namespace **cms**

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

7.310 src/main/cms/Xid.h File Reference

```
#include <cms/Config.h> #include <cms/CMSException.h>
```

Data Structures

- class **cms::Xid**

An interface which provides a mapping for the X/Open XID transaction identifier structure.

Namespaces

- namespace **cms**

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

7.311 src/main/decaf/internal/AprPool.h File Reference

```
#include <decaf/util/Config.h> #include <apr_pools.h>
```

Data Structures

- class **decaf::internal::AprPool**

Wraps an APR pool object so that classes in decaf can create a static member for use in static methods where apr function calls that need a pool are made.

Namespaces

- namespace **decaf**
Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.
- namespace **decaf::internal**

7.312 src/main/decaf/internal/DecafRuntime.h File Reference

```
#include <decaf/util/Config.h>      #include <decaf/lang/-  
Runtime.h>      #include <decaf/util/concurrent/Mutex.h> ×  
#include <apr_pools.h>
```

Data Structures

- class **decaf::internal::DecafRuntime**
Handles APR initialization and termination.

Namespaces

- namespace **decaf**
Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.
- namespace **decaf::internal**

7.313 src/main/decaf/internal/io/StandardErrorOutputStream.h File Reference

```
#include <decaf/util/Config.h> #include <decaf/io/Output-  
Stream.h>
```

Data Structures

- class **decaf::internal::io::StandardErrorOutputStream**
Wrapper Around the Standard error Output facility on the current platform.

Namespaces

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

- namespace **decaf::internal**
- namespace **decaf::internal::io**

7.314 src/main/decaf/internal/io/StandardInputStream.h File - Reference

```
#include <decaf/util/Config.h> #include <decaf/io/Input-  
Stream.h>
```

Data Structures

- class **decaf::internal::io::StandardInputStream**

Namespaces

- namespace **decaf**
*Licensed to the Apache Software Foundation (ASF) under one or more contributor
license agreements.*
- namespace **decaf::internal**
- namespace **decaf::internal::io**

7.315 src/main/decaf/internal/io/StandardOutputStream.h File - Reference

```
#include <decaf/util/Config.h> #include <decaf/io/Output-  
Stream.h>
```

Data Structures

- class **decaf::internal::io::StandardOutputStream**

Namespaces

- namespace **decaf**
*Licensed to the Apache Software Foundation (ASF) under one or more contributor
license agreements.*
- namespace **decaf::internal**
- namespace **decaf::internal::io**

7.316 src/main/decaf/internal/net/DefaultServerSocketFactory.h File Reference 3127

7.316 src/main/decaf/internal/net/DefaultServerSocketFactory.h - File Reference

```
#include <decaf/util/Config.h>      #include <decaf/net/-  
ServerSocketFactory.h>
```

Data Structures

- class **decaf::internal::net::DefaultServerSocketFactory**
Default implementation of the Decaf ServerSocketFactory, creates ServerSocket objects with supplied options.

Namespaces

- namespace **decaf**
Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.
- namespace **decaf::internal**
- namespace **decaf::internal::net**

7.317 src/main/decaf/internal/net/DefaultSocketFactory.h File - Reference

```
#include <decaf/util/Config.h>      #include <decaf/net/-  
SocketFactory.h>
```

Data Structures

- class **decaf::internal::net::DefaultSocketFactory**
SocketFactory implementation that is used to create Sockets.

Namespaces

- namespace **decaf**
Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.
- namespace **decaf::internal**
- namespace **decaf::internal::net**

7.318 src/main/decaf/internal/net/Network.h File Reference

```
#include <decaf/util/Config.h> #include <decaf/util/concurrent/-  
Mutex.h> #include <decaf/internal/util/Resource.h> #include  
<decaf/internal/util/GenericResource.h>
```

Data Structures

- class **decaf::internal::net::Network**

Internal class used to manage Networking related resources and hide platform dependent calls from the higher level API.

Namespaces

- namespace **decaf**

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

- namespace **decaf::internal**
- namespace **decaf::internal::net**

7.319 src/main/decaf/internal/net/SocketFileDescriptor.h File - Reference

```
#include <decaf/util/Config.h> #include <decaf/io/File-  
Descriptor.h>
```

Data Structures

- class **decaf::internal::net::SocketFileDescriptor**

File Descriptor type used internally by Decaf Socket objects.

Namespaces

- namespace **decaf**

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

- namespace **decaf::internal**
- namespace **decaf::internal::net**

7.320 src/main/decaf/internal/net/ssl/DefaultSSLContext.h File - Reference

```
#include <decaf/util/Config.h> #include <decaf/net/ssl/SSLContext.h>
```

Data Structures

- class **decaf::internal::net::ssl::DefaultSSLContext**

Default SSLContext manager for the Decaf library.

Namespaces

- namespace **decaf**
Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.
- namespace **decaf::internal**
- namespace **decaf::internal::net**
- namespace **decaf::internal::net::ssl**

7.321 src/main/decaf/internal/net/ssl/DefaultSSLServerSocketFactory.h File Reference

```
#include <decaf/util/Config.h> #include <decaf/net/ssl/SSLServerSocketFactory.h>
```

Data Structures

- class **decaf::internal::net::ssl::DefaultSSLServerSocketFactory**

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

Namespaces

- namespace **decaf**
Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.
- namespace **decaf::internal**
- namespace **decaf::internal::net**
- namespace **decaf::internal::net::ssl**

7.322 src/main/decaf/internal/net/ssl/DefaultSSLSocketFactory.h File Reference

```
#include <decaf/util/Config.h> #include <decaf/net/ssl/SSLSocketFactory.h> #include <string> #include <vector>
```

Data Structures

- class **decaf::internal::net::ssl::DefaultSSLSocketFactory**

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

Namespaces

- namespace **decaf**

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

- namespace **decaf::internal**
- namespace **decaf::internal::net**
- namespace **decaf::internal::net::ssl**

7.323 src/main/decaf/internal/net/ssl/openssl/OpenSSLContextSpi.h File Reference

```
#include <decaf/util/Config.h> #include <decaf/net/ssl/SSLContextSpi.h>
```

Data Structures

- class **decaf::internal::net::ssl::openssl::OpenSSLContextSpi**

Provides an SSLContext that wraps the OpenSSL API.

Namespaces

- namespace **decaf**

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

- namespace **decaf::internal**
- namespace **decaf::internal::net**
- namespace **decaf::internal::net::ssl**
- namespace **decaf::internal::net::ssl::openssl**

7.324 src/main/decaf/internal/net/ssl/openssl/OpenSSLParameters.h File Reference

```
#include <decaf/util/Config.h> #include <string> #include  
<vector>
```

Data Structures

- class **decaf::internal::net::ssl::openssl::OpenSSLParameters**
Container class for parameters that are Common to OpenSSL socket classes.

Namespaces

- namespace **decaf**
*Licensed to the Apache Software Foundation (ASF) under one or more contributor
license agreements.*
- namespace **decaf::internal**
- namespace **decaf::internal::net**
- namespace **decaf::internal::net::ssl**
- namespace **decaf::internal::net::ssl::openssl**

7.325 src/main/decaf/internal/net/ssl/openssl/OpenSSLServer- Socket.h File Reference

```
#include <decaf/util/Config.h> #include <decaf/net/ssl/S-  
SLServerSocket.h>
```

Data Structures

- class **decaf::internal::net::ssl::openssl::OpenSSLServerSocket**
SSLServerSocket based on OpenSSL library code.

Namespaces

- namespace **decaf**
*Licensed to the Apache Software Foundation (ASF) under one or more contributor
license agreements.*
- namespace **decaf::internal**
- namespace **decaf::internal::net**
- namespace **decaf::internal::net::ssl**
- namespace **decaf::internal::net::ssl::openssl**

7.326 src/main/decaf/internal/net/ssl/openssl/OpenSSLServerSocketFactory.h File Reference

```
#include <decaf/util/Config.h> #include <decaf/net/ssl/SSLServerSocketFactory.h>
```

Data Structures

- class **decaf::internal::net::ssl::openssl::OpenSSLServerSocketFactory**
SSLServerSocketFactory that creates Server Sockets that use OpenSSL.

Namespaces

- namespace **decaf**
Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.
- namespace **decaf::internal**
- namespace **decaf::internal::net**
- namespace **decaf::internal::net::ssl**
- namespace **decaf::internal::net::ssl::openssl**

7.327 src/main/decaf/internal/net/ssl/openssl/OpenSSLSocket.h - File Reference

```
#include <decaf/util/Config.h> #include <decaf/net/ssl/SSLSocket.h> #include <decaf/io/InputStream.h> #include <decaf/io/OutputStream.h>
```

Data Structures

- class **decaf::internal::net::ssl::openssl::OpenSSLSocket**
Wraps a a Normal Socket object and extends or overrides functions in that class to make use of the OpenSSL Socket API.

Namespaces

- namespace **decaf**
Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.
- namespace **decaf::internal**
- namespace **decaf::internal::net**
- namespace **decaf::internal::net::ssl**
- namespace **decaf::internal::net::ssl::openssl**

7.328 src/main/decaf/internal/net/ssl/openssl/OpenSSLSocketException.h File Reference

```
#include <decaf/util/Config.h>      #include <decaf/net/-  
SocketException.h>
```

Data Structures

- class **decaf::internal::net::ssl::openssl::OpenSSLSocketException**
Subclass of the standard SocketException that knows how to produce an error message from the OpenSSL error stack.

Namespaces

- namespace **decaf**
Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.
- namespace **decaf::internal**
- namespace **decaf::internal::net**
- namespace **decaf::internal::net::ssl**
- namespace **decaf::internal::net::ssl::openssl**

7.329 src/main/decaf/internal/net/ssl/openssl/OpenSSLSocketFactory.h File Reference

```
#include <decaf/util/Config.h> #include <decaf/net/ssl/S-  
SSLSocketFactory.h>
```

Data Structures

- class **decaf::internal::net::ssl::openssl::OpenSSLSocketFactory**
Client Socket Factory that creates SSL based client sockets using the OpenSSL library.

Namespaces

- namespace **decaf**
Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.
- namespace **decaf::internal**
- namespace **decaf::internal::net**
- namespace **decaf::internal::net::ssl**
- namespace **decaf::internal::net::ssl::openssl**

7.330 src/main/decaf/internal/net/ssl/openssl/OpenSSLSocket-InputStream.h File Reference

```
#include <decaf/util/Config.h> #include <decaf/io/Input-Stream.h>
```

Data Structures

- class **decaf::internal::net::ssl::openssl::OpenSSLSocketInputStream**
An output stream for reading data from an OpenSSL Socket instance.

Namespaces

- namespace **decaf**
Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.
- namespace **decaf::internal**
- namespace **decaf::internal::net**
- namespace **decaf::internal::net::ssl**
- namespace **decaf::internal::net::ssl::openssl**

7.331 src/main/decaf/internal/net/ssl/openssl/OpenSSLSocket-OutputStream.h File Reference

```
#include <decaf/util/Config.h> #include <decaf/io/Output-Stream.h>
```

Data Structures

- class **decaf::internal::net::ssl::openssl::OpenSSLSocketOutputStream**
*OutputStream implementation used to write data to an **OpenSSLSocket** (p.2014) instance.*

Namespaces

- namespace **decaf**
Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.
- namespace **decaf::internal**
- namespace **decaf::internal::net**
- namespace **decaf::internal::net::ssl**
- namespace **decaf::internal::net::ssl::openssl**

7.332 src/main/decaf/internal/net/tcp/TcpSocket.h File Reference

```
#include <decaf/net/SocketException.h> #include <decaf/net/-  
SocketImpl.h> #include <decaf/io/InputStream.h> #include  
<decaf/io/OutputStream.h> #include <decaf/util/Config.-  
h> #include <decaf/internal/AprPool.h> #include <apr_-  
network_io.h> #include <decaf/io/IOException.h> #include  
<decaf/net/SocketTimeoutException.h> #include <decaf/lang/exceptions/-  
NullPointerException.h> #include <decaf/lang/exceptions/-  
IndexOutOfBoundsException.h>
```

Data Structures

- class **decaf::internal::net::tcp::TcpSocket**
Platform-independent implementation of the socket interface.

Namespaces

- namespace **decaf**
Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.
- namespace **decaf::internal**
- namespace **decaf::internal::net**
- namespace **decaf::internal::net::tcp**

7.333 src/main/decaf/internal/net/tcp/TcpSocketInputStream.h File Reference

```
#include <decaf/util/Config.h> #include <decaf/io/Input-  
Stream.h> #include <decaf/io/IOException.h> #include  
<decaf/lang/exceptions/NullPointerException.h> #include  
<decaf/lang/exceptions/IndexOutOfBoundsException.h>
```

Data Structures

- class **decaf::internal::net::tcp::TcpSocketInputStream**
Input stream for performing reads on a socket.

Namespaces

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

- namespace **decaf::internal**
- namespace **decaf::internal::net**
- namespace **decaf::internal::net::tcp**

7.334 src/main/decaf/internal/net/tcp/TcpSocketOutputStream.h - File Reference

```
#include <decaf/util/Config.h> #include <decaf/io/OutputStream.h>
```

Data Structures

- class **decaf::internal::net::tcp::TcpSocketOutputStream**
Output stream for performing write operations on a socket.

Namespaces

- namespace **decaf**
Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.
- namespace **decaf::internal**
- namespace **decaf::internal::net**
- namespace **decaf::internal::net::tcp**

7.335 src/main/decaf/internal/net/URLEncoderDecoder.h File - Reference

```
#include <decaf/util/Config.h> #include <decaf/net/URI-SyntaxException.h> #include <string>
```

Data Structures

- class **decaf::internal::net::URLEncoderDecoder**

Namespaces

- namespace **decaf**
Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.
- namespace **decaf::internal**
- namespace **decaf::internal::net**

7.336 src/main/decaf/internal/net/URIHelper.h File Reference

```
#include <string> #include <decaf/util/Config.h> #include  
<decaf/net/URISyntaxException.h> #include <decaf/internal/net/-  
URIType.h>
```

Data Structures

- class **decaf::internal::net::URIHelper**

Helper class used by the URI classes in encoding and decoding of URI's.

Namespaces

- namespace **decaf**
Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.
- namespace **decaf::internal**
- namespace **decaf::internal::net**

7.337 src/main/decaf/internal/net/URIType.h File Reference

```
#include <string> #include <decaf/util/Config.h>
```

Data Structures

- class **decaf::internal::net::URIType**

Basic type object that holds data that composes a given URI.

Namespaces

- namespace **decaf**
Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.
- namespace **decaf::internal**
- namespace **decaf::internal::net**

7.338 src/main/decaf/internal/nio/BufferFactory.h File Reference

```
#include <decaf/nio/ByteBuffer.h> #include <decaf/nio/-  
CharBuffer.h> #include <decaf/nio/DoubleBuffer.h> #include
```

```
<decaf/nio/FloatBuffer.h> #include <decaf/nio/LongBuffer.-
h> #include <decaf/nio/IntBuffer.h> #include <decaf/nio/-
ShortBuffer.h> #include <decaf/lang/exceptions/IndexOut-
OfBoundsException.h>
```

Data Structures

- class **decaf::internal::nio::BufferFactory**

*Factory class used by static methods in the **decaf::nio** (p. 94) package to create the various default version of the NIO interfaces.*

Namespaces

- namespace **decaf**

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

- namespace **decaf::internal**
- namespace **decaf::internal::nio**

7.339 src/main/decaf/internal/nio/ByteArrayBuffer.h File Reference

```
#include <decaf/nio/ByteBuffer.h> #include <decaf/lang/exceptions/-
NullPointerException.h> #include <decaf/lang/exceptions/-
IndexOutOfBoundsException.h> #include <decaf/lang/exceptions/-
IllegalArgumentException.h> #include <decaf/nio/Buffer-
UnderflowException.h> #include <decaf/nio/BufferOverflow-
Exception.h> #include <decaf/nio/ReadOnlyBufferException.-
h> #include <decaf/internal/util/ByteArrayAdapter.h> ×
#include <decaf/nio/CharBuffer.h> #include <decaf/nio/-
DoubleBuffer.h> #include <decaf/nio/FloatBuffer.h> #include
<decaf/nio/ShortBuffer.h> #include <decaf/nio/IntBuffer.-
h> #include <decaf/nio/LongBuffer.h> #include <decaf/lang/-
Pointer.h>
```

Data Structures

- class **decaf::internal::nio::ByteArrayBuffer**

This class defines six categories of operations upon byte buffers:

Namespaces

- namespace **decaf**

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

- namespace **decaf::internal**
- namespace **decaf::internal::nio**

7.340 src/main/decaf/internal/nio/CharArrayBuffer.h File Reference

```
#include <decaf/nio/CharBuffer.h> #include <decaf/lang/exceptions/-  
NullPointerException.h> #include <decaf/lang/exceptions/-  
IndexOutOfBoundsException.h> #include <decaf/nio/Buffer-  
UnderflowException.h> #include <decaf/nio/BufferOverflow-  
Exception.h> #include <decaf/nio/ReadOnlyBufferException.-  
h> #include <decaf/internal/util/ByteArrayAdapter.h> ×  
#include <decaf/lang/Pointer.h>
```

Data Structures

- class **decaf::internal::nio::CharArrayBuffer**

Namespaces

- namespace **decaf**
*Licensed to the Apache Software Foundation (ASF) under one or more contributor
license agreements.*
- namespace **decaf::internal**
- namespace **decaf::internal::nio**

7.341 src/main/decaf/internal/nio/DoubleArrayBuffer.h File Reference

```
#include <decaf/nio/DoubleBuffer.h> #include <decaf/lang/exceptions/-  
NullPointerException.h> #include <decaf/lang/exceptions/-  
IndexOutOfBoundsException.h> #include <decaf/nio/Buffer-  
UnderflowException.h> #include <decaf/nio/BufferOverflow-  
Exception.h> #include <decaf/nio/ReadOnlyBufferException.-  
h> #include <decaf/internal/util/ByteArrayAdapter.h> ×  
#include <decaf/lang/Pointer.h>
```

Data Structures

- class **decaf::internal::nio::DoubleArrayBuffer**

Namespaces

- namespace **decaf**

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

- namespace **decaf::internal**
- namespace **decaf::internal::nio**

7.342 src/main/decaf/internal/nio/FloatArrayBuffer.h File Reference

```
#include <decaf/nio/FloatBuffer.h> #include <decaf/lang/exceptions/-  
NullPointerException.h> #include <decaf/lang/exceptions/-  
IndexOutOfBoundsException.h> #include <decaf/nio/Buffer-  
UnderflowException.h> #include <decaf/nio/BufferOverflow-  
Exception.h> #include <decaf/nio/ReadOnlyBufferException.-  
h> #include <decaf/internal/util/ByteArrayAdapter.h> ×  
#include <decaf/lang/Pointer.h>
```

Data Structures

- class **decaf::internal::nio::FloatArrayBuffer**

Namespaces

- namespace **decaf**

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

- namespace **decaf::internal**
- namespace **decaf::internal::nio**

7.343 src/main/decaf/internal/nio/IntArrayBuffer.h File Reference

```
#include <decaf/nio/IntBuffer.h> #include <decaf/lang/exceptions/-  
NullPointerException.h> #include <decaf/lang/exceptions/-  
IndexOutOfBoundsException.h> #include <decaf/nio/Buffer-  
UnderflowException.h> #include <decaf/nio/BufferOverflow-  
Exception.h> #include <decaf/nio/ReadOnlyBufferException.-  
h> #include <decaf/internal/util/ByteArrayAdapter.h> ×  
#include <decaf/lang/Pointer.h>
```

Data Structures

- class **decaf::internal::nio::IntArrayBuffer**

Namespaces

- namespace **decaf**
Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.
- namespace **decaf::internal**
- namespace **decaf::internal::nio**

7.344 src/main/decaf/internal/nio/LongArrayBuffer.h File Reference

```
#include <decaf/nio/LongBuffer.h> #include <decaf/lang/exceptions/-  
NullPointerException.h> #include <decaf/lang/exceptions/-  
IndexOutOfBoundsException.h> #include <decaf/nio/Buffer-  
UnderflowException.h> #include <decaf/nio/BufferOverflow-  
Exception.h> #include <decaf/nio/ReadOnlyBufferException.-  
h> #include <decaf/internal/util/ByteArrayAdapter.h> ×  
#include <decaf/lang/Pointer.h>
```

Data Structures

- class **decaf::internal::nio::LongArrayBuffer**

Namespaces

- namespace **decaf**
Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.
- namespace **decaf::internal**
- namespace **decaf::internal::nio**

7.345 src/main/decaf/internal/nio/ShortArrayBuffer.h File Reference

```
#include <decaf/nio/ShortBuffer.h> #include <decaf/lang/exceptions/-  
NullPointerException.h> #include <decaf/lang/exceptions/-  
IndexOutOfBoundsException.h> #include <decaf/nio/Buffer-  
UnderflowException.h> #include <decaf/nio/BufferOverflow-  
Exception.h> #include <decaf/nio/ReadOnlyBufferException.-  
h> #include <decaf/internal/util/ByteArrayAdapter.h> ×  
#include <decaf/lang/Pointer.h>
```

Data Structures

- class **decaf::internal::nio::ShortArrayBuffer**

Namespaces

- namespace **decaf**
Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.
- namespace **decaf::internal**
- namespace **decaf::internal::nio**

7.346 src/main/decaf/internal/security/unix/SecureRandomImpl.h - File Reference

```
#include <decaf/util/Config.h> #include <decaf/security/-  
SecureRandomSpi.h>
```

Data Structures

- class **decaf::internal::security::SecureRandomImpl**
Secure Random Number Generator for Unix based platforms that attempts to obtain secure bytes with high entropy from known sources.

Namespaces

- namespace **decaf**
Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.
- namespace **decaf::internal**
- namespace **decaf::internal::security**

7.347 src/main/decaf/internal/security/windows/SecureRandom-Impl.h File Reference

```
#include <decaf/util/Config.h> #include <decaf/security/-  
SecureRandomSpi.h>
```

Data Structures

- class **decaf::internal::security::SecureRandomImpl**
Secure Random Number Generator for Unix based platforms that attempts to obtain secure bytes with high entropy from known sources.

Namespaces

- namespace **decaf**
Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.
- namespace **decaf::internal**
- namespace **decaf::internal::security**

7.348 src/main/decaf/internal/util/ByteArrayAdapter.h File Reference

```
#include <decaf/lang/exceptions/InvalidStateException.-  
h> #include <decaf/lang/exceptions/IllegalArgumentException.-  
h> #include <decaf/lang/exceptions/IndexOutOfBoundsException.-  
h> #include <decaf/lang/exceptions/NullPointerException.-  
h> #include <decaf/nio/BufferUnderflowException.h> #include  
<decaf/nio/BufferOverflowException.h>
```

Data Structures

- class **decaf::internal::util::ByteArrayAdapter**
This class adapts primitive type arrays to a base byte array so that the classes can inter-operate on the same base byte array without copying data.
- union **decaf::internal::util::ByteArrayAdapter::Array**

Namespaces

- namespace **decaf**
Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.
- namespace **decaf::internal**
- namespace **decaf::internal::util**

7.349 src/main/decaf/internal/util/concurrent/ConditionImpl.h File - Reference

```
#include <decaf/util/Config.h>
```

Data Structures

- class **decaf::internal::util::concurrent::ConditionImpl**

Namespaces

- namespace **decaf**
Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.
- namespace **decaf::util**
- namespace **decaf::util::concurrent**
- namespace **decaf::internal**
- namespace **decaf::internal::util**
- namespace **decaf::internal::util::concurrent**

7.350 src/main/decaf/internal/util/concurrent/MutexImpl.h File - Reference

```
#include <decaf/util/Config.h>
```

Data Structures

- class **decaf::internal::util::concurrent::MutexImpl**

Namespaces

- namespace **decaf**
Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.
- namespace **decaf::util**
- namespace **decaf::util::concurrent**
- namespace **decaf::internal**
- namespace **decaf::internal::util**
- namespace **decaf::internal::util::concurrent**

7.351 src/main/decaf/internal/util/concurrent/SynchronizableImpl.h File Reference

```
#include <decaf/util/Config.h> #include <decaf/util/concurrent/-  
Synchronizable.h> #include <decaf/util/concurrent/Mutex.-  
h>
```

Data Structures

- class **decaf::internal::util::concurrent::SynchronizableImpl**
A convenience class used by some Decaf classes to implement the Synchronizable interface when there is no issues related to multiple inheritance.

Namespaces

- namespace **decaf**
Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.
- namespace **decaf::internal**
- namespace **decaf::internal::util**
- namespace **decaf::internal::util::concurrent**

7.352 src/main/decaf/internal/util/concurrent/Transferer.h File Reference

```
#include <decaf/lang/exceptions/InterruptedException.h>
#include <decaf/util/concurrent/TimeoutException.h>
```

Data Structures

- class **decaf::internal::util::concurrent::Transferer< E >**
Shared internal API for dual stacks and queues.

Namespaces

- namespace **decaf**
Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.
- namespace **decaf::internal**
- namespace **decaf::internal::util**
- namespace **decaf::internal::util::concurrent**

7.353 src/main/decaf/internal/util/concurrent/TransferQueue.h File Reference

```
#include <decaf/internal/util/concurrent/Transferer.h> ×
#include <decaf/util/concurrent/locks/LockSupport.h> ×
#include <decaf/util/concurrent/atomic/AtomicReference.h> ×
#include <decaf/util/concurrent/TimeoutException.h> ×
#include <decaf/lang/exceptions/InterruptedException.h>
#include <decaf/lang/Thread.h>
```

Data Structures

- class **decaf::internal::util::concurrent::TransferQueue**< E >

This extends Scherer-Scott dual queue algorithm, differing, among other ways, by using modes within nodes rather than marked pointers.

Namespaces

- namespace **decaf**

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

- namespace **decaf::internal**
- namespace **decaf::internal::util**
- namespace **decaf::internal::util::concurrent**

7.354 src/main/decaf/internal/util/concurrent/TransferStack.h File - Reference

```
#include <decaf/internal/util/concurrent/Transferer.-
h> #include <decaf/util/concurrent/TimeoutException.h> ×
#include <decaf/lang/exceptions/InterruptedException.h>
```

Data Structures

- class **decaf::internal::util::concurrent::TransferStack**< E >

Namespaces

- namespace **decaf**

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

- namespace **decaf::internal**
- namespace **decaf::internal::util**
- namespace **decaf::internal::util::concurrent**

7.355 src/main/decaf/internal/util/concurrent/unix/ConditionHandle.h File Reference

```
#include <decaf/util/Config.h>
```


Data Structures

- class **decaf::util::concurrent::ConditionHandle**

Namespaces

- namespace **decaf**
Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.
- namespace **decaf::util**
- namespace **decaf::util::concurrent**

7.356 src/main/decaf/internal/util/concurrent/windows/ConditionHandle.h File Reference

```
#include <decaf/util/Config.h>
```

Data Structures

- class **decaf::util::concurrent::ConditionHandle**

Namespaces

- namespace **decaf**
Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.
- namespace **decaf::util**
- namespace **decaf::util::concurrent**

7.357 src/main/decaf/internal/util/concurrent/unix/MutexHandle.h - File Reference

```
#include <decaf/util/Config.h>
```

Data Structures

- class **decaf::util::concurrent::MutexHandle**

Namespaces

- namespace **decaf**
Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.
- namespace **decaf::util**
- namespace **decaf::util::concurrent**

7.358 src/main/decaf/internal/util/concurrent/windows/Mutex-Handle.h File Reference

```
#include <decaf/util/Config.h>
```

Data Structures

- class **decaf::util::concurrent::MutexHandle**

Namespaces

- namespace **decaf**
Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.
- namespace **decaf::util**
- namespace **decaf::util::concurrent**

7.359 src/main/decaf/internal/util/GenericResource.h File Reference

```
#include <decaf/util/Config.h> #include <decaf/internal/util/-Resource.h>
```

Data Structures

- class **decaf::internal::util::GenericResource< T >**
*A Generic **Resource** (p. 2293) wraps some type and will delete it when the **Resource** (p. 2293) itself is deleted.*

Namespaces

- namespace **decaf**
Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.
- namespace **decaf::internal**
- namespace **decaf::internal::util**

7.360 src/main/decaf/internal/util/HexStringParser.h File Reference

```
#include <decaf/util/Config.h> #include <string>
```

Data Structures

- class **decaf::internal::util::HexStringParser**

Namespaces

- namespace **decaf**
Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.
- namespace **decaf::internal**
- namespace **decaf::internal::util**

7.361 src/main/decaf/internal/util/Resource.h File Reference

```
#include <decaf/util/Config.h>
```

Data Structures

- class **decaf::internal::util::Resource**
Interface for all Managed Resources in Decaf, these objects are added to the Runtime System and are destroyed at shutdown.

Namespaces

- namespace **decaf**
Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.
- namespace **decaf::internal**
- namespace **decaf::internal::util**

7.362 src/main/decaf/internal/util/TimerTaskHeap.h File Reference

```
#include <decaf/util/Config.h> #include <decaf/util/-  
TimerTask.h> #include <decaf/lang/Pointer.h>
```

Data Structures

- class **decaf::internal::util::TimerTaskHeap**

A Binary Heap implemented specifically for the Timer class in Decaf Util.

Namespaces

- namespace **decaf**

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

- namespace **decaf::internal**
- namespace **decaf::internal::util**

7.363 src/main/decaf/internal/util/zip/crc32.h File Reference

Variables

- **local** const unsigned long FAR **crc_table** [TBLS][256]

7.363.1 Variable Documentation

7.363.1.1 **local** const unsigned long FAR **crc_table**[TBLS][256]

7.364 src/main/decaf/internal/util/zip/deflate.h File Reference

```
#include "zutil.h"
```

Data Structures

- struct **ct_data_s**
- struct **tree_desc_s**
- struct **internal_state**

Defines

- #define **GZIP**
- #define **LENGTH_CODES** 29
- #define **LITERALS** 256
- #define **L_CODES** (LITERALS+1+LENGTH_CODES)
- #define **D_CODES** 30
- #define **BL_CODES** 19
- #define **HEAP_SIZE** (2*L_CODES+1)

- #define **MAX_BITS** 15
- #define **INIT_STATE** 42
- #define **EXTRA_STATE** 69
- #define **NAME_STATE** 73
- #define **COMMENT_STATE** 91
- #define **HCRC_STATE** 103
- #define **BUSY_STATE** 113
- #define **FINISH_STATE** 666
- #define **Freq** fc.freq
- #define **Code** fc.code
- #define **Dad** dl.dad
- #define **Len** dl.len
- #define **max_insert_length** max_lazy_match
- #define **put_byte**(s, c) {s->pending_buf[s->pending++] = (c);}
- #define **MIN_LOOKAHEAD** (**MAX_MATCH**+**MIN_MATCH**+1)
- #define **MAX_DIST**(s) ((s)->w_size-**MIN_LOOKAHEAD**)
- #define **WIN_INIT** **MAX_MATCH**
- #define **d_code**(dist) ((dist) < 256 ? **_dist_code**[dist] : **_dist_code**[256+((dist)>>7)])
- #define **_tr_tally_lit**(s, c, flush)
- #define **_tr_tally_dist**(s, distance, length, flush)

Typedefs

- typedef struct **ct_data_s** **ct_data**
- typedef struct static_tree_desc_s **static_tree_desc**
- typedef struct **tree_desc_s** **tree_desc**
- typedef **ush** **Pos**
- typedef **Pos** FAR **Posf**
- typedef unsigned **IPos**
- typedef struct **internal_state** **deflate_state**

Functions

- void **ZLIB_INTERNAL** **_tr_init** OF ((**deflate_state** *s))
- int **ZLIB_INTERNAL** **_tr_tally** OF ((**deflate_state** *s, unsigned dist, unsigned lc))
- void **ZLIB_INTERNAL** **_tr_flush_block** OF ((**deflate_state** *s, **charf** *buf, **ulg** stored_len, int last))

Variables

- **uch** **ZLIB_INTERNAL** **_length_code** []
- **uch** **ZLIB_INTERNAL** **_dist_code** []

7.364.1 Define Documentation

7.364.1.1 `#define _tr_tally_dist(s, distance, length, flush)`

Value:

```
{ uch len = (length); \
  ush dist = (distance); \
  s->d_buf[s->last_lit] = dist; \
  s->l_buf[s->last_lit++] = len; \
  dist--; \
  s->dyn_ltree[_length_code[len]+LITERALS+1].Freq++; \
  s->dyn_dtree[d_code(dist)].Freq++; \
  flush = (s->last_lit == s->lit_bufsize-1); \
}
```

7.364.1.2 `#define _tr_tally_lit(s, c, flush)`

Value:

```
{ uch cc = (c); \
  s->d_buf[s->last_lit] = 0; \
  s->l_buf[s->last_lit++] = cc; \
  s->dyn_ltree[cc].Freq++; \
  flush = (s->last_lit == s->lit_bufsize-1); \
}
```

7.364.1.3 `#define BL_CODES 19`

7.364.1.4 `#define BUSY_STATE 113`

7.364.1.5 `#define Code fc.code`

7.364.1.6 `#define COMMENT_STATE 91`

7.364.1.7 `#define d_code(dist) ((dist) < 256 ? _dist_code[dist] : _dist_code[256+((dist)>>7)])`

7.364.1.8 `#define D_CODES 30`

7.364.1.9 `#define Dad dl.dad`

7.364.1.10 `#define EXTRA_STATE 69`

7.364.1.11 `#define FINISH_STATE 666`

7.364.1.12 `#define Freq fc.freq`

7.364.1.13 `#define GZIP`

- 7.364.1.14 `#define HCRC_STATE 103`
- 7.364.1.15 `#define HEAP_SIZE (2*L_CODES+1)`
- 7.364.1.16 `#define INIT_STATE 42`
- 7.364.1.17 `#define L_CODES (LITERALS+1+LENGTH_CODES)`
- 7.364.1.18 `#define Len dl.len`
- 7.364.1.19 `#define LENGTH_CODES 29`
- 7.364.1.20 `#define LITERALS 256`
- 7.364.1.21 `#define MAX_BITS 15`
- 7.364.1.22 `#define MAX_DIST(s) ((s)->w_size-MIN_LOOKAHEAD)`
- 7.364.1.23 `#define max_insert_length max_lazy_match`
- 7.364.1.24 `#define MIN_LOOKAHEAD (MAX_MATCH+MIN_MATCH+1)`
- 7.364.1.25 `#define NAME_STATE 73`
- 7.364.1.26 `#define put_byte(s, c) { s->pending_buf[s->pending++] = (c); }`
- 7.364.1.27 `#define WIN_INIT MAX_MATCH`

7.364.2 Typedef Documentation

- 7.364.2.1 `typedef struct ct_data_s ct_data`
- 7.364.2.2 `typedef struct internal_state deflate_state`
- 7.364.2.3 `typedef unsigned IPos`
- 7.364.2.4 `typedef ush Pos`
- 7.364.2.5 `typedef Pos FAR Posf`
- 7.364.2.6 `typedef struct static_tree_desc_s static_tree_desc`
- 7.364.2.7 `typedef struct tree_desc_s tree_desc`

7.364.3 Function Documentation

- 7.364.3.1 `void ZLIB_INTERNAL _tr_init OF ((deflate_state *s))`

7.364.3.2 int ZLIB_INTERNAL _tr_tally OF ((deflate_state *s, unsigned dist, unsigned lc)
)

7.364.3.3 void ZLIB_INTERNAL _tr_flush_block OF ((deflate_state *s, charf *buf, ulg
 stored_len, int last))

7.364.4 Variable Documentation

7.364.4.1 uch ZLIB_INTERNAL _dist_code[]

7.364.4.2 uch ZLIB_INTERNAL _length_code[]

7.365 src/main/decaf/internal/util/zip/gzguts.h File Reference

```
#include <stdio.h> #include "zlib.h" #include <fcntl.h>
```

Data Structures

- struct **gz_state**

Defines

- #define **ZLIB_INTERNAL**
- #define **local** static
- #define **zstrerror()** "stdio error (consult errno)"
- #define **GZBUFSIZE** 8192
- #define **GZ_NONE** 0
- #define **GZ_READ** 7247
- #define **GZ_WRITE** 31153
- #define **GZ_APPEND** 1 /* mode set to **GZ_WRITE** after the file is opened */
- #define **LOOK** 0 /* look for a gzip header */
- #define **COPY** 1 /* copy input directly */
- #define **GZIP** 2 /* decompress a gzip stream */
- #define **GT_OFF**(x) (sizeof(int) == sizeof(**z_off64_t**) && (x) > gz_intmax())

Typedefs

- typedef **gz_state** FAR * **gz_statep**

Functions

- **voidp** malloc OF ((**uInt** size))
- void free OF ((**voidpf** ptr))
- ZEXTERN **gzFile** ZEXPORT gzopen64 OF ((const char *, const char *))

- ZEXTERN **z_off64_t** ZEXPORT gzseek64 **OF** ((**gzFile**, **z_off64_t**, int))
- ZEXTERN **z_off64_t** ZEXPORT gztell64 **OF** ((**gzFile**))
- void **ZLIB_INTERNAL** gz_error **OF** ((**gz_statep**, int, const char *))
- unsigned **ZLIB_INTERNAL** gz_intmax **OF** ((void))

7.365.1 Define Documentation

7.365.1.1 **#define COPY 1** /* copy input directly */

7.365.1.2 **#define GT_OFF(x)** (sizeof(int) == sizeof(**z_off64_t**) && (x) > gz_intmax())

7.365.1.3 **#define GZ_APPEND 1** /* mode set to **GZ_WRITE** after the file is opened */

7.365.1.4 **#define GZ_NONE 0**

7.365.1.5 **#define GZ_READ 7247**

7.365.1.6 **#define GZ_WRITE 31153**

7.365.1.7 **#define GZBUFSIZE 8192**

7.365.1.8 **#define GZIP 2** /* decompress a gzip stream */

7.365.1.9 **#define local static**

7.365.1.10 **#define LOOK 0** /* look for a gzip header */

7.365.1.11 **#define ZLIB_INTERNAL**

7.365.1.12 **#define zstrerror()** "stdio error (consult errno)"

7.365.2 Typedef Documentation

7.365.2.1 **typedef gz_state FAR* gz_statep**

7.365.3 Function Documentation

7.365.3.1 **voidp malloc OF ((uInt size))**

7.365.3.2 **void free OF ((voidpf ptr))**

7.365.3.3 **ZEXTERN gzFile ZEXPORT gzopen64 OF ((const char *, const char *))**

7.365.3.4 **ZEXTERN z_off64_t ZEXPORT gzseek64 OF ((gzFile, z_off64_t, int))**

7.365.3.5 **ZEXTERN z_off64_t ZEXPORT gztell64 OF ((gzFile))**

7.365.3.6 void ZLIB_INTERNAL gz_error OF ((gz_statep, int, const char *))

7.365.3.7 unsigned ZLIB_INTERNAL gz_intmax OF ((void))

7.366 src/main/decaf/internal/util/zip/inffast.h File Reference

Functions

- void ZLIB_INTERNAL inflate_fast OF ((z_streamp strm, unsigned start))

7.366.1 Function Documentation

7.366.1.1 void ZLIB_INTERNAL inflate_fast OF ((z_streamp strm, unsigned start))

7.367 src/main/decaf/internal/util/zip/inffixed.h File Reference

7.368 src/main/decaf/internal/util/zip/inflate.h File Reference

Data Structures

- struct inflate_state

Defines

- #define GUNZIP

Enumerations

- enum inflate_mode { HEAD, FLAGS, TIME, OS, EXLEN, EXTRA, NAME, COMMENT, HCRC, DICTID, DICT, TYPE, TYPEDO, STORED, COPY_, COPY, TABLE, LENLENS, CODELENS, LEN_, LEN, LENEXT, DIST, DISTEXT, MATCH, LIT, CHECK, LENGTH, DONE, BAD, MEM, SYNC }

7.368.1 Define Documentation

7.368.1.1 #define GUNZIP

7.368.2 Enumeration Type Documentation

7.368.2.1 enum inflate_mode

Enumerator:

HEAD

FLAGS
TIME
OS
EXLEN
EXTRA
NAME
COMMENT
HCRC
DICTID
DICT
TYPE
TYPEDO
STORED
COPY_
COPY
TABLE
LENLENS
CODELENS
LEN_
LEN
LENEXT
DIST
DISTEXT
MATCH
LIT
CHECK
LENGTH
DONE
BAD
MEM
SYNC

7.369 src/main/decaf/internal/util/zip/inftrees.h File Reference

Data Structures

- struct **code**

Defines

- `#define ENOUGH_LENS 852`
- `#define ENOUGH_DISTS 592`
- `#define ENOUGH (ENOUGH_LENS+ENOUGH_DISTS)`

Enumerations

- `enum codetype { CODES, LENS, DISTS }`

Functions

- `int ZLIB_INTERNAL inflate_table OF ((codetype type, unsigned short FAR *lens, unsigned codes, code FAR *FAR *table, unsigned FAR *bits, unsigned short FAR *work))`

7.369.1 Define Documentation

7.369.1.1 `#define ENOUGH (ENOUGH_LENS+ENOUGH_DISTS)`

7.369.1.2 `#define ENOUGH_DISTS 592`

7.369.1.3 `#define ENOUGH_LENS 852`

7.369.2 Enumeration Type Documentation

7.369.2.1 `enum codetype`

Enumerator:

CODES

LENS

DISTS

7.369.3 Function Documentation

7.369.3.1 `int ZLIB_INTERNAL inflate_table OF ((codetype type, unsigned short FAR *lens, unsigned codes, code FAR *FAR *table, unsigned FAR *bits, unsigned short FAR *work))`

7.370 src/main/decaf/internal/util/zip/trees.h File Reference

Variables

- `local const ct_data static_ltree [L_CODES+2]`

- **local** const **ct_data** **static_dtree** [D_CODES]
- const **uch** **ZLIB_INTERNAL_dist_code** [DIST_CODE_LEN]
- const **uch** **ZLIB_INTERNAL_length_code** [MAX_MATCH-MIN_MATCH+1]
- **local** const int **base_length** [LENGTH_CODES]
- **local** const int **base_dist** [D_CODES]

7.370.1 Variable Documentation

7.370.1.1 const uch ZLIB_INTERNAL_dist_code[DIST_CODE_LEN]

Initial value:

```
{
  0,  1,  2,  3,  4,  4,  5,  5,  6,  6,  6,  6,  7,  7,  7,  7,  8,  8,  8,  8,
  8,  8,  8,  8,  9,  9,  9,  9,  9,  9,  9,  9, 10, 10, 10, 10, 10, 10, 10, 10,
10, 10, 10, 10, 10, 10, 10, 10, 10, 11, 11, 11, 11, 11, 11, 11, 11, 11, 11, 11, 11,
11, 11, 11, 11, 12, 12, 12, 12, 12, 12, 12, 12, 12, 12, 12, 12, 12, 12, 12, 12,
12, 12, 12, 12, 12, 12, 12, 12, 12, 12, 12, 12, 12, 12, 12, 12, 13, 13, 13, 13,
13, 13, 13, 13, 13, 13, 13, 13, 13, 13, 13, 13, 13, 13, 13, 13, 13, 13, 13, 13,
13, 13, 13, 13, 13, 13, 13, 13, 13, 14, 14, 14, 14, 14, 14, 14, 14, 14, 14, 14,
14, 14, 14, 14, 14, 14, 14, 14, 14, 14, 14, 14, 14, 14, 14, 14, 14, 14, 14, 14,
14, 14, 14, 14, 14, 14, 14, 14, 14, 14, 14, 14, 14, 14, 14, 14, 14, 14, 14, 14,
14, 14, 14, 14, 14, 14, 14, 14, 14, 14, 14, 14, 14, 14, 14, 14, 14, 14, 14, 14,
15, 15, 15, 15, 15, 15, 15, 15, 15, 15, 15, 15, 15, 15, 15, 15, 15, 15, 15, 15,
15, 15, 15, 15, 15, 15, 15, 15, 15, 15, 15, 15, 15, 15, 15, 15, 15, 15, 15, 15,
15, 15, 15, 15, 15, 15, 15, 15, 15, 15, 15, 15, 15, 15, 15, 15, 15, 15, 15, 15,
15, 15, 15, 15, 15, 15, 15, 15, 15, 15, 15, 15, 15, 15, 15, 15, 15, 15, 15, 15,
18, 18, 19, 19, 20, 20, 20, 20, 20, 21, 21, 21, 21, 21, 22, 22, 22, 22, 22, 22, 22,
22, 22, 22, 22, 22, 22, 22, 22, 22, 22, 22, 22, 22, 22, 22, 22, 22, 22, 22, 22,
23, 23, 23, 23, 23, 23, 23, 23, 23, 23, 24, 24, 24, 24, 24, 24, 24, 24, 24, 24,
24, 24, 24, 24, 24, 25, 25, 25, 25, 25, 25, 25, 25, 25, 25, 25, 25, 25, 25, 25,
26, 26, 26, 26, 26, 26, 26, 26, 26, 26, 26, 26, 26, 26, 26, 26, 26, 26, 26, 26,
26, 26, 26, 26, 26, 26, 26, 26, 26, 26, 26, 26, 26, 26, 26, 26, 26, 26, 26, 26,
27, 27, 27, 27, 27, 27, 27, 27, 27, 27, 27, 27, 27, 27, 27, 27, 27, 27, 27, 27,
27, 27, 27, 27, 27, 27, 27, 27, 27, 27, 27, 27, 27, 27, 27, 27, 27, 27, 27, 27,
27, 27, 27, 27, 27, 27, 27, 27, 27, 27, 27, 27, 27, 27, 27, 27, 27, 27, 27, 27,
28, 28, 28, 28, 28, 28, 28, 28, 28, 28, 28, 28, 28, 28, 28, 28, 28, 28, 28, 28,
28, 28, 28, 28, 28, 28, 28, 28, 28, 28, 28, 28, 28, 28, 28, 28, 28, 28, 28, 28,
28, 28, 28, 28, 28, 28, 28, 28, 28, 28, 28, 28, 28, 28, 28, 28, 28, 28, 28, 28,
28, 28, 28, 28, 28, 28, 28, 28, 28, 28, 28, 28, 28, 28, 28, 28, 28, 28, 28, 28,
29, 29, 29, 29, 29, 29, 29, 29, 29, 29, 29, 29, 29, 29, 29, 29, 29, 29, 29, 29,
29, 29, 29, 29, 29, 29, 29, 29, 29, 29, 29, 29, 29, 29, 29, 29, 29, 29, 29, 29,
29, 29, 29, 29, 29, 29, 29, 29, 29, 29, 29, 29, 29, 29, 29, 29, 29, 29, 29, 29,
29, 29, 29, 29, 29, 29, 29, 29, 29, 29, 29, 29, 29, 29, 29, 29, 29, 29, 29, 29,
}
}
```

7.370.1.2 const uch ZLIB_INTERNAL_length_code[MAX_MATCH-MIN_MATCH+1]

Initial value:

```
{
  0,  1,  2,  3,  4,  5,  6,  7,  8,  8,  9,  9, 10, 10, 11, 11, 12, 12, 12, 12,
13, 13, 13, 13, 14, 14, 14, 14, 15, 15, 15, 15, 16, 16, 16, 16, 16, 16, 16, 16,
17, 17, 17, 17, 17, 17, 17, 17, 18, 18, 18, 18, 18, 18, 18, 18, 19, 19, 19, 19,
19, 19, 19, 19, 20, 20, 20, 20, 20, 20, 20, 20, 20, 20, 20, 20, 20, 20, 20, 20,
21, 21, 21, 21, 21, 21, 21, 21, 21, 21, 21, 21, 21, 21, 21, 21, 22, 22, 22, 22,
22, 22, 22, 22, 22, 22, 22, 22, 22, 22, 22, 22, 22, 22, 22, 22, 23, 23, 23, 23,
23, 23, 23, 23, 23, 23, 23, 23, 24, 24, 24, 24, 24, 24, 24, 24, 24, 24, 24, 24,
24, 24, 24, 24, 24, 24, 24, 24, 24, 24, 24, 24, 24, 24, 24, 24, 24, 24, 24, 24,
25, 25, 25, 25, 25, 25, 25, 25, 25, 25, 25, 25, 25, 25, 25, 25, 25, 25, 25, 25,
}
}
```

```

25, 25, 25, 25, 25, 25, 25, 25, 25, 25, 25, 25, 26, 26, 26, 26, 26, 26, 26, 26,
26, 26, 26, 26, 26, 26, 26, 26, 26, 26, 26, 26, 26, 26, 26, 26, 26, 26, 26,
26, 26, 26, 26, 27, 27, 27, 27, 27, 27, 27, 27, 27, 27, 27, 27, 27, 27, 27,
27, 27, 27, 27, 27, 27, 27, 27, 27, 27, 27, 27, 27, 27, 27, 27, 28
}

```

7.370.1.3 local const int base_dist[D_CODES]

Initial value:

```

{
    0,      1,      2,      3,      4,      6,      8,      12,      16,      24,
    32,     48,     64,     96,    128,    192,    256,    384,    512,    768,
 1024,  1536,  2048,  3072,  4096,  6144,  8192, 12288, 16384, 24576
}

```

7.370.1.4 local const int base_length[LENGTH_CODES]

Initial value:

```

{
0, 1, 2, 3, 4, 5, 6, 7, 8, 10, 12, 14, 16, 20, 24, 28, 32, 40, 48, 56,
64, 80, 96, 112, 128, 160, 192, 224, 0
}

```

7.370.1.5 local const ct_data static_dtree[D_CODES]

Initial value:

```

{
{{ 0},{ 5}}, {{16},{ 5}}, {{ 8},{ 5}}, {{24},{ 5}}, {{ 4},{ 5}},
{{20},{ 5}}, {{12},{ 5}}, {{28},{ 5}}, {{ 2},{ 5}}, {{18},{ 5}},
{{10},{ 5}}, {{26},{ 5}}, {{ 6},{ 5}}, {{22},{ 5}}, {{14},{ 5}},
{{30},{ 5}}, {{ 1},{ 5}}, {{17},{ 5}}, {{ 9},{ 5}}, {{25},{ 5}},
{{ 5},{ 5}}, {{21},{ 5}}, {{13},{ 5}}, {{29},{ 5}}, {{ 3},{ 5}},
{{19},{ 5}}, {{11},{ 5}}, {{27},{ 5}}, {{ 7},{ 5}}, {{23},{ 5}}
}

```

7.370.1.6 local const ct_data static_ltree[L_CODES+2]

7.371 src/main/decaf/internal/util/zip/zconf.h File Reference

```
#include <decaf/util/Config.h>
```

Defines

- `#define z_off64_t z_off_t`

Typedefs

- typedef unsigned char **Byte**
- typedef unsigned int **ulnt**
- typedef unsigned long **uLong**
- typedef **Byte** FAR **Bytef**
- typedef char FAR **charf**
- typedef int FAR **intf**
- typedef **ulnt** FAR **ulntf**
- typedef **uLong** FAR **uLongf**
- typedef **Byte** const * **voidpc**
- typedef **Byte** FAR * **voidpf**
- typedef **Byte** * **voidp**

7.371.1 Define Documentation

7.371.1.1 `#define z_off64_t z_off_t`

7.371.2 Typedef Documentation

7.371.2.1 typedef unsigned char **Byte**

7.371.2.2 typedef **Byte** FAR **Bytef**

7.371.2.3 typedef char FAR **charf**

7.371.2.4 typedef int FAR **intf**

7.371.2.5 typedef unsigned int **ulnt**

7.371.2.6 typedef **ulnt** FAR **ulntf**

7.371.2.7 typedef unsigned long **uLong**

7.371.2.8 typedef **uLong** FAR **uLongf**

7.371.2.9 typedef **Byte*** **voidp**

7.371.2.10 typedef **Byte** const* **voidpc**

7.371.2.11 typedef **Byte** FAR* **voidpf**

7.372 src/main/decaf/internal/util/zip/zlib.h File Reference

```
#include "zconf.h"
```

Data Structures

- struct **z_stream_s**
- struct **gz_header_s**
- struct **internal_state**

Defines

- #define **ZLIB_VERSION** "1.2.5"
- #define **ZLIB_VERNUM** 0x1250
- #define **ZLIB_VER_MAJOR** 1
- #define **ZLIB_VER_MINOR** 2
- #define **ZLIB_VER_REVISION** 5
- #define **ZLIB_VER_SUBREVISION** 0
- #define **Z_NO_FLUSH** 0
- #define **Z_PARTIAL_FLUSH** 1
- #define **Z_SYNC_FLUSH** 2
- #define **Z_FULL_FLUSH** 3
- #define **Z_FINISH** 4
- #define **Z_BLOCK** 5
- #define **Z_TREES** 6
- #define **Z_OK** 0
- #define **Z_STREAM_END** 1
- #define **Z_NEED_DICT** 2
- #define **Z_ERRNO** (-1)
- #define **Z_STREAM_ERROR** (-2)
- #define **Z_DATA_ERROR** (-3)
- #define **Z_MEM_ERROR** (-4)
- #define **Z_BUF_ERROR** (-5)
- #define **Z_VERSION_ERROR** (-6)
- #define **Z_NO_COMPRESSION** 0
- #define **Z_BEST_SPEED** 1
- #define **Z_BEST_COMPRESSION** 9
- #define **Z_DEFAULT_COMPRESSION** (-1)
- #define **Z_FILTERED** 1
- #define **Z_HUFFMAN_ONLY** 2
- #define **Z_RLE** 3
- #define **Z_FIXED** 4
- #define **Z_DEFAULT_STRATEGY** 0
- #define **Z_BINARY** 0
- #define **Z_TEXT** 1
- #define **Z_ASCII** **Z_TEXT** /* for compatibility with 1.2.2 and earlier */
- #define **Z_UNKNOWN** 2
- #define **Z_DEFLATED** 8
- #define **Z_NULL** 0 /* for initializing zalloc, zfree, opaque */
- #define **zlib_version** zlibVersion()

- #define **deflateInit**(strm, level) deflateInit_((strm), (level), **ZLIB_VERSION**, sizeof(**z_stream**))
- #define **inflateInit**(strm) inflateInit_((strm), **ZLIB_VERSION**, sizeof(**z_stream**))
- #define **deflateInit2**(strm, level, method, windowBits, memLevel, strategy)
- #define **inflateInit2**(strm, windowBits) inflateInit2_((strm), (windowBits), **ZLIB_VERSION**, sizeof(**z_stream**))
- #define **inflateBackInit**(strm, windowBits, window)

Typedefs

- typedef **voidpf** alloc_func **OF** ((**voidpf** opaque, **uInt** items, **uInt** size))
- typedef struct **z_stream_s** **z_stream**
- typedef **z_stream** FAR * **z_streamp**
- typedef struct **gz_header_s** **gz_header**
- typedef **gz_header** FAR * **gz_headerp**
- typedef **voidp** **gzFile**

Functions

- ZEXTERN const char *ZEXPORT zlibVersion **OF** ((void))
- ZEXTERN int ZEXPORT deflate **OF** ((**z_streamp** strm, int flush))
- ZEXTERN int ZEXPORT deflateEnd **OF** ((**z_streamp** strm))
- ZEXTERN int ZEXPORT deflateSetDictionary **OF** ((**z_streamp** strm, const **Bytef** *dictionary, **uInt**dictLength))
- ZEXTERN int ZEXPORT deflateCopy **OF** ((**z_streamp** dest, **z_streamp** source))
- ZEXTERN int ZEXPORT deflateParams **OF** ((**z_streamp** strm, int level, int strategy))
- ZEXTERN int ZEXPORT deflateTune **OF** ((**z_streamp** strm, int good_length, int max_lazy, int nice_length, int max_chain))
- ZEXTERN **uLong** ZEXPORT deflateBound **OF** ((**z_streamp** strm, **uLong** sourceLen))
- ZEXTERN int ZEXPORT deflatePrime **OF** ((**z_streamp** strm, int bits, int value))
- ZEXTERN int ZEXPORT deflateSetHeader **OF** ((**z_streamp** strm, **gz_headerp** head))
- ZEXTERN int ZEXPORT inflateReset2 **OF** ((**z_streamp** strm, int windowBits))
- ZEXTERN int ZEXPORT inflateBack **OF** ((**z_streamp** strm, in_func in, void FAR *in_desc, out_func out, void FAR *out_desc))
- ZEXTERN int ZEXPORT compress **OF** ((**Bytef** *dest, **uLongf** *destLen, const **Bytef** *source, **uLong** sourceLen))
- ZEXTERN int ZEXPORT compress2 **OF** ((**Bytef** *dest, **uLongf** *destLen, const **Bytef** *source, **uLong** sourceLen, int level))
- ZEXTERN **uLong** ZEXPORT compressBound **OF** ((**uLong** sourceLen))
- ZEXTERN **gzFile** ZEXPORT gzdopen **OF** ((int fd, const char *mode))
- ZEXTERN int ZEXPORT gzbuffer **OF** ((**gzFile** file, unsigned size))
- ZEXTERN int ZEXPORT gzsetparams **OF** ((**gzFile** file, int level, int strategy))
- ZEXTERN int ZEXPORT gzread **OF** ((**gzFile** file, **voidp** buf, unsigned len))

- ZEXTERN int ZEXPORT gzwrite **OF** ((**gzFile** file, **voidpc** buf, unsigned len))
- ZEXTERN int ZEXPORTVA gzprintf **OF** ((**gzFile** file, const char *format,...))
- ZEXTERN int ZEXPORT gzputs **OF** ((**gzFile** file, const char *s))
- ZEXTERN char *ZEXPORT gzgets **OF** ((**gzFile** file, char *buf, int len))
- ZEXTERN int ZEXPORT gzputc **OF** ((**gzFile** file, int c))
- ZEXTERN int ZEXPORT gzgetc **OF** ((**gzFile** file))
- ZEXTERN int ZEXPORT gzungetc **OF** ((int c, **gzFile** file))
- ZEXTERN int ZEXPORT gzflush **OF** ((**gzFile** file, int flush))
- ZEXTERN const char *ZEXPORT gzerror **OF** ((**gzFile** file, int *errnum))
- ZEXTERN **uLong** ZEXPORT Adler32 **OF** ((**uLong** Adler, const **Bytef** *buf, **uInt** len))
- ZEXTERN **uLong** ZEXPORT crc32 **OF** ((**uLong** crc, const **Bytef** *buf, **uInt** len))
- ZEXTERN int ZEXPORT deflateInit_ **OF** ((**z_stream** strm, int level, const char *version, int stream_size))
- ZEXTERN int ZEXPORT inflateInit_ **OF** ((**z_stream** strm, const char *version, int stream_size))
- ZEXTERN int ZEXPORT deflateInit2_ **OF** ((**z_stream** strm, intlevel, intmethod, int windowBits, int memLevel, int strategy, const char *version, int stream_size))
- ZEXTERN int ZEXPORT inflateInit2_ **OF** ((**z_stream** strm, intwindowBits, const char *version, int stream_size))
- ZEXTERN int ZEXPORT inflateBackInit_ **OF** ((**z_stream** strm, int windowBits, unsigned char FAR *window, const char *version, int stream_size))
- ZEXTERN **gzFile** ZEXPORT gzopen **OF** ((const char *, const char *))
- ZEXTERN **z_off_t** ZEXPORT gzseek **OF** ((**gzFile**, **z_off_t**, int))
- ZEXTERN **z_off_t** ZEXPORT gztell **OF** ((**gzFile**))
- ZEXTERN **uLong** ZEXPORT adler32_combine **OF** ((**uLong**, **uLong**, **z_off_t**))
- ZEXTERN const char *ZEXPORT zError **OF** ((int))
- ZEXTERN int ZEXPORT inflateSyncPoint **OF** ((**z_stream**))
- ZEXTERN int ZEXPORT inflateUndermine **OF** ((**z_stream**, int))

7.372.1 Define Documentation

7.372.1.1 **#define deflateInit(strm, level) deflateInit_((strm), (level), ZLIB_VERSION, sizeof(z_stream))**

7.372.1.2 **#define deflateInit2(strm, level, method, windowBits, memLevel, strategy)**

Value:

```
deflateInit2_((strm), (level), (method), (windowBits), (memLevel), \
              (strategy), ZLIB_VERSION, sizeof(z_stream))
```

7.372.1.3 **#define inflateBackInit(strm, windowBits, window)**

Value:

```
inflateBackInit_((strm), (windowBits), (window), \
                 ZLIB_VERSION, sizeof(z_stream))
```

- 7.372.1.4 `#define inflateInit(strm) inflateInit_((strm), ZLIB_VERSION, sizeof(z_stream))`
- 7.372.1.5 `#define inflateInit2(strm, windowBits) inflateInit2_((strm), (windowBits), ZLIB_VERSION, sizeof(z_stream))`
- 7.372.1.6 `#define Z_ASCII Z_TEXT /* for compatibility with 1.2.2 and earlier */`
- 7.372.1.7 `#define Z_BEST_COMPRESSION 9`
- 7.372.1.8 `#define Z_BEST_SPEED 1`
- 7.372.1.9 `#define Z_BINARY 0`
- 7.372.1.10 `#define Z_BLOCK 5`
- 7.372.1.11 `#define Z_BUF_ERROR (-5)`
- 7.372.1.12 `#define Z_DATA_ERROR (-3)`
- 7.372.1.13 `#define Z_DEFAULT_COMPRESSION (-1)`
- 7.372.1.14 `#define Z_DEFAULT_STRATEGY 0`
- 7.372.1.15 `#define Z_DEFLATED 8`
- 7.372.1.16 `#define Z_ERRNO (-1)`
- 7.372.1.17 `#define Z_FILTERED 1`
- 7.372.1.18 `#define Z_FINISH 4`
- 7.372.1.19 `#define Z_FIXED 4`
- 7.372.1.20 `#define Z_FULL_FLUSH 3`
- 7.372.1.21 `#define Z_HUFFMAN_ONLY 2`
- 7.372.1.22 `#define Z_MEM_ERROR (-4)`
- 7.372.1.23 `#define Z_NEED_DICT 2`
- 7.372.1.24 `#define Z_NO_COMPRESSION 0`
- 7.372.1.25 `#define Z_NO_FLUSH 0`
- 7.372.1.26 `#define Z_NULL 0 /* for initializing zalloc, zfree, opaque */`

7.372.1.27 `#define Z_OK 0`

7.372.1.28 `#define Z_PARTIAL_FLUSH 1`

7.372.1.29 `#define Z_RLE 3`

7.372.1.30 `#define Z_STREAM_END 1`

7.372.1.31 `#define Z_STREAM_ERROR (-2)`

7.372.1.32 `#define Z_SYNC_FLUSH 2`

7.372.1.33 `#define Z_TEXT 1`

7.372.1.34 `#define Z_TREES 6`

7.372.1.35 `#define Z_UNKNOWN 2`

7.372.1.36 `#define Z_VERSION_ERROR (-6)`

7.372.1.37 `#define ZLIB_VER_MAJOR 1`

7.372.1.38 `#define ZLIB_VER_MINOR 2`

7.372.1.39 `#define ZLIB_VER_REVISION 5`

7.372.1.40 `#define ZLIB_VER_SUBREVISION 0`

7.372.1.41 `#define ZLIB_VERNUM 0x1250`

7.372.1.42 `#define ZLIB_VERSION "1.2.5"`

7.372.1.43 `#define zlib_version zlibVersion()`

7.372.2 Typedef Documentation

7.372.2.1 `typedef struct gz_header_s gz_header`

7.372.2.2 `typedef gz_header FAR* gz_headerp`

7.372.2.3 `typedef voidp gzFile`

7.372.2.4 `ZEXTERN uLong ZEXPORT crc32_combine64 OF ((voidpf opaque, uInt items, uInt size))`

7.372.2.5 `typedef struct z_stream_s z_stream`

7.372.2.6 typedef z_stream FAR* z_streamp

7.372.3 Function Documentation

7.372.3.1 ZEXTERN const char* ZEXPORT zlibVersion OF ((void))

7.372.3.2 ZEXTERN int ZEXPORT deflate OF ((z_streamp strm, int flush))

7.372.3.3 ZEXTERN int ZEXPORT deflateEnd OF ((z_streamp strm))

7.372.3.4 ZEXTERN int ZEXPORT deflateSetDictionary OF ((z_streamp strm, const Bytef *dictionary, ulndictLength))

7.372.3.5 ZEXTERN int ZEXPORT deflateCopy OF ((z_streamp dest, z_streamp source))

7.372.3.6 ZEXTERN int ZEXPORT deflateParams OF ((z_streamp strm, int level, int strategy))

7.372.3.7 ZEXTERN int ZEXPORT deflateTune OF ((z_streamp strm, int good_length, int max_lazy, int nice_length, int max_chain))

7.372.3.8 ZEXTERN uLong ZEXPORT deflateBound OF ((z_streamp strm, uLong sourceLen))

7.372.3.9 ZEXTERN int ZEXPORT deflatePrime OF ((z_streamp strm, int bits, int value))

7.372.3.10 ZEXTERN int ZEXPORT deflateSetHeader OF ((z_streamp strm, gz_headerp head))

7.372.3.11 ZEXTERN int ZEXPORT inflateReset2 OF ((z_streamp strm, int windowBits))

7.372.3.12 ZEXTERN int ZEXPORT inflateBack OF ((z_streamp strm, in_func in, void FAR *in_desc, out_func out, void FAR *out_desc))

7.372.3.13 ZEXTERN int ZEXPORT compress OF ((Bytef *dest, uLongf *destLen, const Bytef *source, uLong sourceLen))

7.372.3.14 ZEXTERN int ZEXPORT compress2 OF ((Bytef *dest, uLongf *destLen, const Bytef *source, uLong sourceLen, int level))

7.372.3.15 ZEXTERN uLong ZEXPORT compressBound OF ((uLong sourceLen))

7.372.3.16 ZEXTERN gzFile ZEXPORT gzdopen OF ((int fd, const char *mode))

7.372.3.17 ZEXTERN int ZEXPORT gzbuffer OF ((gzFile file, unsigned size))

7.372.3.18 ZEXTERN int ZEXPORT gzsetparams OF ((gzFile file, int level, int strategy))

- 7.372.3.19 ZEXTERN int ZEXPORT gzread OF ((gzFile file, voidp buf, unsigned len))
- 7.372.3.20 ZEXTERN int ZEXPORT gzwrite OF ((gzFile file, voidpc buf, unsigned len))
- 7.372.3.21 ZEXTERN int ZEXPORTVA gzprintf OF ((gzFile file, const char *format,...))
- 7.372.3.22 ZEXTERN int ZEXPORT gzputs OF ((gzFile file, const char *s))
- 7.372.3.23 ZEXTERN char* ZEXPORT gzgets OF ((gzFile file, char *buf, int len))
- 7.372.3.24 ZEXTERN int ZEXPORT gzputc OF ((gzFile file, int c))
- 7.372.3.25 ZEXTERN int ZEXPORT gzgetc OF ((gzFile file))
- 7.372.3.26 ZEXTERN int ZEXPORT gzungetc OF ((int c, gzFile file))
- 7.372.3.27 ZEXTERN int ZEXPORT gzflush OF ((gzFile file, int flush))
- 7.372.3.28 ZEXTERN const char* ZEXPORT gzerror OF ((gzFile file, int *errnum))
- 7.372.3.29 ZEXTERN uLong ZEXPORT Adler32 OF ((uLong Adler, const Bytef *buf, uInt len))
- 7.372.3.30 ZEXTERN uLong ZEXPORT Crc32 OF ((uLong Crc, const Bytef *buf, uInt len))
- 7.372.3.31 ZEXTERN int ZEXPORT deflateInit_ OF ((z_streamp strm, int level, const char *version, int stream_size))
- 7.372.3.32 ZEXTERN int ZEXPORT inflateInit_ OF ((z_streamp strm, const char *version, int stream_size))
- 7.372.3.33 ZEXTERN int ZEXPORT deflateInit2_ OF ((z_streamp strm, intlevel, intmethod, int windowBits, int memLevel, int strategy, const char *version, int stream_size))
- 7.372.3.34 ZEXTERN int ZEXPORT inflateInit2_ OF ((z_streamp strm, intwindowBits, const char *version, int stream_size))
- 7.372.3.35 ZEXTERN int ZEXPORT inflateBackInit_ OF ((z_streamp strm, int windowBits, unsigned char FAR *window, const char *version, int stream_size))
- 7.372.3.36 ZEXTERN gzFile ZEXPORT gzopen OF ((const char *, const char *))
- 7.372.3.37 ZEXTERN z_off_t ZEXPORT gzseek OF ((gzFile, z_off_t, int))
- 7.372.3.38 ZEXTERN z_off_t ZEXPORT gztell OF ((gzFile))
- 7.372.3.39 ZEXTERN uLong ZEXPORT Adler32_combine OF ((uLong, uLong, z_off_t))

7.372.3.40 ZEXTERN const char* ZEXPORT zError OF ((int))

7.372.3.41 ZEXTERN int ZEXPORT inflateSyncPoint OF ((z_streamp))

7.372.3.42 ZEXTERN int ZEXPORT inflateUndermine OF ((z_streamp, int))

7.373 src/main/decaf/internal/util/zip/zutil.h File Reference

```
#include "zlib.h"
```

Defines

- #define **ZLIB_INTERNAL**
- #define **ERR_MSG**(err) **z_errmsg**[**Z_NEED_DICT**-(err)]
- #define **ERR_RETURN**(strm, err) return (strm->msg = (char*)**ERR_MSG**(err), (err))
- #define **DEF_MEM_LEVEL** 8
- #define **STORED_BLOCK** 0
- #define **STATIC_TREES** 1
- #define **DYN_TREES** 2
- #define **MIN_MATCH** 3
- #define **MAX_MATCH** 258
- #define **PRESET_DICT** 0x20 /* preset dictionary flag in zlib header */
- #define **Assert**(cond, msg)
- #define **Trace**(x)
- #define **Tracev**(x)
- #define **Tracevv**(x)
- #define **Tracec**(c, x)
- #define **Tracecv**(c, x)
- #define **ZALLOC**(strm, items, size) (*((strm)->zalloc)((strm)->opaque, (items), (size))
- #define **ZFREE**(strm, addr) (*((strm)->zfree)((strm)->opaque, (**voidpf**)(addr))
- #define **TRY_FREE**(s, p) {if (p) **ZFREE**(s, p);}

Typedefs

- typedef unsigned char **uch**
- typedef **uch** FAR **uchf**
- typedef unsigned short **ush**
- typedef **ush** FAR **ushf**
- typedef unsigned long **ulg**

Functions

- ZEXTERN uLong ZEXPORT Adler32_combine64 OF ((uLong, uLong, z_off_t))
- void ZLIB_INTERNAL zmemcpy OF ((Bytef *dest, const Bytef *source, ulint len))
- int ZLIB_INTERNAL zmemcmp OF ((const Bytef *s1, const Bytef *s2, ulint len))
- void ZLIB_INTERNAL zmemzero OF ((Bytef *dest, ulint len))
- voidpf ZLIB_INTERNAL zcalloc OF ((voidpf opaque, unsigned items, unsigned size))
- void ZLIB_INTERNAL zcfree OF ((voidpf opaque, voidpf ptr))

Variables

- const char *const z_errmsg [10]

7.373.1 Define Documentation

7.373.1.1 #define Assert(*cond*, *msg*)

7.373.1.2 #define DEF_MEM_LEVEL 8

7.373.1.3 #define DYN_TREES 2

7.373.1.4 #define ERR_MSG(*err*) z_errmsg[Z_NEED_DICT-(err)]

7.373.1.5 #define ERR_RETURN(*strm*, *err*) return (strm->msg = (char*)ERR_MSG(err), (err))

7.373.1.6 #define MAX_MATCH 258

7.373.1.7 #define MIN_MATCH 3

7.373.1.8 #define PRESET_DICT 0x20 /* preset dictionary flag in zlib header */

7.373.1.9 #define STATIC_TREES 1

7.373.1.10 #define STORED_BLOCK 0

7.373.1.11 #define Trace(*x*)

7.373.1.12 #define Tracec(*c*, *x*)

7.373.1.13 #define Tracecv(*c*, *x*)

7.373.1.14 #define Tracev(*x*)

7.373.1.15 `#define Tracevv(x)`

7.373.1.16 `#define TRY_FREE(s, p) { if (p) ZFREE(s, p); }`

7.373.1.17 `#define ZALLOC(strm, items, size) (*((strm)->zalloc)((strm)->opaque, (items), (size))`

7.373.1.18 `#define ZFREE(strm, addr) (*((strm)->zfree)((strm)->opaque, (voidpf)(addr))`

7.373.1.19 `#define ZLIB_INTERNAL`

7.373.2 Typedef Documentation

7.373.2.1 `typedef unsigned char uch`

7.373.2.2 `typedef uch FAR uchf`

7.373.2.3 `typedef unsigned long ulg`

7.373.2.4 `typedef unsigned short ush`

7.373.2.5 `typedef ush FAR ushf`

7.373.3 Function Documentation

7.373.3.1 `ZEXTERN uLong ZEXPORT Adler32_combine64 OF ((uLong, uLong, z_off_t))`

7.373.3.2 `void ZLIB_INTERNAL zmemcpy OF ((Bytef *dest, const Bytef *source, ulint len))`

7.373.3.3 `int ZLIB_INTERNAL zmemcmp OF ((const Bytef *s1, const Bytef *s2, ulint len))`

7.373.3.4 `void ZLIB_INTERNAL zmemzero OF ((Bytef *dest, ulint len))`

7.373.3.5 `voidpf ZLIB_INTERNAL zcalloc OF ((voidpf opaque, unsigned items, unsigned size))`

7.373.3.6 `void ZLIB_INTERNAL zcfree OF ((voidpf opaque, voidpf ptr))`

7.373.4 Variable Documentation

7.373.4.1 `const char* const z_errmsg[10]`

7.374 src/main/decaf/io/BlockingByteArrayInputStream.h File - Reference

```
#include <decaf/io/InputStream.h> #include <vector>
```

Data Structures

- class **decaf::io::BlockingByteArrayInputStream**

This is a blocking version of a byte buffer stream.

Namespaces

- namespace **decaf**
Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.
- namespace **decaf::io**

7.375 src/main/decaf/io/BufferedInputStream.h File Reference

```
#include <decaf/util/Config.h> #include <decaf/io/Filter-  
InputStream.h> #include <decaf/lang/exceptions/Illegal-  
ArgumentException.h>
```

Data Structures

- class **decaf::io::BufferedInputStream**

A wrapper around another input stream that performs a buffered read, where it reads more data than it needs in order to reduce the number of io operations on the input stream.

Namespaces

- namespace **decaf**
Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.
- namespace **decaf::io**

7.376 src/main/decaf/io/BufferedOutputStream.h File Reference

```
#include <decaf/io/FilterOutputStream.h> #include <decaf/lang/exceptions/-  
IllegalArgumentException.h>
```

Data Structures

- class **decaf::io::BufferedOutputStream**

Wrapper around another output stream that buffers output before writing to the target output stream.

Namespaces

- namespace **decaf**

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

- namespace **decaf::io**

7.377 src/main/decaf/io/ByteArrayInputStream.h File Reference

```
#include <decaf/io/InputStream.h> #include <decaf/util/concurrent/-  
Mutex.h> #include <vector>
```

Data Structures

- class **decaf::io::ByteArrayInputStream**

*A **ByteArrayInputStream** (p. 679) contains an internal buffer that contains bytes that may be read from the stream.*

Namespaces

- namespace **decaf**

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

- namespace **decaf::io**

7.378 src/main/decaf/io/ByteArrayOutputStream.h File Reference

```
#include <decaf/util/Config.h> #include <decaf/io/Output-  
Stream.h> #include <decaf/lang/exceptions/IllegalArgumentException-  
Exception.h> #include <utility>
```

Data Structures

- class **decaf::io::ByteArrayOutputStream**

Namespaces

- namespace **decaf**

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

- namespace **decaf::io**

7.379 src/main/decaf/io/DataInput.h File Reference

```
#include <decaf/util/Config.h> #include <vector> #include
<string> #include <decaf/io/IOException.h> #include <decaf/io/-
EOFException.h> #include <decaf/io/UTFDataFormatException.-
h> #include <decaf/lang/exceptions/NullPointerException.-
h> #include <decaf/lang/exceptions/IndexOutOfBoundsException.-
h>
```

Data Structures

- class **decaf::io::DataInput**

*The **DataInput** (p. 1086) interface provides for reading bytes from a binary stream and reconstructing from them data in any of the C++ primitive types.*

Namespaces

- namespace **decaf**

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

- namespace **decaf::io**

7.380 src/main/decaf/io/DataInputStream.h File Reference

```
#include <decaf/io/FilterInputStream.h> #include <decaf/io/-
IOException.h> #include <decaf/io/EOFException.h> #include
<decaf/io/UTFDataFormatException.h> #include <decaf/lang/exceptions/-
NullPointerException.h> #include <decaf/lang/exceptions/-
IndexOutOfBoundsException.h>
```

Data Structures

- class **decaf::io::DataInputStream**

A data input stream lets an application read primitive Java data types from an underlying input stream in a machine-independent way.

Namespaces

- namespace **decaf**

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

- namespace **decaf::io**

7.381 src/main/decaf/io/DataOutput.h File Reference

```
#include <decaf/util/Config.h> #include <vector> #include  
<string> #include <decaf/io/IOException.h> #include <decaf/io/-  
EOFException.h> #include <decaf/lang/exceptions/Null-  
PointerException.h> #include <decaf/lang/exceptions/-  
IndexOutOfBoundsException.h>
```

Data Structures

- class **decaf::io::DataOutput**

*The **DataOutput** (p. 1103) interface provides for converting data from any of the C++ primitive types to a series of bytes and writing these bytes to a binary stream.*

Namespaces

- namespace **decaf**

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

- namespace **decaf::io**

7.382 src/main/decaf/io/DataOutputStream.h File Reference

```
#include <decaf/io/FilterOutputStream.h> #include <decaf/io/-  
IOException.h> #include <decaf/io/UTFDataFormatException.-  
h> #include <string>
```

Data Structures

- class **decaf::io::DataOutputStream**

A data output stream lets an application write primitive Java data types to an output stream in a portable way.

Namespaces

- namespace **decaf**

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

- namespace **decaf::io**

7.383 src/main/decaf/io/EOFException.h File Reference

```
#include <decaf/io/IOException.h>
```

Data Structures

- class **decaf::io::EOFException**

Namespaces

- namespace **decaf**

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

- namespace **decaf::io**

7.384 src/main/decaf/io/FileDescriptor.h File Reference

```
#include <decaf/util/Config.h>
```

Data Structures

- class **decaf::io::FileDescriptor**

This class servers as an opaque wrapper around an underlying OS level resource that can be used as a source / sink for byte level data such as sockets and files.

Namespaces

- namespace **decaf**

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

- namespace **decaf::io**

7.385 src/main/decaf/io/FilterInputStream.h File Reference

```
#include <decaf/io/InputStream.h> #include <decaf/io/IO-  
Exception.h> #include <decaf/lang/exceptions/NullPointer-  
Exception.h> #include <decaf/lang/exceptions/IndexOutOf-  
BoundsException.h>
```

Data Structures

- class **decaf::io::FilterInputStream**

A **FilterInputStream** (p. 1334) contains some other input stream, which it uses as its basic source of data, possibly transforming the data along the way or providing additional functionality.

Namespaces

- namespace **decaf**

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

- namespace **decaf::io**

7.386 src/main/decaf/io/FilterOutputStream.h File Reference

```
#include <decaf/io/OutputStream.h> #include <decaf/io/IO-  
Exception.h> #include <decaf/lang/exceptions/NullPointer-  
Exception.h> #include <decaf/lang/exceptions/IndexOutOf-  
BoundsException.h>
```

Data Structures

- class **decaf::io::FilterOutputStream**

This class is the superclass of all classes that filter output streams.

Namespaces

- namespace **decaf**

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

- namespace **decaf::io**

7.387 src/main/decaf/io/Flushable.h File Reference

```
#include <decaf/util/Config.h>      #include <decaf/io/IO-  
Exception.h>
```

Data Structures

- class **decaf::io::Flushable**

*A **Flushable** (p. 1380) is a destination of data that can be flushed.*

Namespaces

- namespace **decaf**

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

- namespace **decaf::io**

7.388 src/main/decaf/io/InputStream.h File Reference

```
#include <decaf/io/IOException.h>    #include <decaf/io/-  
Closeable.h> #include <decaf/util/concurrent/Synchronizable.-  
h>    #include <decaf/util/concurrent/Mutex.h>    #include  
<decaf/lang/exceptions/UnsupportedOperationException.h>  
#include <decaf/lang/exceptions/NullPointerException.h>  
#include <decaf/lang/exceptions/IndexOutOfBoundsException.-  
h> #include <decaf/util/Config.h>
```

Data Structures

- class **decaf::io::InputStream**

A base class that must be implemented by all classes wishing to provide a class that reads in a stream of bytes.

Namespaces

- namespace **decaf**

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

- namespace **decaf::io**

7.389 src/main/decaf/io/InputStreamReader.h File Reference

```
#include <decaf/util/Config.h> #include <decaf/io/Reader.h>
```

Data Structures

- class **decaf::io::InputStreamReader**

An *InputStreamReader* (p. 1475) is a bridge from byte streams to character streams.

Namespaces

- namespace **decaf**

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

- namespace **decaf::io**

7.390 src/main/decaf/io/InterruptedIOException.h File Reference

```
#include <decaf/lang/Exception.h> #include <decaf/io/IOException.h>
```

Data Structures

- class **decaf::io::InterruptedIOException**

Namespaces

- namespace **decaf**

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

- namespace **decaf::io**

7.391 src/main/decaf/io/IOException.h File Reference

```
#include <decaf/lang/Exception.h>
```

Data Structures

- class **decaf::io::IOException**

Namespaces

- namespace **decaf**

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

- namespace **decaf::io**

7.392 src/main/decaf/io/OutputStream.h File Reference

```
#include <decaf/io/Closeable.h>          #include <decaf/io/-  
Flushable.h> #include <decaf/io/IOException.h> #include  
<decaf/util/concurrent/Synchronizable.h> #include <decaf/util/concurrent/-  
Mutex.h> #include <decaf/util/Config.h> #include <decaf/lang/exceptions/-  
NullPointerException.h> #include <decaf/lang/exceptions/-  
IndexOutOfBoundsException.h>
```

Data Structures

- class **decaf::io::OutputStream**

Base interface for any class that wants to represent an output stream of bytes.

Namespaces

- namespace **decaf**

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

- namespace **decaf::io**

7.393 src/main/decaf/io/OutputStreamWriter.h File Reference

```
#include <decaf/util/Config.h> #include <decaf/io/Writer.-  
h>
```

Data Structures

- class **decaf::io::OutputStreamWriter**

A class for turning a character stream into a byte stream.

Namespaces

- namespace **decaf**

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

- namespace **decaf::io**

7.394 src/main/decaf/io/PushbackInputStream.h File Reference

```
#include <decaf/util/Config.h> #include <decaf/io/Input-  
Stream.h> #include <decaf/io/FilterInputStream.h>
```

Data Structures

- class **decaf::io::PushbackInputStream**

*A **PushbackInputStream** (p. 2214) adds functionality to another input stream, namely the ability to "push back" or "unread" one byte.*

Namespaces

- namespace **decaf**

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

- namespace **decaf::io**

7.395 src/main/decaf/io/Reader.h File Reference

```
#include <string> #include <decaf/lang/Readable.h> ×  
#include <decaf/io/Closeable.h> #include <decaf/io/IO-  
Exception.h> #include <decaf/io/InputStream.h> #include  
<decaf/lang/exceptions/IndexOutOfBoundsException.h> ×  
#include <decaf/lang/exceptions/NullPointerException.h>
```

Data Structures

- class **decaf::io::Reader**

Namespaces

- namespace **decaf**

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

- namespace **decaf::io**

7.396 src/main/decaf/io/UnsupportedEncodingException.h File - Reference

```
#include <decaf/io/IOException.h>
```

Data Structures

- class **decaf::io::UnsupportedEncodingException**
Thrown when the the Character Encoding is not supported.

Namespaces

- namespace **decaf**
Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.
- namespace **decaf::io**

7.397 src/main/decaf/io/UTFDataFormatException.h File Reference

```
#include <decaf/io/IOException.h>
```

Data Structures

- class **decaf::io::UTFDataFormatException**
Thrown from classes that attempt to read or write a UTF-8 encoded string and an encoding error is encountered.

Namespaces

- namespace **decaf**
Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.
- namespace **decaf::io**

7.398 src/main/decaf/io/Writer.h File Reference

```
#include <string> #include <vector> #include <decaf/io/-  
IOException.h> #include <decaf/io/Closeable.h> #include  
<decaf/io/Flushable.h> #include <decaf/lang/Appendable.-  
h> #include <decaf/lang/exceptions/NullPointerException.-  
h> #include <decaf/lang/exceptions/IndexOutOfBoundsException.-  
h>
```

Data Structures

- class **decaf::io::Writer**

Namespaces

- namespace **decaf**
Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.
- namespace **decaf::io**

7.399 src/main/decaf/lang/Appendable.h File Reference

```
#include <decaf/lang/Exception.h> #include <decaf/util/-
Config.h>
```

Data Structures

- class **decaf::lang::Appendable**
An object to which char sequences and values can be appended.

Namespaces

- namespace **decaf**
Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.
- namespace **decaf::lang**

7.400 src/main/decaf/lang/ArrayPointer.h File Reference

```
#include <decaf/util/Config.h> #include <decaf/lang/-
System.h> #include <decaf/lang/exceptions/NullPointer-
Exception.h> #include <decaf/lang/exceptions/IndexOut-
OfBoundsException.h> #include <decaf/lang/exceptions/-
IllegalArgumentException.h> #include <decaf/util/concurrent/atomic/-
AtomicRefCounter.h> #include <decaf/util/Comparator.h> ×
#include <decaf/util/Arrays.h> #include <memory> #include
<typeinfo> #include <algorithm>
```

Data Structures

- class **decaf::lang::ArrayPointer< T, REFCOUNTER >**

*Decaf's implementation of a Smart **Pointer** (p. 2083) that is a template on a Type and is **Thread** (p. 2703) Safe if the default Reference Counter is used.*

- struct **decaf::lang::ArrayPointer**< T, REFCOUNTER >::ArrayData
- class **decaf::lang::ArrayPointerComparator**< T, R >

*This implementation of Comparator is designed to allows objects in a Collection to be sorted or tested for equality based on the value of the value of the actual pointer to the array being contained in this **ArrayPointer** (p. 462).*

- struct **std::less**< **decaf::lang::ArrayPointer**< T > >

An override of the less function object so that the Pointer objects can be stored in STL Maps, etc.

Namespaces

- namespace **decaf**

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

- namespace **decaf::lang**
- namespace **std**

Functions

- template<typename T , typename R , typename U >
bool **decaf::lang::operator==** (const ArrayPointer< T, R > &left, const U *right)
- template<typename T , typename R , typename U >
bool **decaf::lang::operator==** (const U *left, const ArrayPointer< T, R > &right)
- template<typename T , typename R , typename U >
bool **decaf::lang::operator!=** (const ArrayPointer< T, R > &left, const U *right)
- template<typename T , typename R , typename U >
bool **decaf::lang::operator!=** (const U *left, const ArrayPointer< T, R > &right)

7.401 src/main/decaf/lang/Boolean.h File Reference

```
#include <string>      #include <decaf/lang/Comparable.h> ×
#include <decaf/util/Config.h>
```

Data Structures

- class **decaf::lang::Boolean**

Namespaces

- namespace **decaf**

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

- namespace **decaf::lang**

7.402 src/main/decaf/lang/Byte.h File Reference

```
#include <decaf/util/Config.h>      #include <decaf/lang/-  
Number.h>      #include <decaf/lang/Comparable.h>      #include  
<decaf/lang/exceptions/NumberFormatException.h>      #include  
<string>
```

Data Structures

- class **decaf::lang::Byte**

Namespaces

- namespace **decaf**
*Licensed to the Apache Software Foundation (ASF) under one or more contributor
license agreements.*
- namespace **decaf::lang**

7.403 src/main/decaf/lang/Character.h File Reference

```
#include <decaf/util/Config.h>      #include <decaf/lang/-  
Number.h>      #include <decaf/lang/Comparable.h>      #include  
<string>
```

Data Structures

- class **decaf::lang::Character**

Namespaces

- namespace **decaf**
*Licensed to the Apache Software Foundation (ASF) under one or more contributor
license agreements.*
- namespace **decaf::lang**

7.404 src/main/decaf/lang/CharSequence.h File Reference

```
#include <decaf/lang/exceptions/IndexOutOfBoundsException.-  
h> #include <decaf/util/Config.h>
```

Data Structures

- class **decaf::lang::CharSequence**

*A **CharSequence** (p. 803) is a readable sequence of char values.*

Namespaces

- namespace **decaf**

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

- namespace **decaf::lang**

7.405 src/main/decaf/lang/Comparable.h File Reference

```
#include <decaf/util/Config.h>
```

Data Structures

- class **decaf::lang::Comparable< T >**

This interface imposes a total ordering on the objects of each class that implements it.

Namespaces

- namespace **decaf**

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

- namespace **decaf::lang**

7.406 src/main/decaf/lang/Double.h File Reference

```
#include <decaf/util/Config.h>      #include <decaf/lang/-  
Comparable.h>  #include <decaf/lang/Number.h>  #include  
<decaf/lang/exceptions/NumberFormatException.h>  #include  
<string>
```

Data Structures

- class **decaf::lang::Double**

Namespaces

- namespace **decaf**
Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.
- namespace **decaf::lang**

7.407 src/main/decaf/lang/Exception.h File Reference

```
#include <decaf/lang/Throwable.h> #include <decaf/lang/exceptions/-  
ExceptionDefines.h> #include <decaf/util/Config.h> #include  
<stdarg.h> #include <sstream>
```

Data Structures

- class **decaf::lang::Exception**

Namespaces

- namespace **decaf**
Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.
- namespace **decaf::lang**

7.408 src/main/decaf/lang/exceptions/ClassCastException.h File - Reference

```
#include <decaf/lang/Exception.h>
```

Data Structures

- class **decaf::lang::exceptions::ClassCastException**

Namespaces

- namespace **decaf**
Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.
- namespace **decaf::lang**
- namespace **decaf::lang::exceptions**

7.409 src/main/decaf/lang/exceptions/IllegalArgumentException.h File Reference

```
#include <decaf/lang/Exception.h>
```

Data Structures

- class **decaf::lang::exceptions::IllegalArgumentException**

Namespaces

- namespace **decaf**
Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.
- namespace **decaf::lang**
- namespace **decaf::lang::exceptions**

7.410 src/main/decaf/lang/exceptions/IllegalMonitorStateException.h File Reference

```
#include <decaf/lang/Exception.h>
```

Data Structures

- class **decaf::lang::exceptions::IllegalMonitorStateException**

Namespaces

- namespace **decaf**
Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.
- namespace **decaf::lang**
- namespace **decaf::lang::exceptions**

7.411 src/main/decaf/lang/exceptions/IllegalThreadStateException.h File Reference

```
#include <decaf/lang/Exception.h>
```

Data Structures

- class **decaf::lang::exceptions::IllegalThreadStateException**

Namespaces

- namespace **decaf**
Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.
- namespace **decaf::lang**
- namespace **decaf::lang::exceptions**

7.412 src/main/decaf/lang/exceptions/IndexOutOfBoundsException.h File Reference

```
#include <decaf/lang/Exception.h>
```

Data Structures

- class **decaf::lang::exceptions::IndexOutOfBoundsException**

Namespaces

- namespace **decaf**
Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.
- namespace **decaf::lang**
- namespace **decaf::lang::exceptions**

7.413 src/main/decaf/lang/exceptions/InterruptedException.h File - Reference

```
#include <decaf/lang/Exception.h>
```

Data Structures

- class **decaf::lang::exceptions::InterruptedException**

Namespaces

- namespace **decaf**
Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.
- namespace **decaf::lang**
- namespace **decaf::lang::exceptions**

7.414 src/main/decaf/lang/exceptions/InvalidStateException.h File Reference

```
#include <decaf/lang/Exception.h>
```

Data Structures

- class **decaf::lang::exceptions::InvalidStateException**

Namespaces

- namespace **decaf**
Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.
- namespace **decaf::lang**
- namespace **decaf::lang::exceptions**

7.415 src/main/decaf/lang/exceptions/NullPointerException.h File - Reference

```
#include <decaf/lang/Exception.h>
```

Data Structures

- class **decaf::lang::exceptions::NullPointerException**

Namespaces

- namespace **decaf**
Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.
- namespace **decaf::lang**
- namespace **decaf::lang::exceptions**

7.416 src/main/decaf/lang/exceptions/NumberFormatException.h File Reference

7.416 src/main/decaf/lang/exceptions/NumberFormatException.h - File Reference

```
#include <decaf/lang/Exception.h>
```

Data Structures

- class **decaf::lang::exceptions::NumberFormatException**

Namespaces

- namespace **decaf**
Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.
- namespace **decaf::lang**
- namespace **decaf::lang::exceptions**

7.417 src/main/decaf/lang/exceptions/RuntimeException.h File - Reference

```
#include <decaf/lang/Exception.h>
```

Data Structures

- class **decaf::lang::exceptions::RuntimeException**

Namespaces

- namespace **decaf**
Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.
- namespace **decaf::lang**
- namespace **decaf::lang::exceptions**

7.418 src/main/decaf/lang/Float.h File Reference

```
#include <decaf/util/Config.h>      #include <decaf/lang/-  
Number.h>      #include <decaf/lang/Comparable.h>      #include  
<decaf/lang/exceptions/NumberFormatException.h>      #include  
<string>
```

Data Structures

- class **decaf::lang::Float**

Namespaces

- namespace **decaf**

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

- namespace **decaf::lang**

7.419 src/main/decaf/lang/Integer.h File Reference

```
#include <decaf/util/Config.h>      #include <decaf/lang/-  
Number.h>    #include <decaf/lang/Comparable.h>    #include  
<string>      #include <decaf/lang/exceptions/NumberFormatException-  
Exception.h>
```

Data Structures

- class **decaf::lang::Integer**

Namespaces

- namespace **decaf**

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

- namespace **decaf::lang**

7.420 src/main/decaf/lang/Iterable.h File Reference

```
#include <decaf/util/Config.h>      #include <decaf/util/-  
Iterator.h>
```

Data Structures

- class **decaf::lang::Iterable< E >**

*Implementing this interface allows an object to be cast to an **Iterable** (p. 1556) type for generic collections API calls.*

Namespaces

- namespace **decaf**
Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.
- namespace **decaf::lang**

7.421 src/main/decaf/lang/Long.h File Reference

```
#include <decaf/lang/Number.h>      #include <decaf/lang/-  
Comparable.h>      #include <decaf/lang/exceptions/Number-  
FormatException.h> #include <string>
```

Data Structures

- class **decaf::lang::Long**

Namespaces

- namespace **decaf**
Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.
- namespace **decaf::lang**

7.422 src/main/decaf/lang/Math.h File Reference

```
#include <decaf/util/Config.h>
```

Data Structures

- class **decaf::lang::Math**
*The class **Math** (p. 1800) contains methods for performing basic numeric operations such as the elementary exponential, logarithm, square root, and trigonometric functions.*

Namespaces

- namespace **decaf**
Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.
- namespace **decaf::lang**

7.423 src/main/decaf/lang/Number.h File Reference

```
#include <decaf/util/Config.h>
```

Data Structures

- class **decaf::lang::Number**

The abstract class **Number** (p. 1992) is the superclass of classes **Byte** (p. 614), **Double** (p. 1235), **Float** (p. 1344), **Integer** (p. 1500), **Long** (p. 1726), and **Short** (p. 2398).

Namespaces

- namespace **decaf**

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

- namespace **decaf::lang**

7.424 src/main/decaf/lang/Pointer.h File Reference

```
#include <decaf/util/Config.h> #include <decaf/lang/exceptions/-
NullPointerException.h> #include <decaf/lang/exceptions/-
ClassCastException.h> #include <decaf/util/concurrent/atomic/-
AtomicRefCounter.h> #include <decaf/util/Comparator.h>
#include <memory> #include <typeinfo> #include <algorithm> ×
#include <functional>
```

Data Structures

- struct **decaf::lang::STATIC_CAST_TOKEN**
- struct **decaf::lang::DYNAMIC_CAST_TOKEN**
- class **decaf::lang::Pointer**< T, REFCOUNTER >

Decaf's implementation of a Smart **Pointer** (p. 2083) that is a template on a Type and is **Thread** (p. 2703) Safe if the default Reference Counter is used.

- class **decaf::lang::PointerComparator**< T, R >

This implementation of Comparator is designed to allows objects in a Collection to be sorted or tested for equality based on the value of the Object being Pointed to and not the value of the contained pointer in the **Pointer** (p. 2083) instance.

- struct **std::less**< **decaf::lang::Pointer**< T > >

An override of the less function object so that the Pointer objects can be stored in STL Maps, etc.

Namespaces

- namespace **decaf**
Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.
- namespace **decaf::lang**
- namespace **std**

Functions

- template<typename T, typename R, typename U >
bool **decaf::lang::operator==** (const Pointer< T, R > &left, const U *right)
- template<typename T, typename R, typename U >
bool **decaf::lang::operator==** (const U *left, const Pointer< T, R > &right)
- template<typename T, typename R, typename U >
bool **decaf::lang::operator!=** (const Pointer< T, R > &left, const U *right)
- template<typename T, typename R, typename U >
bool **decaf::lang::operator!=** (const U *left, const Pointer< T, R > &right)

7.425 src/main/decaf/lang/Readable.h File Reference

```
#include <decaf/util/Config.h>      #include <decaf/io/IO-
Exception.h> #include <decaf/lang/exceptions/NullPointer-
Exception.h> #include <decaf/nio/ReadOnlyBufferException.-
h>
```

Data Structures

- class **decaf::lang::Readable**
*A **Readable** (p. 2235) is a source of characters.*

Namespaces

- namespace **decaf**
Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.
- namespace **decaf::nio**
- namespace **decaf::lang**

7.426 src/main/decaf/lang/Runnable.h File Reference

```
#include <decaf/util/Config.h>
```

Data Structures

- class **decaf::lang::Runnable**

Interface for a runnable object - defines a task that can be run by a thread.

Namespaces

- namespace **decaf**

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

- namespace **decaf::lang**

7.427 src/main/decaf/lang/Runtime.h File Reference

```
#include <decaf/util/Config.h>
```

Data Structures

- class **decaf::lang::Runtime**

Namespaces

- namespace **decaf**

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

- namespace **decaf::lang**

7.428 src/main/decaf/lang/Short.h File Reference

```
#include <decaf/util/Config.h>      #include <decaf/lang/-  
Number.h>      #include <decaf/lang/Comparable.h>      #include  
<decaf/lang/exceptions/NumberFormatException.h>      #include  
<string>
```

Data Structures

- class **decaf::lang::Short**

Namespaces

- namespace **decaf**

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

- namespace **decaf::lang**

7.429 src/main/decaf/lang/String.h File Reference

```
#include <decaf/util/Config.h> #include <decaf/lang/Char-Sequence.h> #include <decaf/lang/Comparable.h> #include <string>
```

Data Structures

- class **decaf::lang::String**

*The **String** (p. 2620) class represents an immutable sequence of chars.*

Namespaces

- namespace **decaf**

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

- namespace **decaf::lang**

7.430 src/main/decaf/lang/System.h File Reference

```
#include <decaf/util/Config.h> #include <decaf/util/Map.-h> #include <decaf/util/Properties.h> #include <decaf/lang/-Exception.h> #include <decaf/lang/exceptions/NullPointerException.h> #include <decaf/internal/AprPool.h> #include <string>
```

Data Structures

- class **decaf::lang::System**

*The **System** (p. 2665) class provides static methods for accessing system level resources and performing some system dependent tasks such as looking up environment values and copying memory and arrays.*

Namespaces

- namespace **decaf**

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

- namespace **decaf::lang**

7.431 src/main/decaf/lang/Thread.h File Reference

```
#include <decaf/lang/exceptions/IllegalThreadStateException.-
h> #include <decaf/lang/exceptions/IllegalArgumentException.-
h> #include <decaf/lang/exceptions/InterruptedException.-
h> #include <decaf/lang/exceptions/RuntimeException.h> ×
#include <decaf/lang/Exception.h> #include <decaf/lang/-
Runnable.h> #include <decaf/util/Config.h>
```

Data Structures

- class **decaf::lang::Thread**

*A **Thread** (p. 2703) is a concurrent unit of execution.*

- class **decaf::lang::Thread::UncaughtExceptionHandler**

*Interface for handlers invoked when a **Thread** (p. 2703) abruptly terminates due to an uncaught exception.*

Namespaces

- namespace **decaf**

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

- namespace **decaf::util**
- namespace **decaf::util::concurrent**
- namespace **decaf::util::concurrent::locks**
- namespace **decaf::lang**

7.432 src/main/decaf/lang/ThreadGroup.h File Reference

```
#include <decaf/util/Config.h>
```

Data Structures

- class **decaf::lang::ThreadGroup**

Namespaces

- namespace **decaf**

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

- namespace **decaf::lang**

7.433 src/main/decaf/lang/Throwable.h File Reference

```
#include <string> #include <vector> #include <iostream> ×  
#include <exception> #include <decaf/util/Config.h>
```

Data Structures

- class **decaf::lang::Throwable**

This class represents an error that has occurred.

Namespaces

- namespace **decaf**

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

- namespace **decaf::lang**

7.434 src/main/decaf/net/BindException.h File Reference

```
#include <decaf/util/Config.h> #include <decaf/net/-  
SocketException.h>
```

Data Structures

- class **decaf::net::BindException**

Namespaces

- namespace **decaf**

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

- namespace **decaf::net**

7.435 src/main/decaf/net/ConnectException.h File Reference

```
#include <decaf/util/Config.h>          #include <decaf/net/-  
SocketException.h>
```

Data Structures

- class **decaf::net::ConnectException**

Namespaces

- namespace **decaf**
Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.
- namespace **decaf::net**

7.436 src/main/decaf/net/DatagramPacket.h File Reference

```
#include <decaf/util/Config.h> #include <decaf/net/Inet4-  
Address.h> #include <decaf/net/SocketAddress.h> #include  
<decaf/lang/exceptions/NullPointerException.h> #include  
<decaf/lang/exceptions/IndexOutOfBoundsException.h> ×  
#include <decaf/lang/exceptions/IllegalArgumentException.-  
h> #include <decaf/util/concurrent/Mutex.h>
```

Data Structures

- class **decaf::net::DatagramPacket**
Class that represents a single datagram packet.

Namespaces

- namespace **decaf**
Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.
- namespace **decaf::net**

7.437 src/main/decaf/net/HttpRetryException.h File Reference

```
#include <decaf/util/Config.h>          #include <decaf/io/IO-  
Exception.h>
```

Data Structures

- class **decaf::net::HttpRetryException**

Namespaces

- namespace **decaf**
Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.
- namespace **decaf::net**

7.438 src/main/decaf/net/Inet4Address.h File Reference

```
#include <decaf/util/Config.h> #include <decaf/net/Inet-  
Address.h>
```

Data Structures

- class **decaf::net::Inet4Address**

Namespaces

- namespace **decaf**
Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.
- namespace **decaf::net**

7.439 src/main/decaf/net/Inet6Address.h File Reference

```
#include <decaf/util/Config.h> #include <decaf/net/Inet-  
Address.h>
```

Data Structures

- class **decaf::net::Inet6Address**

Namespaces

- namespace **decaf**
Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.
- namespace **decaf::net**

7.440 src/main/decaf/net/InetAddress.h File Reference

```
#include <decaf/util/Config.h>      #include <decaf/lang/-  
ArrayPointer.h>
```

Data Structures

- class **decaf::net::InetAddress**

Represents an IP address.

Namespaces

- namespace **decaf**

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

- namespace **decaf::net**

7.441 src/main/decaf/net/InetSocketAddress.h File Reference

```
#include <decaf/util/Config.h>      #include <decaf/net/-  
SocketAddress.h> #include <string>
```

Data Structures

- class **decaf::net::InetSocketAddress**

Namespaces

- namespace **decaf**

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

- namespace **decaf::net**

7.442 src/main/decaf/net/MalformedURLException.h File Reference

```
#include <decaf/util/Config.h>      #include <decaf/io/IO-  
Exception.h>
```

Data Structures

- class **decaf::net::MalformedURLException**

Namespaces

- namespace **decaf**

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

- namespace **decaf::net**

7.443 src/main/decaf/net/NoRouteToHostException.h File Reference

```
#include <decaf/util/Config.h>      #include <decaf/net/-  
SocketException.h>
```

Data Structures

- class **decaf::net::NoRouteToHostException**

Namespaces

- namespace **decaf**

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

- namespace **decaf::net**

7.444 src/main/decaf/net/PortUnreachableException.h File Reference

```
#include <decaf/util/Config.h>      #include <decaf/net/-  
SocketException.h>
```

Data Structures

- class **decaf::net::PortUnreachableException**

Namespaces

- namespace **decaf**

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

- namespace **decaf::net**

7.445 src/main/decaf/net/ProtocolException.h File Reference

```
#include <decaf/util/Config.h>      #include <decaf/io/IO-  
Exception.h>
```

Data Structures

- class **decaf::net::ProtocolException**

Namespaces

- namespace **decaf**
Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.
- namespace **decaf::net**

7.446 src/main/decaf/net/ServerSocket.h File Reference

```
#include <decaf/util/Config.h> #include <decaf/net/Inet-  
Address.h> #include <decaf/net/SocketImpl.h> #include  
<decaf/net/SocketImplFactory.h> #include <decaf/lang/exceptions/-  
NullPointerException.h> #include <decaf/lang/exceptions/-  
IllegalArgumentException.h> #include <decaf/net/Unknown-  
HostException.h> #include <decaf/net/SocketTimeoutException.-  
h> #include <decaf/io/IOException.h> #include <string>
```

Data Structures

- class **decaf::net::ServerSocket**
This class implements server sockets.

Namespaces

- namespace **decaf**
Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.
- namespace **decaf::net**

7.447 src/main/decaf/net/ServerSocketFactory.h File Reference

```
#include <decaf/util/Config.h> #include <decaf/net/Inet-  
Address.h>
```

Data Structures

- class **decaf::net::ServerSocketFactory**

Class used to create Server Sockets, subclasses can be created that create certain types of server sockets according to specific policies.

Namespaces

- namespace **decaf**

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

- namespace **decaf::net**

7.448 src/main/decaf/net/Socket.h File Reference

```
#include <decaf/net/InetAddress.h> #include <decaf/net/-  
SocketImplFactory.h> #include <decaf/net/SocketException.-  
h> #include <decaf/io/InputStream.h> #include <decaf/io/-  
OutputStream.h> #include <decaf/io/Closeable.h> #include  
<decaf/util/Config.h> #include <decaf/lang/exceptions/-  
NullPointerException.h> #include <decaf/lang/exceptions/-  
IllegalArgumentException.h> #include <decaf/net/Unknown-  
HostException.h> #include <decaf/net/SocketTimeoutException.-  
h> #include <decaf/io/IOException.h>
```

Data Structures

- class **decaf::net::Socket**

Namespaces

- namespace **decaf**

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

- namespace **decaf::net**

7.449 src/main/decaf/net/SocketAddress.h File Reference

```
#include <decaf/util/Config.h>
```

Data Structures

- class **decaf::net::SocketAddress**

*Base class for protocol specific **Socket** (p. 2452) addresses.*

Namespaces

- namespace **decaf**

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

- namespace **decaf::net**

7.450 src/main/decaf/net/SocketError.h File Reference

```
#include <string> #include <decaf/util/Config.h>
```

Data Structures

- class **decaf::net::SocketError**

Static utility class to simplify handling of error codes for socket operations.

Namespaces

- namespace **decaf**

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

- namespace **decaf::net**

7.451 src/main/decaf/net/SocketException.h File Reference

```
#include <decaf/io/IOException.h>
```

Data Structures

- class **decaf::net::SocketException**

Exception for errors when manipulating sockets.

Namespaces

- namespace **decaf**

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

- namespace **decaf::net**

7.452 src/main/decaf/net/SocketFactory.h File Reference

```
#include <decaf/util/Config.h>      #include <decaf/io/IO-  
Exception.h>    #include <decaf/net/UnknownHostException.-  
h>
```

Data Structures

- class **decaf::net::SocketFactory**

*The **SocketFactory** (p. 2473) is used to create **Socket** (p. 2452) objects and can be sub-classed to provide other types of Sockets or Sockets with varying configurations.*

Namespaces

- namespace **decaf**

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

- namespace **decaf::net**

7.453 src/main/decaf/net/SocketImpl.h File Reference

```
#include <decaf/util/Config.h>      #include <decaf/io/IO-  
Exception.h>    #include <decaf/io/InputStream.h>    #include  
<decaf/io/OutputStream.h> #include <decaf/io/FileDescriptor.-  
h> #include <decaf/net/SocketException.h> #include <decaf/net/-  
SocketTimeoutException.h>    #include <decaf/net/Socket-  
Options.h> #include <string>
```

Data Structures

- class **decaf::net::SocketImpl**

*Acts as a base class for all physical **Socket** (p. 2452) implementations.*

Namespaces

- namespace **decaf**
Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.
- namespace **decaf::net**

7.454 src/main/decaf/net/SocketImplFactory.h File Reference

```
#include <decaf/util/Config.h>
```

Data Structures

- class **decaf::net::SocketImplFactory**
Factory class interface for a Factory that creates SocketImpl objects.

Namespaces

- namespace **decaf**
Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.
- namespace **decaf::net**

7.455 src/main/decaf/net/SocketOptions.h File Reference

```
#include <decaf/util/Config.h>
```

Data Structures

- class **decaf::net::SocketOptions**

Namespaces

- namespace **decaf**
Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.
- namespace **decaf::net**

7.456 src/main/decaf/net/SocketTimeoutException.h File Reference

```
#include <decaf/util/Config.h> #include <decaf/io/Interrupted-  
IOException.h>
```

Data Structures

- class **decaf::net::SocketTimeoutException**

Namespaces

- namespace **decaf**
Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.
- namespace **decaf::net**

7.457 src/main/decaf/net/ssl/SSLContext.h File Reference

```
#include <decaf/util/Config.h> #include <decaf/net/ssl/SSLContextSpi.h>
```

Data Structures

- class **decaf::net::ssl::SSLContext**
*Represents an implementation of the Secure **Socket** (p. 2452) Layer for streaming based sockets.*

Namespaces

- namespace **decaf**
Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.
- namespace **decaf::net**
- namespace **decaf::net::ssl**

7.458 src/main/decaf/net/ssl/SSLContextSpi.h File Reference

```
#include <decaf/util/Config.h> #include <decaf/security/SecureRandom.h>
```

Data Structures

- class **decaf::net::ssl::SSLContextSpi**
*Defines the interface that should be provided by an **SSLContext** (p. 2495) provider.*

Namespaces

- namespace **decaf**
Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.
- namespace **decaf::net**
- namespace **decaf::net::ssl**

7.459 src/main/decaf/net/ssl/SSLParameters.h File Reference

```
#include <decaf/util/Config.h> #include <string> #include  
<vector>
```

Data Structures

- class **decaf::net::ssl::SSLParameters**

Namespaces

- namespace **decaf**
Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.
- namespace **decaf::net**
- namespace **decaf::net::ssl**

7.460 src/main/decaf/net/ssl/SSLServerSocket.h File Reference

```
#include <decaf/util/Config.h> #include <decaf/net/-  
ServerSocket.h>
```

Data Structures

- class **decaf::net::ssl::SSLServerSocket**
Represents a server socket that is used to accept connections from clients using the Secure Sockets protocol or the Top Level Security protocol.

Namespaces

- namespace **decaf**
Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.
- namespace **decaf::net**
- namespace **decaf::net::ssl**

7.461 src/main/decaf/net/ssl/SSLServerSocketFactory.h File Reference

```
#include <decaf/util/Config.h>          #include <decaf/net/-  
ServerSocketFactory.h> #include <vector> #include <string> x
```

Data Structures

- class **decaf::net::ssl::SSLServerSocketFactory**
Factory class interface that provides methods to create SSL Server Sockets.

Namespaces

- namespace **decaf**
Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.
- namespace **decaf::net**
- namespace **decaf::net::ssl**

7.462 src/main/decaf/net/ssl/SSLSocket.h File Reference

```
#include <decaf/util/Config.h>          #include <decaf/net/-  
Socket.h> #include <decaf/net/ssl/SSLParameters.h>
```

Data Structures

- class **decaf::net::ssl::SSLSocket**

Namespaces

- namespace **decaf**
Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.
- namespace **decaf::net**
- namespace **decaf::net::ssl**

7.463 src/main/decaf/net/ssl/SSLSocketFactory.h File Reference

```
#include <decaf/util/Config.h>          #include <decaf/net/-  
SocketFactory.h> #include <vector> #include <string>
```

Data Structures

- class **decaf::net::ssl::SSLSocketFactory**

*Factory class interface for a **SocketFactory** (p. 2473) that can create **SSLSocket** (p. 2513) objects.*

Namespaces

- namespace **decaf**

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

- namespace **decaf::net**
- namespace **decaf::net::ssl**

7.464 src/main/decaf/net/UnknownHostException.h File Reference

```
#include <decaf/util/Config.h>      #include <decaf/io/IO-
Exception.h>
```

Data Structures

- class **decaf::net::UnknownHostException**

Namespaces

- namespace **decaf**

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

- namespace **decaf::net**

7.465 src/main/decaf/net/UnknownServiceException.h File Reference

```
#include <decaf/util/Config.h>      #include <decaf/io/IO-
Exception.h>
```

Data Structures

- class **decaf::net::UnknownServiceException**

Namespaces

- namespace **decaf**
Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.
- namespace **decaf::net**

7.466 src/main/decaf/net/URI.h File Reference

```
#include <decaf/util/Config.h>      #include <decaf/lang/-  
Comparable.h>      #include <decaf/lang/exceptions/Illegal-  
ArgumentException.h> #include <decaf/net/URISyntaxException.-  
h> #include <decaf/net/MalformedURLException.h> #include  
<decaf/net/URL.h> #include <decaf/internal/net/URIType.-  
h> #include <string>
```

Data Structures

- class **decaf::net::URI**
*This class represents an instance of a **URI** (p. 2828) as defined by RFC 2396.*

Namespaces

- namespace **decaf**
Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.
- namespace **decaf::net**

7.467 src/main/decaf/net/URISyntaxException.h File Reference

```
#include <decaf/util/Config.h>      #include <decaf/lang/-  
Exception.h>
```

Data Structures

- class **decaf::net::URISyntaxException**

Namespaces

- namespace **decaf**
Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.
- namespace **decaf::net**

7.468 src/main/decaf/net/URL.h File Reference

```
#include <decaf/util/Config.h> #include <string>
```

Data Structures

- class **decaf::net::URL**

*Class **URL** (p. 2868) represents a Uniform Resource Locator, a pointer to a "resource" on the World Wide Web.*

Namespaces

- namespace **decaf**

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

- namespace **decaf::net**

7.469 src/main/decaf/net/URLDecoder.h File Reference

```
#include <decaf/util/Config.h> #include <string>
```

Data Structures

- class **decaf::net::URLDecoder**

Namespaces

- namespace **decaf**

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

- namespace **decaf::net**

7.470 src/main/decaf/net/URLEncoder.h File Reference

```
#include <decaf/util/Config.h> #include <string>
```

Data Structures

- class **decaf::net::URLEncoder**

Namespaces

- namespace **decaf**

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

- namespace **decaf::net**

7.471 src/main/decaf/nio/Buffer.h File Reference

```
#include <decaf/lang/exceptions/IllegalArgumentException.-  
h> #include <decaf/nio/InvalidMarkException.h>
```

Data Structures

- class **decaf::nio::Buffer**

A container for data of a specific primitive type.

Namespaces

- namespace **decaf**

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

- namespace **decaf::nio**

7.472 src/main/decaf/nio/BufferOverflowException.h File Reference

```
#include <decaf/util/Config.h>      #include <decaf/lang/-  
Exception.h>
```

Data Structures

- class **decaf::nio::BufferOverflowException**

Namespaces

- namespace **decaf**

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

- namespace **decaf::nio**

7.473 src/main/decaf/nio/BufferUnderflowException.h File Reference

```
#include <decaf/util/Config.h>      #include <decaf/lang/-  
Exception.h>
```

Data Structures

- class **decaf::nio::BufferUnderflowException**

Namespaces

- namespace **decaf**
Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.
- namespace **decaf::nio**

7.474 src/main/decaf/nio/ByteBuffer.h File Reference

```
#include <decaf/nio/Buffer.h>      #include <decaf/lang/-  
Comparable.h> #include <decaf/lang/exceptions/NullPointerException.h> #include <decaf/lang/exceptions/IndexOutOfBoundsException.h> #include <decaf/nio/BufferUnderflowException.h> #include <decaf/nio/BufferOverflowException.h> #include <decaf/nio/ReadOnlyBufferException.h>
```

Data Structures

- class **decaf::nio::ByteBuffer**
This class defines six categories of operations upon byte buffers:

Namespaces

- namespace **decaf**
Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.
- namespace **decaf::nio**

7.475 src/main/decaf/nio/CharBuffer.h File Reference

```
#include <decaf/nio/Buffer.h>      #include <decaf/lang/-  
Comparable.h> #include <decaf/lang/exceptions/NullPointerException.h> #include <decaf/lang/exceptions/IndexOutOfBoundsException.h>
```

```
BoundsException.h>    #include <decaf/nio/BufferUnderflow-  
Exception.h> #include <decaf/nio/BufferOverflowException.-  
h> #include <decaf/nio/ReadOnlyBufferException.h> #include  
<decaf/lang/CharSequence.h> #include <decaf/lang/Appendable.-  
h>
```

Data Structures

- class **decaf::nio::CharBuffer**

This class defines four categories of operations upon character buffers:

Namespaces

- namespace **decaf**

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

- namespace **decaf::nio**

7.476 src/main/decaf/nio/DoubleBuffer.h File Reference

```
#include <decaf/nio/Buffer.h>    #include <decaf/lang/-  
Comparable.h> #include <decaf/lang/exceptions/NullPointer-  
Exception.h> #include <decaf/lang/exceptions/IndexOutOf-  
BoundsException.h> #include <decaf/nio/BufferUnderflow-  
Exception.h> #include <decaf/nio/BufferOverflowException.-  
h> #include <decaf/nio/ReadOnlyBufferException.h>
```

Data Structures

- class **decaf::nio::DoubleBuffer**

This class defines four categories of operations upon double buffers:

Namespaces

- namespace **decaf**

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

- namespace **decaf::nio**

7.477 src/main/decaf/nio/FloatBuffer.h File Reference

```
#include <decaf/nio/Buffer.h>          #include <decaf/lang/-  
Comparable.h> #include <decaf/lang/exceptions/NullPointer-  
Exception.h> #include <decaf/lang/exceptions/IndexOutOf-  
BoundsException.h> #include <decaf/nio/BufferUnderflow-  
Exception.h> #include <decaf/nio/BufferOverflowException.-  
h> #include <decaf/nio/ReadOnlyBufferException.h>
```

Data Structures

- class **decaf::nio::FloatBuffer**

This class defines four categories of operations upon float buffers:

Namespaces

- namespace **decaf**

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

- namespace **decaf::nio**

7.478 src/main/decaf/nio/IntBuffer.h File Reference

```
#include <decaf/nio/Buffer.h>          #include <decaf/lang/-  
Comparable.h> #include <decaf/lang/exceptions/NullPointer-  
Exception.h> #include <decaf/lang/exceptions/IndexOutOf-  
BoundsException.h> #include <decaf/nio/BufferUnderflow-  
Exception.h> #include <decaf/nio/BufferOverflowException.-  
h> #include <decaf/nio/ReadOnlyBufferException.h>
```

Data Structures

- class **decaf::nio::IntBuffer**

This class defines four categories of operations upon int buffers:

Namespaces

- namespace **decaf**

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

- namespace **decaf::nio**

7.479 src/main/decaf/nio/InvalidMarkException.h File Reference

```
#include <decaf/util/Config.h> #include <decaf/lang/exceptions/-  
IllegalStateException.h>
```

Data Structures

- class **decaf::nio::InvalidMarkException**

Namespaces

- namespace **decaf**
*Licensed to the Apache Software Foundation (ASF) under one or more contributor
license agreements.*
- namespace **decaf::nio**

7.480 src/main/decaf/nio/LongBuffer.h File Reference

```
#include <decaf/nio/Buffer.h>          #include <decaf/lang/-  
Comparable.h> #include <decaf/lang/exceptions/NullPointer-  
Exception.h> #include <decaf/lang/exceptions/IndexOutOf-  
BoundsException.h> #include <decaf/nio/BufferUnderflow-  
Exception.h> #include <decaf/nio/BufferOverflowException.-  
h> #include <decaf/nio/ReadOnlyBufferException.h>
```

Data Structures

- class **decaf::nio::LongBuffer**
This class defines four categories of operations upon long long buffers:

Namespaces

- namespace **decaf**
*Licensed to the Apache Software Foundation (ASF) under one or more contributor
license agreements.*
- namespace **decaf::nio**

7.481 src/main/decaf/nio/ReadOnlyBufferException.h File Reference

```
#include <decaf/util/Config.h> #include <decaf/lang/exceptions/-  
UnsupportedOperationException.h>
```

Data Structures

- class **decaf::nio::ReadOnlyBufferException**

Namespaces

- namespace **decaf**
Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.
- namespace **decaf::nio**

7.482 src/main/decaf/nio/ShortBuffer.h File Reference

```
#include <decaf/nio/Buffer.h>          #include <decaf/lang/-
Comparable.h> #include <decaf/lang/exceptions/NullPointerException.h>
#include <decaf/lang/exceptions/IndexOutOfBoundsException.h>
#include <decaf/nio/BufferUnderflowException.h> #include <decaf/nio/BufferOverflowException.h>
#include <decaf/nio/ReadOnlyBufferException.h>
```

Data Structures

- class **decaf::nio::ShortBuffer**
This class defines four categories of operations upon short buffers:

Namespaces

- namespace **decaf**
Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.
- namespace **decaf::nio**

7.483 src/main/decaf/security/auth/x500/X500Principal.h File Reference

```
#include <string> #include <vector> #include <decaf/security/-
Principal.h> #include <decaf/util/Map.h> #include <decaf/io/-
InputStream.h>
```

Data Structures

- class **decaf::security::auth::x500::X500Principal**

Namespaces

- namespace **decaf**
Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.
- namespace **decaf::security**
- namespace **decaf::security::auth**
- namespace **decaf::security::auth::x500**

7.484 src/main/decaf/security/cert/Certificate.h File Reference

```
#include <vector> #include <decaf/util/Config.h> #include  
<decaf/security/InvalidKeyException.h> #include <decaf/security/-  
NoSuchAlgorithmException.h> #include <decaf/security/-  
SignatureException.h> #include <decaf/security/cert/-  
CertificateEncodingException.h> #include <decaf/security/cert/-  
CertificateException.h>
```

Data Structures

- class **decaf::security::cert::Certificate**
Base interface for all identity certificates.

Namespaces

- namespace **decaf**
Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.
- namespace **decaf::security**
- namespace **decaf::security::cert**

7.485 src/main/decaf/security/cert/CertificateEncodingException.h File Reference

```
#include <decaf/util/Config.h> #include <decaf/security/cert/-  
CertificateException.h>
```

Data Structures

- class **decaf::security::cert::CertificateEncodingException**

Namespaces

- namespace **decaf**
Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.
- namespace **decaf::security**
- namespace **decaf::security::cert**

7.486 src/main/decaf/security/cert/CertificateException.h File - Reference

```
#include <decaf/util/Config.h> #include <decaf/security/-  
GeneralSecurityException.h>
```

Data Structures

- class **decaf::security::cert::CertificateException**

Namespaces

- namespace **decaf**
Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.
- namespace **decaf::security**
- namespace **decaf::security::cert**

7.487 src/main/decaf/security/cert/CertificateExpiredException.h File Reference

```
#include <decaf/util/Config.h> #include <decaf/security/cert/-  
CertificateException.h>
```

Data Structures

- class **decaf::security::cert::CertificateExpiredException**

Namespaces

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

- namespace **decaf::security**
- namespace **decaf::security::cert**

7.488 src/main/decaf/security/cert/CertificateNotYetValidException.h File Reference

```
#include <decaf/util/Config.h> #include <decaf/security/cert/-  
CertificateException.h>
```

Data Structures

- class **decaf::security::cert::CertificateNotYetValidException**

Namespaces

- namespace **decaf**
*Licensed to the Apache Software Foundation (ASF) under one or more contributor
license agreements.*
- namespace **decaf::security**
- namespace **decaf::security::cert**

7.489 src/main/decaf/security/cert/CertificateParsingException.h File Reference

```
#include <decaf/util/Config.h> #include <decaf/security/cert/-  
CertificateException.h>
```

Data Structures

- class **decaf::security::cert::CertificateParsingException**

Namespaces

- namespace **decaf**
*Licensed to the Apache Software Foundation (ASF) under one or more contributor
license agreements.*
- namespace **decaf::security**
- namespace **decaf::security::cert**

7.490 src/main/decaf/security/cert/X509Certificate.h File Reference

```
#include <decaf/security/cert/Certificate.h>      #include  
<decaf/util/Config.h> #include <decaf/util/Date.h>
```

Data Structures

- class **decaf::security::cert::X509Certificate**

Base interface for all identity certificates.

Namespaces

- namespace **decaf**
Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.
- namespace **decaf::security**
- namespace **decaf::security::cert**

7.491 src/main/decaf/security/GeneralSecurityException.h File - Reference

```
#include <decaf/util/Config.h>      #include <decaf/lang/-  
Exception.h>
```

Data Structures

- class **decaf::security::GeneralSecurityException**

Namespaces

- namespace **decaf**
Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.
- namespace **decaf::security**

7.492 src/main/decaf/security/InvalidKeyException.h File Reference

```
#include <decaf/security/KeyException.h>
```

Data Structures

- class **decaf::security::InvalidKeyException**

Namespaces

- namespace **decaf**
Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.
- namespace **decaf::security**

7.493 src/main/decaf/security/Key.h File Reference

```
#include <vector> #include <string> #include <decaf/util/-  
Config.h>
```

Data Structures

- class **decaf::security::Key**
*The **Key** (p. 1598) interface is the top-level interface for all keys.*

Namespaces

- namespace **decaf**
Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.
- namespace **decaf::security**

7.494 src/main/decaf/security/KeyException.h File Reference

```
#include <decaf/util/Config.h> #include <decaf/security/-  
GeneralSecurityException.h>
```

Data Structures

- class **decaf::security::KeyException**

Namespaces

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

- namespace **decaf::security**

7.495 **src/main/decaf/security/KeyManagementException.h** File - Reference

```
#include <decaf/util/Config.h> #include <decaf/security/-  
KeyException.h>
```

Data Structures

- class **decaf::security::KeyManagementException**

Namespaces

- namespace **decaf**
Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.
- namespace **decaf::security**

7.496 **src/main/decaf/security/NoSuchAlgorithmException.h** File - Reference

```
#include <decaf/security/GeneralSecurityException.h>
```

Data Structures

- class **decaf::security::NoSuchAlgorithmException**

Namespaces

- namespace **decaf**
Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.
- namespace **decaf::security**

7.497 **src/main/decaf/security/NoSuchProviderException.h** File - Reference

```
#include <decaf/util/Config.h> #include <decaf/security/-  
GeneralSecurityException.h>
```


Data Structures

- class **decaf::security::NoSuchProviderException**

Namespaces

- namespace **decaf**
Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.
- namespace **decaf::security**

7.498 src/main/decaf/security/Principal.h File Reference

```
#include <decaf/util/Config.h>
```

Data Structures

- class **decaf::security::Principal**
Base interface for a principal, which can represent an individual or organization.

Namespaces

- namespace **decaf**
Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.
- namespace **decaf::security**

7.499 src/main/decaf/security/PublicKey.h File Reference

```
#include <decaf/security/Key.h>    #include <decaf/util/Config.h>
```

Data Structures

- class **decaf::security::PublicKey**
A public key.

Namespaces

- namespace **decaf**

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

- namespace **decaf::security**

7.500 src/main/decaf/security/SecureRandom.h File Reference

```
#include <decaf/util/Config.h>      #include <decaf/util/-  
Random.h>      #include <decaf/security/SecureRandomSpi.h> ×  
#include <memory>
```

Data Structures

- class **decaf::security::SecureRandom**

Namespaces

- namespace **decaf**

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

- namespace **decaf::security**

7.501 src/main/decaf/security/SecureRandomSpi.h File Reference

```
#include <decaf/util/Config.h>
```

Data Structures

- class **decaf::security::SecureRandomSpi**

Interface class used by Security Service Providers to implement a source of secure random bytes.

Namespaces

- namespace **decaf**

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

- namespace **decaf::security**

7.502 src/main/decaf/security/SignatureException.h File Reference

```
#include <decaf/util/Config.h> #include <decaf/security/-  
GeneralSecurityException.h>
```

Data Structures

- class **decaf::security::SignatureException**

Namespaces

- namespace **decaf**

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

- namespace **decaf::security**

7.503 src/main/decaf/util/AbstractCollection.h File Reference

```
#include <decaf/util/Config.h> #include <decaf/lang/exceptions/-  
UnsupportedOperationException.h> #include <decaf/lang/exceptions/-  
NullPointerException.h> #include <decaf/lang/exceptions/-  
IllegalArgumentException.h> #include <decaf/lang/Iterable.-  
h> #include <decaf/util/Iterator.h> #include <decaf/util/-  
Collection.h> #include <decaf/util/concurrent/Synchronizable.-  
h> #include <decaf/util/concurrent/Mutex.h> #include  
<memory>
```

Data Structures

- class **decaf::util::AbstractCollection**< E >

*This class provides a skeletal implementation of the **Collection** (p. 851) interface, to minimize the effort required to implement this interface.*

Namespaces

- namespace **decaf**

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

- namespace **decaf::util**

7.504 src/main/decaf/util/AbstractList.h File Reference

```
#include <decaf/util/Config.h> #include <decaf/util/NoSuchElementException.h> #include <decaf/lang/exceptions/UnsupportedOperationException.h> #include <decaf/lang/exceptions/NullPointerException.h> #include <decaf/lang/exceptions/IllegalArgumentException.h> #include <decaf/util/ConcurrentModificationException.h> #include <decaf/lang/Iterable.h> #include <decaf/util/Iterator.h> #include <decaf/util/List.h> #include <decaf/util/AbstractCollection.h> #include <memory>
```

Data Structures

- class **decaf::util::AbstractList**< E >

*This class provides a skeletal implementation of the **List** (p. 1658) interface to minimize the effort required to implement this interface backed by a "random access" data store (such as an array).*

- class **decaf::util::AbstractList**< E >::SimpleListIterator
- class **decaf::util::AbstractList**< E >::ConstSimpleListIterator

Namespaces

- namespace **decaf**

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

- namespace **decaf::util**

7.505 src/main/decaf/util/AbstractMap.h File Reference

```
#include <decaf/util/Config.h> #include <decaf/lang/exceptions/UnsupportedOperationException.h> #include <decaf/lang/exceptions/NullPointerException.h> #include <decaf/lang/exceptions/IllegalArgumentException.h> #include <decaf/util/Iterator.h> #include <decaf/util/Map.h> #include <decaf/util/Set.h> #include <memory>
```

Data Structures

- class **decaf::util::AbstractMap**< K, V, COMPARATOR >

*This class provides a skeletal implementation of the **Map** (p. 1768) interface, to minimize the effort required to implement this interface.*

Namespaces

- namespace **decaf**
Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.
- namespace **decaf::util**

7.506 src/main/decaf/util/AbstractQueue.h File Reference

```
#include <decaf/util/Config.h> #include <decaf/lang/exceptions/-  
UnsupportedOperationException.h> #include <decaf/lang/exceptions/-  
NullPointerException.h> #include <decaf/lang/exceptions/-  
IllegalArgumentException.h> #include <decaf/lang/exceptions/-  
IllegalStateException.h> #include <decaf/lang/Iterable.-  
h> #include <decaf/util/Iterator.h> #include <decaf/util/-  
Queue.h> #include <memory>
```

Data Structures

- class **decaf::util::AbstractQueue**< **E** >
*This class provides skeletal implementations of some **Queue** (p. 2222) operations.*

Namespaces

- namespace **decaf**
Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.
- namespace **decaf::util**

7.507 src/main/decaf/util/AbstractSequentialList.h File Reference

```
#include <decaf/util/Config.h> #include <decaf/lang/exceptions/-  
UnsupportedOperationException.h> #include <decaf/lang/exceptions/-  
NullPointerException.h> #include <decaf/lang/exceptions/-  
IllegalArgumentException.h> #include <decaf/lang/Iterable.-  
h> #include <decaf/util/Iterator.h> #include <decaf/util/-  
AbstractList.h> #include <memory>
```

Data Structures

- class **decaf::util::AbstractSequentialList**< **E** >
*This class provides a skeletal implementation of the **List** (p. 1658) interface to minimize the effort required to implement this interface backed by a "sequential access" data store (such as a linked list).*

Namespaces

- namespace **decaf**

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

- namespace **decaf::util**

7.508 src/main/decaf/util/AbstractSet.h File Reference

```
#include <decaf/util/Config.h> #include <decaf/lang/exceptions/-
UnsupportedOperationException.h> #include <decaf/lang/exceptions/-
NullPointerException.h> #include <decaf/lang/exceptions/-
IllegalArgumentException.h> #include <decaf/lang/Iterable.-
h> #include <decaf/util/Iterator.h> #include <decaf/util/-
Set.h> #include <memory>
```

Data Structures

- class **decaf::util::AbstractSet< E >**

*This class provides a skeletal implementation of the **Set** (p. 2397) interface to minimize the effort required to implement this interface.*

Namespaces

- namespace **decaf**

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

- namespace **decaf::util**

7.509 src/main/decaf/util/ArrayList.h File Reference

```
#include <memory> #include <decaf/util/NoSuchElement-
Exception.h> #include <decaf/lang/exceptions/Unsupported-
OperationException.h> #include <decaf/lang/exceptions/-
IndexOutOfBoundsException.h> #include <decaf/lang/System.-
h> #include <decaf/lang/Integer.h> #include <decaf/util/-
Config.h> #include <decaf/util/Iterator.h> #include <decaf/util/-
ListIterator.h> #include <decaf/util/List.h> #include
<decaf/util/AbstractList.h>
```

Data Structures

- class **decaf::util::ArrayList< E >**

Namespaces

- namespace **decaf**

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

- namespace **decaf::util**

7.510 src/main/decaf/util/Arrays.h File Reference

```
#include <decaf/lang/exceptions/NullPointerException.h>
#include <decaf/lang/exceptions/IllegalArgumentException.-
h> #include <decaf/lang/exceptions/IndexOutOfBoundsException.-
h>
```

Data Structures

- class **decaf::util::Arrays**

Namespaces

- namespace **decaf**

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

- namespace **decaf::util**

7.511 src/main/decaf/util/Collection.h File Reference

```
#include <decaf/util/Config.h> #include <decaf/lang/exceptions/-
UnsupportedOperationException.h> #include <decaf/lang/exceptions/-
NullPointerException.h> #include <decaf/lang/exceptions/-
IllegalArgumentException.h> #include <decaf/lang/Iterable.-
h> #include <decaf/util/Iterator.h> #include <decaf/util/concurrent/-
Synchronizable.h>
```

Data Structures

- class **decaf::util::Collection< E >**

The root interface in the collection hierarchy.

Namespaces

- namespace **decaf**

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

- namespace **decaf::util**

7.512 src/main/decaf/util/Comparator.h File Reference

```
#include <decaf/util/Config.h>
```

Data Structures

- class **decaf::util::Comparator**< **T** >

A comparison function, which imposes a total ordering on some collection of objects.

Namespaces

- namespace **decaf**

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

- namespace **decaf::util**

7.513 src/main/decaf/util/comparators/Less.h File Reference

```
#include <decaf/util/Comparator.h>
```

Data Structures

- class **decaf::util::comparators::Less**< **E** >

*Simple **Less** (p. 1612) **Comparator** (p. 888) that compares to elements to determine if the first is less than the second.*

Namespaces

- namespace **decaf**

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

- namespace **decaf::util**

- namespace **decaf::util::comparators**

7.514 src/main/decaf/util/concurrent/AbstractExecutorService.h - File Reference

```
#include <decaf/util/Config.h> #include <decaf/util/concurrent/-  
Executor.h>      #include <decaf/util/concurrent/Executor-  
Service.h>
```

Data Structures

- class **decaf::util::concurrent::AbstractExecutorService**
*Provides a default implementation for the methods of the **ExecutorService** (p. 1302) interface.*

Namespaces

- namespace **decaf**
Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.
- namespace **decaf::util**
- namespace **decaf::util::concurrent**

7.515 src/main/decaf/util/concurrent/atomic/AtomicBoolean.h File Reference

```
#include <string> #include <decaf/util/Config.h>
```

Data Structures

- class **decaf::util::concurrent::atomic::AtomicBoolean**
A boolean value that may be updated atomically.

Namespaces

- namespace **decaf**
Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.
- namespace **decaf::util**
- namespace **decaf::util::concurrent**
- namespace **decaf::util::concurrent::atomic**

7.516 src/main/decaf/util/concurrent/atomic/AtomicInteger.h File - Reference

```
#include <decaf/util/Config.h>      #include <decaf/lang/Number.h> #include <string>
```

Data Structures

- class **decaf::util::concurrent::atomic::AtomicInteger**

An int value that may be updated atomically.

Namespaces

- namespace **decaf**
Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.
- namespace **decaf::util**
- namespace **decaf::util::concurrent**
- namespace **decaf::util::concurrent::atomic**

7.517 src/main/decaf/util/concurrent/atomic/AtomicRefCounter.h File Reference

```
#include <decaf/util/concurrent/atomic/AtomicInteger.h>
```

Data Structures

- class **decaf::util::concurrent::atomic::AtomicRefCounter**

Namespaces

- namespace **decaf**
Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.
- namespace **decaf::util**
- namespace **decaf::util::concurrent**
- namespace **decaf::util::concurrent::atomic**

7.518 src/main/decaf/util/concurrent/atomic/AtomicReference.h - File Reference

```
#include <decaf/util/Config.h>      #include <decaf/lang/Long.h> #include <apr_atomic.h>
```

Data Structures

- class **decaf::util::concurrent::atomic::AtomicReference**< T >
An Pointer reference that may be updated atomically.

Namespaces

- namespace **decaf**
Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.
- namespace **decaf::util**
- namespace **decaf::util::concurrent**
- namespace **decaf::util::concurrent::atomic**

7.519 src/main/decaf/util/concurrent/BlockingQueue.h File - Reference

```
#include <decaf/util/Config.h>      #include <decaf/util/AbstractQueue.h>      #include <decaf/util/concurrent/TimeUnit.h> #include <decaf/lang/exceptions/InterruptedException.h>
```

Data Structures

- class **decaf::util::concurrent::BlockingQueue**< E >
*A **decaf::util::Queue** (p. 2222) that additionally supports operations that wait for the queue to become non-empty when retrieving an element, and wait for space to become available in the queue when storing an element.*

Namespaces

- namespace **decaf**
Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.
- namespace **decaf::util**
- namespace **decaf::util::concurrent**

7.520 src/main/decaf/util/concurrent/BrokenBarrierException.h - File Reference

```
#include <decaf/lang/Exception.h>
```

Data Structures

- class **decaf::util::concurrent::BrokenBarrierException**

Namespaces

- namespace **decaf**
Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.
- namespace **decaf::util**
- namespace **decaf::util::concurrent**

7.521 src/main/decaf/util/concurrent/Callable.h File Reference

```
#include <decaf/util/Config.h>      #include <decaf/lang/-  
Exception.h>
```

Data Structures

- class **decaf::util::concurrent::Callable< V >**
A task that returns a result and may throw an exception.

Namespaces

- namespace **decaf**
Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.
- namespace **decaf::util**
- namespace **decaf::util::concurrent**

7.522 src/main/decaf/util/concurrent/CancellationException.h File - Reference

```
#include <decaf/lang/Exception.h>
```

Data Structures

- class **decaf::util::concurrent::CancellationException**

Namespaces

- namespace **decaf**
Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.
- namespace **decaf::util**
- namespace **decaf::util::concurrent**

7.523 src/main/decaf/util/concurrent/Concurrent.h File Reference

```
#include <decaf/util/concurrent/Lock.h>
```

Namespaces

- namespace **decaf**
Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.
- namespace **decaf::util**
- namespace **decaf::util::concurrent**

Defines

- #define **WAIT_INFINITE** 0xFFFFFFFF
The synchronized macro defines a mechanism for synchronizing a section of code.
- #define **synchronized(W)**

7.523.1 Define Documentation

7.523.1.1 #define synchronized(W)

Value:

```
if(false){} \
else \
for( decaf::util::concurrent::Lock lock_W(W); \
    lock_W.isLocked(); lock_W.unlock() )
```

7.523.1.2 #define WAIT_INFINITE 0xFFFFFFFF

The synchronized macro defines a mechanism for synchronizing a section of code.

The macro must be passed an object that implements the Synchronizable interface.

The macro works by creating a for loop that will loop exactly once, creating a Lock object that is scoped to the loop. Once the loop completes and exits the Lock object goes out of scope releasing the lock on object W. For added safety the if else is used because not all compilers restrict the lifetime of loop variables to the loop, they will however restrict them to the scope of the else.

The macro would be used as follows.

Synchronizable X;

```
somefunction() { synchronized(X) (p. 3231) { // Do something that needs synchronizing.
} }
```

7.524 src/main/decaf/util/concurrent/ConcurrentMap.h File - Reference

```
#include <decaf/util/Config.h> #include <decaf/util/Map.-
h> #include <decaf/lang/exceptions/UnsupportedOperation-
Exception.h> #include <decaf/lang/exceptions/Illegal-
StateException.h> #include <decaf/util/NoSuchElement-
Exception.h>
```

Data Structures

- class **decaf::util::concurrent::ConcurrentMap**< K, V, COMPARATOR >
Interface for a **Map** (p. 1768) type that provides additional atomic putIfAbsent, remove, and replace methods alongside the already available **Map** (p. 1768) interface.

Namespaces

- namespace **decaf**
Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.
- namespace **decaf::util**
- namespace **decaf::util::concurrent**

7.525 src/main/decaf/util/concurrent/ConcurrentStlMap.h File - Reference

```
#include <map> #include <vector> #include <decaf/util/No-
```

7.526 src/main/decaf/util/concurrent/CopyOnWriteArrayList.h File Reference 3241

```
SuchElementException.h> #include <decaf/util/concurrent/-  
Synchronizable.h> #include <decaf/util/concurrent/Concurrent-  
Map.h> #include <decaf/util/concurrent/Mutex.h> #include  
<decaf/util/Map.h>
```

Data Structures

- class **decaf::util::concurrent::ConcurrentStlMap**< K, V, COMPARATOR >
Map (p. 1768) template that wraps around a `std::map` to provide a more user-friendly interface and to provide common functions that do not exist in `std::map`.

Namespaces

- namespace **decaf**
Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.
- namespace **decaf::util**
- namespace **decaf::util::concurrent**

7.526 src/main/decaf/util/concurrent/CopyOnWriteArrayList.h File - Reference

```
#include <decaf/util/NoSuchElementException.h> #include  
<decaf/lang/exceptions/IndexOutOfBoundsException.h> ×  
#include <decaf/util/concurrent/Synchronizable.h> #include  
<decaf/util/concurrent/Mutex.h> #include <decaf/lang/-  
ArrayPointer.h> #include <decaf/lang/System.h> #include  
<decaf/util/List.h>
```

Data Structures

- class **decaf::util::concurrent::CopyOnWriteArrayList**< E >
- class **decaf::util::concurrent::CopyOnWriteArrayList**< E >::ArrayList-
Iterator

Namespaces

- namespace **decaf**
Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.
- namespace **decaf::util**
- namespace **decaf::util::concurrent**

7.527 src/main/decaf/util/concurrent/CopyOnWriteArraySet.h File - Reference

```
#include <decaf/util/NoSuchElementException.h>    #include
<decaf/lang/exceptions/IndexOutOfBoundsException.h> ×
#include <decaf/lang/ArrayPointer.h> #include <decaf/util/concurrent/-
CopyOnWriteArrayList.h> #include <decaf/util/concurrent/-
Synchronizable.h> #include <decaf/util/Set.h> #include
<decaf/util/Arrays.h> #include <decaf/util/AbstractSet.-
h>
```

Data Structures

- class **decaf::util::concurrent::CopyOnWriteArraySet**< E >
*Since the **CopyOnWriteArraySet** (p. 1050) and the **CopyOnWriteArrayList** (p. 1030) share much of the same operational semantics this class uses the **CopyOnWriteArrayList** (p. 1030) for all its underlying operations.*

Namespaces

- namespace **decaf**
Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.
- namespace **decaf::util**
- namespace **decaf::util::concurrent**

7.528 src/main/decaf/util/concurrent/CountDownLatch.h File - Reference

```
#include <decaf/util/concurrent/Mutex.h> #include <decaf/util/concurrent/-
TimeUnit.h> #include <decaf/util/Config.h> #include <decaf/lang/exceptions/
InterruptedException.h> #include <decaf/lang/Exception.-
h>
```

Data Structures

- class **decaf::util::concurrent::CountDownLatch**

Namespaces

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

- namespace **decaf::util**
- namespace **decaf::util::concurrent**

7.529 src/main/decaf/util/concurrent/Delayed.h File Reference

```
#include <decaf/util/Config.h>      #include <decaf/lang/-  
Comparable.h>  #include <decaf/util/concurrent/TimeUnit.-  
h>
```

Data Structures

- class **decaf::util::concurrent::Delayed**
A mix-in style interface for marking objects that should be acted upon after a given delay.

Namespaces

- namespace **decaf**
Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.
- namespace **decaf::util**
- namespace **decaf::util::concurrent**

7.530 src/main/decaf/util/concurrent/ExecutionException.h File - Reference

```
#include <decaf/lang/Exception.h>
```

Data Structures

- class **decaf::util::concurrent::ExecutionException**

Namespaces

- namespace **decaf**
Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.
- namespace **decaf::util**
- namespace **decaf::util::concurrent**

7.531 src/main/decaf/util/concurrent/Executor.h File Reference

```
#include <decaf/util/Config.h>      #include <decaf/lang/-  
Runnable.h> #include <decaf/lang/exceptions/NullPointer-  
Exception.h> #include <decaf/util/concurrent/Rejected-  
ExecutionException.h>
```

Data Structures

- class **decaf::util::concurrent::Executor**

*An object that executes submitted **decaf.lang.Runnable** (p. 2312) tasks.*

Namespaces

- namespace **decaf**

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

- namespace **decaf::util**
- namespace **decaf::util::concurrent**

7.532 src/main/decaf/util/concurrent/Executors.h File Reference

```
#include <decaf/util/Config.h>      #include <decaf/lang/-  
Thread.h> #include <decaf/util/concurrent/ExecutorService.-  
h> #include <decaf/util/concurrent/ThreadFactory.h>
```

Data Structures

- class **decaf::util::concurrent::Executors**

*Implements a set of utilities for use with **Executors** (p. 1299), **ExecutorService** (p. 1302), **ThreadFactory** (p. 2714), and **Callable** (p. 746) types, as well as providing factory methods for instance of these types configured for the most common use cases.*

Namespaces

- namespace **decaf**

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

- namespace **decaf::util**
- namespace **decaf::util::concurrent**

7.533 src/main/decaf/util/concurrent/ExecutorService.h File Reference

```
#include <decaf/util/Config.h>      #include <decaf/lang/-
Runnable.h> #include <decaf/util/ArrayList.h> #include
<decaf/util/concurrent/Executor.h> #include <decaf/util/concurrent/-
TimeUnit.h> #include <decaf/lang/exceptions/InterruptedException.h>
```

Data Structures

- class **decaf::util::concurrent::ExecutorService**

An **Executor** (p. 1297) that provides methods to manage termination and methods that can produce a **Future** (p. 1390) for tracking progress of one or more asynchronous tasks.

Namespaces

- namespace **decaf**

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

- namespace **decaf::util**
- namespace **decaf::util::concurrent**

7.534 src/main/decaf/util/concurrent/Future.h File Reference

```
#include <decaf/util/concurrent/TimeUnit.h>
```

Data Structures

- class **decaf::util::concurrent::Future< V >**

A **Future** (p. 1390) represents the result of an asynchronous computation.

Namespaces

- namespace **decaf**

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

- namespace **decaf::util**
- namespace **decaf::util::concurrent**

7.535 src/main/decaf/util/concurrent/LinkedBlockingQueue.h File - Reference

```
#include <decaf/util/Config.h> #include <decaf/util/concurrent/atomic/-
AtomicInteger.h> #include <decaf/util/concurrent/Mutex.-
h> #include <decaf/util/concurrent/Lock.h> #include <decaf/util/concurrent/
BlockingQueue.h> #include <decaf/util/AbstractQueue.h> ×
#include <decaf/util/Iterator.h> #include <decaf/lang/-
Integer.h> #include <decaf/lang/Math.h> #include <decaf/lang/-
Pointer.h> #include <decaf/util/NoSuchElementException.-
h> #include <decaf/lang/exceptions/IllegalArgumentException.-
h> #include <decaf/lang/exceptions/IllegalStateException.-
h>
```

Data Structures

- class **decaf::util::concurrent::LinkedBlockingQueue< E >**
*A **BlockingQueue** (p. 538) derivative that allows for a bound to be placed on the number of elements that can be enqueued at any one time.*
- class **decaf::util::concurrent::LinkedBlockingQueue< E >::QueueNode< U >**
- class **decaf::util::concurrent::LinkedBlockingQueue< E >::TotalLock**
- class **decaf::util::concurrent::LinkedBlockingQueue< E >::LinkedIterator**
- class **decaf::util::concurrent::LinkedBlockingQueue< E >::ConstLinkedIterator**

Namespaces

- namespace **decaf**
Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.
- namespace **decaf::util**
- namespace **decaf::util::concurrent**

7.536 src/main/decaf/util/concurrent/Lock.h File Reference

```
#include <decaf/util/concurrent/Synchronizable.h> #include
<decaf/util/Config.h>
```

Data Structures

- class **decaf::util::concurrent::Lock**
A wrapper class around a given synchronization mechanism that provides automatic release upon destruction.

Namespaces

- namespace **decaf**
Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.
- namespace **decaf::util**
- namespace **decaf::util::concurrent**

7.537 src/main/decaf/util/concurrent/locks/Lock.h File Reference

```
#include <decaf/util/Config.h> #include <decaf/lang/exceptions/-  
InterruptedException.h> #include <decaf/lang/exceptions/-  
UnsupportedOperationException.h> #include <decaf/util/concurrent/locks/-  
Condition.h>
```

Data Structures

- class **decaf::util::concurrent::locks::Lock**
***Lock** (p. 1684) implementations provide more extensive locking operations than can be obtained using synchronized statements.*

Namespaces

- namespace **decaf**
Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.
- namespace **decaf::util**
- namespace **decaf::util::concurrent**
- namespace **decaf::util::concurrent::locks**

7.538 src/main/decaf/util/concurrent/locks/AbstractOwnableSynchronizer.h File Reference

```
#include <decaf/util/Config.h>
```

Data Structures

- class **decaf::util::concurrent::locks::AbstractOwnableSynchronizer**
Base class for locks that provide the notion of Ownership, the types of locks that are implemented using this base class would be owned by one specific Thread at any given time.

Namespaces

- namespace **decaf**

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

- namespace **decaf::lang**
- namespace **decaf::util**
- namespace **decaf::util::concurrent**
- namespace **decaf::util::concurrent::locks**

7.539 src/main/decaf/util/concurrent/locks/Condition.h File - Reference

```
#include <decaf/util/Config.h>      #include <decaf/util/-
Date.h>      #include <decaf/util/concurrent/TimeUnit.h> ×
#include <decaf/lang/exceptions/RuntimeException.h> ×
#include <decaf/lang/exceptions/InterruptedException.-
h>      #include <decaf/lang/exceptions/IllegalMonitorState-
Exception.h>
```

Data Structures

- class **decaf::util::concurrent::locks::Condition**

Condition (p. 921) factors out the **Mutex** (p. 1960) monitor methods (wait, notify and notifyAll) into distinct objects to give the effect of having multiple wait-sets per object, by combining them with the use of arbitrary **Lock** (p. 1684) implementations.

Namespaces

- namespace **decaf**

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

- namespace **decaf::util**
- namespace **decaf::util::concurrent**
- namespace **decaf::util::concurrent::locks**

7.540 src/main/decaf/util/concurrent/locks/LockSupport.h File - Reference

```
#include <decaf/util/Config.h>
```

7.541 src/main/decaf/util/concurrent/locks/ReadWriteLock.h File Reference 3249

Data Structures

- class **decaf::util::concurrent::locks::LockSupport**
Basic thread blocking primitives for creating locks and other synchronization classes.

Namespaces

- namespace **decaf**
Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.
- namespace **decaf::lang**
- namespace **decaf::util**
- namespace **decaf::util::concurrent**
- namespace **decaf::util::concurrent::locks**

7.541 src/main/decaf/util/concurrent/locks/ReadWriteLock.h File - Reference

Data Structures

- class **decaf::util::concurrent::locks::ReadWriteLock**
*A **ReadWriteLock** (p. 2246) maintains a pair of associated locks, one for read-only operations and one for writing.*

Namespaces

- namespace **decaf**
Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.
- namespace **decaf::util**
- namespace **decaf::util::concurrent**
- namespace **decaf::util::concurrent::locks**

7.542 src/main/decaf/util/concurrent/locks/ReentrantLock.h File - Reference

```
#include <decaf/util/Config.h> #include <decaf/util/concurrent/locks/-  
Lock.h>
```

Data Structures

- class **decaf::util::concurrent::locks::ReentrantLock**
*A reentrant mutual exclusion **Lock** (p. 1684) with extended capabilities.*

Namespaces

- namespace **decaf**

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

- namespace **decaf::util**
- namespace **decaf::util::concurrent**
- namespace **decaf::util::concurrent::locks**

7.543 src/main/decaf/util/concurrent/Mutex.h File Reference

```
#include <decaf/util/concurrent/Synchronizable.h> #include  
<decaf/util/concurrent/Concurrent.h> #include <decaf/lang/-  
Thread.h> #include <decaf/util/Config.h>
```

Data Structures

- class **decaf::util::concurrent::Mutex**

***Mutex** (p. 1960) object that offers recursive support on all platforms as well as providing the ability to use the standard wait / notify pattern used in languages like Java.*

Namespaces

- namespace **decaf**

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

- namespace **decaf::util**
- namespace **decaf::util::concurrent**

7.544 src/main/decaf/util/concurrent/RejectedExecutionException.h File Reference

```
#include <decaf/lang/Exception.h>
```

Data Structures

- class **decaf::util::concurrent::RejectedExecutionException**

7.545 src/main/decaf/util/concurrent/RejectedExecutionHandler.h File Reference 3251

Namespaces

- namespace **decaf**
Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.
- namespace **decaf::util**
- namespace **decaf::util::concurrent**

7.545 src/main/decaf/util/concurrent/RejectedExecutionHandler.h - File Reference

```
#include <decaf/util/Config.h>      #include <decaf/lang/-  
Runnable.h>      #include <decaf/util/concurrent/Rejected-  
ExecutionException.h>
```

Data Structures

- class **decaf::util::concurrent::RejectedExecutionHandler**
*A handler for tasks that cannot be executed by a **ThreadPoolExecutor** (p. 2715).*

Namespaces

- namespace **decaf**
Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.
- namespace **decaf::util**
- namespace **decaf::util::concurrent**

7.546 src/main/decaf/util/concurrent/Semaphore.h File Reference

```
#include <decaf/util/Config.h> #include <decaf/lang/exceptions/-  
InterruptedException.h> #include <decaf/lang/exceptions/-  
RuntimeException.h>      #include <decaf/lang/exceptions/-  
IllegalArgumentException.h> #include <decaf/util/concurrent/-  
TimeUnit.h> #include <memory>
```

Data Structures

- class **decaf::util::concurrent::Semaphore**
A counting semaphore.

Namespaces

- namespace **decaf**
Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.
- namespace **decaf::util**
- namespace **decaf::util::concurrent**

7.547 src/main/decaf/util/concurrent/Synchronizable.h File - Reference

```
#include <decaf/lang/exceptions/RuntimeException.h> ×
#include <decaf/lang/exceptions/InterruptedException.h>
#include <decaf/lang/exceptions/IllegalArgumentException.h>
#include <decaf/lang/exceptions/IllegalMonitorStateException.h> #include <decaf/util/Config.h>
```

Data Structures

- class **decaf::util::concurrent::Synchronizable**
The interface for all synchronizable objects (that is, objects that can be locked and unlocked).

Namespaces

- namespace **decaf**
Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.
- namespace **decaf::util**
- namespace **decaf::util::concurrent**

7.548 src/main/decaf/util/concurrent/SynchronousQueue.h File - Reference

```
#include <decaf/util/Config.h> #include <decaf/util/concurrent/BlockingQueue.h> #include <vector>
```

Data Structures

- class **decaf::util::concurrent::SynchronousQueue< E >**
*A **blocking queue** (p. 538) in which each insert operation must wait for a corresponding remove operation by another thread, and vice versa.*
- class **decaf::util::concurrent::SynchronousQueue< E >::EmptyIterator**

Namespaces

- namespace **decaf**
Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.
- namespace **decaf::util**
- namespace **decaf::util::concurrent**

7.549 src/main/decaf/util/concurrent/ThreadFactory.h File Reference

```
#include <decaf/util/Config.h>      #include <decaf/lang/-
Thread.h> #include <decaf/lang/Runnable.h>
```

Data Structures

- class **decaf::util::concurrent::ThreadFactory**
*public interface **ThreadFactory** (p. 2714)*

Namespaces

- namespace **decaf**
Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.
- namespace **decaf::util**
- namespace **decaf::util::concurrent**

7.550 src/main/decaf/util/concurrent/ThreadPoolExecutor.h File - Reference

```
#include <decaf/lang/Runnable.h>    #include <decaf/lang/-
Throwable.h> #include <decaf/util/concurrent/ThreadFactory.-
h>      #include <decaf/util/concurrent/BlockingQueue.h> ×
#include <decaf/util/concurrent/TimeUnit.h> #include <decaf/util/concurrent/-
AbstractExecutorService.h> #include <decaf/util/concurrent/-
RejectedExecutionHandler.h> #include <decaf/util/concurrent/-
RejectedExecutionException.h> #include <decaf/util/Linked-
List.h> #include <decaf/util/ArrayList.h> #include <decaf/util/-
Config.h> #include <vector>
```

Data Structures

- class **decaf::util::concurrent::ThreadPoolExecutor**

Defines a Thread Pool object that implements the functionality of pooling threads to perform user tasks.

- class **decaf::util::concurrent::ThreadPoolExecutor::AbortPolicy**
*Handler policy for tasks that are rejected upon a call to **ThreadPoolExecutor::execute** (p. 2724) this class always throws a **RejectedExecutionException** (p. 2263).*
- class **decaf::util::concurrent::ThreadPoolExecutor::CallerRunsPolicy**
*Handler policy for tasks that are rejected upon a call to **ThreadPoolExecutor::execute** (p. 2724) this class will attempt to run the task in the Thread that called the execute method unless the executor is shutdown in which case the task is not run and is destroyed.*
- class **decaf::util::concurrent::ThreadPoolExecutor::CallerRunsPolicy:-DiscardPolicy**
*Handler policy for tasks that are rejected upon a call to **ThreadPoolExecutor::execute** (p. 2724) this class always destroys the rejected task and returns quietly.*
- class **decaf::util::concurrent::ThreadPoolExecutor::CallerRunsPolicy:-DiscardOldestPolicy**
*Handler policy for tasks that are rejected upon a call to **ThreadPoolExecutor::execute** (p. 2724) this class always destroys the oldest unexecuted task in the **Queue** (p. 2222) and then attempts to execute the rejected task using the passed in executor.*

Namespaces

- namespace **decaf**
Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.
- namespace **decaf::util**
- namespace **decaf::util::concurrent**

7.551 src/main/decaf/util/concurrent/TimeoutException.h File - Reference

```
#include <decaf/lang/Exception.h>
```

Data Structures

- class **decaf::util::concurrent::TimeoutException**

Namespaces

- namespace **decaf**
Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.
- namespace **decaf::util**
- namespace **decaf::util::concurrent**

7.552 src/main/decaf/util/concurrent/TimeUnit.h File Reference

```
#include <string> #include <decaf/util/Config.h> #include  
<decaf/lang/Comparable.h> #include <decaf/util/concurrent/-  
Synchronizable.h> #include <decaf/lang/exceptions/Illegal-  
ArgumentException.h> #include <decaf/lang/exceptions/-  
InterruptedException.h> #include <decaf/lang/exceptions/-  
NullPointerException.h>
```

Data Structures

- class **decaf::util::concurrent::TimeUnit**

*A **TimeUnit** (p. 2756) represents time durations at a given unit of granularity and provides utility methods to convert across units, and to perform timing and delay operations in these units.*

Namespaces

- namespace **decaf**
Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.
- namespace **decaf::lang**
- namespace **decaf::util**
- namespace **decaf::util::concurrent**

7.553 src/main/decaf/util/ConcurrentModificationException.h File - Reference

```
#include <decaf/util/Config.h> #include <decaf/lang/exceptions/-  
RuntimeException.h>
```

Data Structures

- class **decaf::util::ConcurrentModificationException**

Namespaces

- namespace **decaf**
Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.
- namespace **decaf::util**

7.554 src/main/decaf/util/Date.h File Reference

```
#include <decaf/util/Config.h>      #include <decaf/lang/-  
Comparable.h> #include <string>
```

Data Structures

- class **decaf::util::Date**
Wrapper class around a time value in milliseconds.

Namespaces

- namespace **decaf**
Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.
- namespace **decaf::util**

7.555 src/main/decaf/util/Deque.h File Reference

```
#include <decaf/util/NoSuchElementException.h>  #include  
<decaf/lang/exceptions/NullPointerException.h>  #include  
<decaf/lang/exceptions/IllegalArgumentException.h> #include  
<decaf/lang/exceptions/IllegalStateException.h> #include  
<decaf/util/Config.h> #include <decaf/util/Queue.h>
```

Data Structures

- class **decaf::util::Deque< E >**
*Defines a 'Double ended **Queue** (p. 2222)' interface that allows for insertion and removal of elements from both ends.*

Namespaces

- namespace **decaf**
Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.
- namespace **decaf::util**

7.556 src/main/decaf/util/Iterator.h File Reference

```
#include <decaf/util/NoSuchElementException.h>  #include  
<decaf/lang/exceptions/IllegalStateException.h>  #include  
<decaf/lang/exceptions/UnsupportedOperationException.h>
```

Data Structures

- class **decaf::util::Iterator**< **E** >

Defines an object that can be used to iterate over the elements of a collection.

Namespaces

- namespace **decaf**

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

- namespace **decaf::util**

7.557 src/main/decaf/util/LinkedList.h File Reference

```
#include <list> #include <memory> #include <decaf/util/-
NoSuchElementException.h> #include <decaf/lang/exceptions/-
UnsupportedOperationException.h> #include <decaf/lang/exceptions/-
IndexOutOfBoundsException.h> #include <decaf/lang/System.-
h> #include <decaf/lang/Integer.h> #include <decaf/util/-
Config.h> #include <decaf/util/Deque.h> #include <decaf/util/-
ArrayList.h> #include <decaf/util/Iterator.h> #include
<decaf/util/ListIterator.h> #include <decaf/util/Abstract-
SequentialList.h>
```

Data Structures

- class **decaf::util::LinkedList**< **E** >

*A complete implementation of the **List** (p. 1658) interface using a doubly linked list data structure.*

- class **decaf::util::LinkedList**< **E** >::**ListNode**< **U** >
- class **decaf::util::LinkedList**< **E** >::**LinkedListIterator**
- class **decaf::util::LinkedList**< **E** >::**ConstLinkedListIterator**
- class **decaf::util::LinkedList**< **E** >::**ReverselIterator**
- class **decaf::util::LinkedList**< **E** >::**ConstReverselIterator**

Namespaces

- namespace **decaf**

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

- namespace **decaf::util**

7.558 src/main/decaf/util/List.h File Reference

```
#include <decaf/util/NoSuchElementException.h>    #include  
<decaf/lang/exceptions/IndexOutOfBoundsException.h> ×  
#include <decaf/util/concurrent/Synchronizable.h> #include  
<decaf/util/Config.h>    #include <decaf/util/Iterator.h> ×  
#include <decaf/util/Collection.h> #include <decaf/util/-  
AbstractCollection.h> #include <decaf/util/ListIterator.-  
h>
```

Data Structures

- class **decaf::util::List**< **E** >
An ordered collection (also known as a sequence).

Namespaces

- namespace **decaf**
Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.
- namespace **decaf::util**

7.559 src/main/decaf/util/ListIterator.h File Reference

```
#include <decaf/util/Iterator.h>    #include <decaf/util/-  
Config.h> #include <decaf/lang/exceptions/IllegalArgument-  
Exception.h>
```

Data Structures

- class **decaf::util::ListIterator**< **E** >
An iterator for lists that allows the programmer to traverse the list in either direction, modify the list during iteration, and obtain the iterator's current position in the list.

Namespaces

- namespace **decaf**
Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.
- namespace **decaf::util**

7.560 src/main/decaf/util/logging/ConsoleHandler.h File Reference

```
#include <decaf/util/Config.h> #include <decaf/util/logging/-  
StreamHandler.h> #include <decaf/util/logging/Simple-  
Formatter.h> #include <decaf/io/IOException.h> #include  
<decaf/internal/io/StandardErrorOutputStream.h>
```

Data Structures

- class **decaf::util::logging::ConsoleHandler**
*This **Handler** (p. 1401) publishes log records to System.err.*

Namespaces

- namespace **decaf**
Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.
- namespace **decaf::util**
- namespace **decaf::util::logging**

7.561 src/main/decaf/util/logging/EventManager.h File Reference

```
#include <decaf/util/Config.h> #include <decaf/lang/-  
Exception.h> #include <decaf/util/concurrent/atomic/-  
AtomicBoolean.h> #include <string>
```

Data Structures

- class **decaf::util::logging::EventManager**
***ErrorManager** (p. 1277) objects can be attached to Handlers to process any error that occur on a **Handler** (p. 1401) during Logging.*

Namespaces

- namespace **decaf**
Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.
- namespace **decaf::util**
- namespace **decaf::util::logging**

7.562 src/main/decaf/util/logging/Filter.h File Reference

```
#include <decaf/util/logging/LogRecord.h>
```

Data Structures

- class **decaf::util::logging::Filter**

*A **Filter** (p. 1333) can be used to provide fine grain control over what is logged, beyond the control provided by log levels.*

Namespaces

- namespace **decaf**

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

- namespace **decaf::util**
- namespace **decaf::util::logging**

7.563 src/main/decaf/util/logging/Formatter.h File Reference

```
#include <decaf/util/Config.h> #include <decaf/util/logging/-  
Handler.h>
```

Data Structures

- class **decaf::util::logging::Formatter**

*A **Formatter** (p. 1387) provides support for formatting LogRecords.*

Namespaces

- namespace **decaf**

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

- namespace **decaf::util**
- namespace **decaf::util::logging**

7.564 src/main/decaf/util/logging/Handler.h File Reference

```
#include <decaf/io/Closeable.h>      #include <decaf/lang/-  
Exception.h>      #include <decaf/util/logging/LogRecord.h>  
#include <decaf/util/logging/Level.h> #include <decaf/lang/exceptions/-  
NullPointerException.h> #include <string>
```

Data Structures

- class **decaf::util::logging::Handler**

A **Handler** (p. 1401) object takes log messages from a **Logger** (p. 1693) and exports them.

Namespaces

- namespace **decaf**
Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.
- namespace **decaf::util**
- namespace **decaf::util::logging**

7.565 src/main/decaf/util/logging/Level.h File Reference

```
#include <decaf/util/Config.h>      #include <decaf/lang/-  
Comparable.h>      #include <decaf/lang/exceptions/Illegal-  
ArgumentException.h>
```

Data Structures

- class **decaf::util::logging::Level**

The **Level** (p. 1616) class defines a set of standard logging levels that can be used to control logging output.

Namespaces

- namespace **decaf**
Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.
- namespace **decaf::util**
- namespace **decaf::util::logging**

7.566 src/main/decaf/util/logging/Logger.h File Reference

```
#include <decaf/util/logging/LoggerCommon.h>      #include  
<decaf/util/logging/LogRecord.h> #include <decaf/util/logging/-  
LogManager.h>      #include <decaf/util/logging/Handler.h>  
#include <decaf/util/concurrent/Mutex.h> #include <decaf/util/-  
Config.h> #include <decaf/lang/exceptions/IllegalArgument-  
Exception.h> #include <decaf/lang/exceptions/NullPointerException-  
Exception.h> #include <list> #include <string> #include  
<stdarg.h>
```

Data Structures

- class **decaf::util::logging::Logger**
*A **Logger** (p. 1693) object is used to log messages for a specific system or application component.*

Namespaces

- namespace **decaf**
Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.
- namespace **decaf::util**
- namespace **decaf::util::logging**

7.567 src/main/decaf/util/logging/LoggerCommon.h File Reference

Namespaces

- namespace **decaf**
Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.
- namespace **decaf::util**
- namespace **decaf::util::logging**

Enumerations

- enum **decaf::util::logging::Levels** { **decaf::util::logging::Off**, **decaf::util::logging::Null**, **decaf::util::logging::Markblock**, **decaf::util::logging::Debug**, **decaf::util::logging::Info**, **decaf::util::logging::Warn**, **decaf::util::logging::Error**, **decaf::util::logging::Fatal**, **decaf::util::logging::Throwing** }
Defines an enumeration for logging levels.

7.568 src/main/decaf/util/logging/LoggerDefines.h File Reference

```
#include <decaf/util/logging/SimpleLogger.h>      #include
<sstream>
```

Defines

- #define **LOGDECAF_DECLARE**(loggerName) static **decaf::util::logging::SimpleLogger** loggerName;
- #define **LOGDECAF_INITIALIZE**(loggerName, className, loggerFamily) **decaf::util::logging::SimpleLogger** className::loggerName(loggerFamily);

- **#define LOGDECAF_DECLARE_LOCAL**(loggerName) **decaf::util::logging::-**
Logger loggerName;
- **#define LOGDECAF_DEBUG**(logger, message) logger.debug(__FILE__, __LIN-
E__, message);
- **#define LOGDECAF_DEBUG_1**(logger, message, value)
- **#define LOGDECAF_INFO**(logger, message) logger.info(__FILE__, __LINE__,
message);
- **#define LOGDECAF_ERROR**(logger, message) logger.error(__FILE__, __LINE-
__, message);
- **#define LOGDECAF_WARN**(logger, message) logger.warn(__FILE__, __LINE-
__, message);
- **#define LOGDECAF_FATAL**(logger, message) logger.fatal(__FILE__, __LINE_-
__, message);

7.568.1 Define Documentation

7.568.1.1 **#define LOGDECAF_DEBUG**(*logger, message*) logger.debug(__FILE__,
__LINE__, message);

7.568.1.2 **#define LOGDECAF_DEBUG_1**(*logger, message, value*)

Value:

```

;          \
{
    std::ostringstream ostream;          \
    ostream << message << value;          \
    logger.debug(__FILE__, __LINE__, ostream.str()); \
}

```

7.568.1.3 **#define LOGDECAF_DECLARE**(*loggerName*) static
decaf::util::logging::SimpleLogger loggerName;

7.568.1.4 **#define LOGDECAF_DECLARE_LOCAL**(*loggerName*
) decaf::util::logging::Logger loggerName;

7.568.1.5 **#define LOGDECAF_ERROR**(*logger, message*) logger.error(__FILE__, __LINE__,
message);

7.568.1.6 **#define LOGDECAF_FATAL**(*logger, message*) logger.fatal(__FILE__, __LINE__,
message);

7.568.1.7 **#define LOGDECAF_INFO**(*logger, message*) logger.info(__FILE__, __LINE__,
message);

7.568.1.8 **#define LOGDECAF_INITIALIZE**(*loggerName, className, loggerFamily*
) decaf::util::logging::SimpleLogger className::loggerName(loggerFamily);

```
7.568.1.9 #define LOGDECAF_WARN( logger, message ) logger.warn(__FILE__, __LINE__,
message);
```

7.569 src/main/decaf/util/logging/LoggerHierarchy.h File Reference

Data Structures

- class **decaf::util::logging::LoggerHierarchy**

Namespaces

- namespace **decaf**
Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.
- namespace **decaf::util**
- namespace **decaf::util::logging**

7.570 src/main/decaf/util/logging/LogManager.h File Reference

```
#include <map> #include <list> #include <string> #include
<vector> #include <decaf/lang/Pointer.h> #include <decaf/util/-
Properties.h> #include <decaf/util/concurrent/Mutex.h> ×
#include <decaf/util/Config.h> #include <decaf/io/IO-
Exception.h> #include <decaf/lang/exceptions/NullPointer-
Exception.h> #include <decaf/lang/exceptions/Illegal-
ArgumentException.h>
```

Data Structures

- class **decaf::util::logging::LogManager**
*There is a single global **LogManager** (p. 1712) object that is used to maintain a set of shared state about Loggers and log services.*

Namespaces

- namespace **decaf**
Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.
- namespace **decaf::lang**
- namespace **decaf::io**
- namespace **decaf::util**
- namespace **decaf::util::logging**

7.571 src/main/decaf/util/logging/LogRecord.h File Reference

```
#include <decaf/lang/Throwable.h> #include <decaf/util/logging/-  
LoggerCommon.h> #include <decaf/util/logging/Level.h> ×  
#include <decaf/util/Config.h> #include <memory> #include  
<string>
```

Data Structures

- class **decaf::util::logging::LogRecord**

***LogRecord** (p. 1719) objects are used to pass logging requests between the logging framework and individual log Handlers.*

Namespaces

- namespace **decaf**
Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.
- namespace **decaf::util**
- namespace **decaf::util::logging**

7.572 src/main/decaf/util/logging/LogWriter.h File Reference

```
#include <decaf/util/concurrent/Mutex.h>
```

Data Structures

- class **decaf::util::logging::LogWriter**

Namespaces

- namespace **decaf**
Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.
- namespace **decaf::util**
- namespace **decaf::util::logging**

7.573 src/main/decaf/util/logging/MarkBlockLogger.h File Reference

```
#include <decaf/util/logging/Logger.h>
```

Data Structures

- class **decaf::util::logging::MarkBlockLogger**
Defines a class that can be used to mark the entry and exit from scoped blocks.

Namespaces

- namespace **decaf**
Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.
- namespace **decaf::util**
- namespace **decaf::util::logging**

7.574 src/main/decaf/util/logging/PropertiesChangeListener.h File Reference

```
#include <decaf/util/Config.h>
```

Data Structures

- class **decaf::util::logging::PropertiesChangeListener**
*Defines the interface that classes can use to listen for change events on **Properties** (p. 2200).*

Namespaces

- namespace **decaf**
Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.
- namespace **decaf::util**
- namespace **decaf::util::logging**

7.575 src/main/decaf/util/logging/SimpleFormatter.h File Reference

```
#include <decaf/util/Config.h> #include <decaf/util/logging/-  
Formatter.h> #include <string>
```

Data Structures

- class **decaf::util::logging::SimpleFormatter**
*Print a brief summary of the **LogRecord** (p. 1719) in a human readable format.*

Namespaces

- namespace **decaf**
Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.
- namespace **decaf::util**
- namespace **decaf::util::logging**

7.576 src/main/decaf/util/logging/SimpleLogger.h File Reference

```
#include <string> #include <decaf/util/Config.h>
```

Data Structures

- class **decaf::util::logging::SimpleLogger**

Namespaces

- namespace **decaf**
Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.
- namespace **decaf::util**
- namespace **decaf::util::logging**

7.577 src/main/decaf/util/logging/StreamHandler.h File Reference

```
#include <decaf/util/logging/LoggerCommon.h> #include  
<decaf/util/logging/Handler.h> #include <decaf/lang/exceptions/-  
NullPointerException.h> #include <decaf/lang/exceptions/-  
InvalidStateException.h> #include <decaf/util/concurrent/-  
Concurrent.h> #include <decaf/util/Config.h>
```

Data Structures

- class **decaf::util::logging::StreamHandler**
*Stream based logging **Handler** (p. 1401).*

Namespaces

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

- namespace **decaf::io**
- namespace **decaf::util**
- namespace **decaf::util::logging**

7.578 src/main/decaf/util/logging/XMLFormatter.h File Reference

```
#include <decaf/util/Config.h> #include <decaf/util/logging/-
Formatter.h> #include <decaf/util/logging/LogRecord.h>
```

Data Structures

- class **decaf::util::logging::XMLFormatter**
*Format a **LogRecord** (p. 1719) into a standard XML format.*

Namespaces

- namespace **decaf**
Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.
- namespace **decaf::util**
- namespace **decaf::util::logging**

7.579 src/main/decaf/util/Map.h File Reference

```
#include <functional> #include <vector> #include <decaf/lang/exceptions/-
UnsupportedOperationException.h> #include <decaf/util/NoSuchElementException.h> #include <decaf/util/concurrent/-
Synchronizable.h> #include <decaf/util/concurrent/Mutex.-
h>
```

Data Structures

- class **decaf::util::Map< K, V, COMPARATOR >**
***Map** (p. 1768) template that wraps around a `std::map` to provide a more user-friendly interface and to provide common functions that do not exist in `std::map`.*
- class **decaf::util::Map< K, V, COMPARATOR >::Entry**

Namespaces

- namespace **decaf**
Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.
- namespace **decaf::util**

7.580 src/main/decaf/util/NoSuchElementException.h File Reference

```
#include <decaf/lang/exceptions/RuntimeException.h>
```

Data Structures

- class **decaf::util::NoSuchElementException**

Namespaces

- namespace **decaf**
Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.
- namespace **decaf::util**

7.581 src/main/decaf/util/PriorityQueue.h File Reference

```
#include <decaf/util/Config.h>      #include <decaf/util/-  
Collection.h>      #include <decaf/util/AbstractQueue.h> ×  
#include <decaf/util/Iterator.h>    #include <decaf/util/-  
Comparator.h>      #include <decaf/util/comparators/Less.-  
h> #include <decaf/lang/Math.h>    #include <decaf/lang/-  
Pointer.h>    #include <decaf/lang/exceptions/NullPointer-  
Exception.h> #include <decaf/lang/exceptions/Unsupported-  
OperationException.h> #include <memory>
```

Data Structures

- class **decaf::util::PriorityQueue< E >**
An unbounded priority queue based on a binary heap algorithm.
- class **decaf::util::PriorityQueue< E >::PriorityQueueIterator**
- class **decaf::util::PriorityQueue< E >::ConstPriorityQueueIterator**

Namespaces

- namespace **decaf**
Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.
- namespace **decaf::util**

7.582 src/main/decaf/util/Properties.h File Reference

```
#include <vector> #include <string> #include <decaf/util/-  
Config.h> #include <decaf/util/StlMap.h> #include <decaf/io/-  
InputStream.h> #include <decaf/io/OutputStream.h> #include  
<decaf/lang/Pointer.h> #include <decaf/lang/exceptions/-  
IllegalArgumentException.h> #include <decaf/lang/exceptions/-  
NullPointerException.h> #include <decaf/io/IOException.-  
h>
```

Data Structures

- class **decaf::util::Properties**

Java-like properties class for mapping string names to string values.

Namespaces

- namespace **decaf**

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

- namespace **decaf::io**
- namespace **decaf::util**

7.583 src/main/decaf/util/Random.h File Reference

```
#include <decaf/lang/exceptions/IllegalArgumentException.-  
h> #include <decaf/util/Config.h> #include <vector> ×  
#include <cmath>
```

Data Structures

- class **decaf::util::Random**

Random (p. 2229) *Value Generator which is used to generate a stream of pseudorandom numbers.*

Namespaces

- namespace **decaf**

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

- namespace **decaf::util**

7.584 src/main/decaf/util/Set.h File Reference

```
#include <decaf/util/NoSuchElementException.h>    #include  
<decaf/util/concurrent/Synchronizable.h> #include <decaf/util/concurrent/-  
Mutex.h> #include <decaf/util/Iterator.h> #include <decaf/util/-  
AbstractCollection.h>
```

Data Structures

- class **decaf::util::Set**< **E** >
A collection that contains no duplicate elements.

Namespaces

- namespace **decaf**
Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.
- namespace **decaf::util**

7.585 src/main/decaf/util/StlList.h File Reference

```
#include <list> #include <algorithm> #include <memory>  
#include <decaf/lang/exceptions/UnsupportedOperation-  
Exception.h> #include <decaf/util/NoSuchElementException.-  
h> #include <decaf/lang/exceptions/IndexOutOfBoundsException.-  
h> #include <decaf/util/Config.h> #include <decaf/util/-  
Iterator.h> #include <decaf/util/ListIterator.h> #include  
<decaf/util/List.h> #include <decaf/util/AbstractList.-  
h>
```

Data Structures

- class **decaf::util::StlList**< **E** >
***List** (p. 1658) class that wraps the STL list object to provide a simpler interface and additional methods not provided by the STL type.*
- class **decaf::util::StlList**< **E** >::**StlListIterator**
- class **decaf::util::StlList**< **E** >::**ConstStlListIterator**

Namespaces

- namespace **decaf**
Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.
- namespace **decaf::util**

7.586 src/main/decaf/util/StlMap.h File Reference

```
#include <map> #include <vector> #include <decaf/util/NoSuchElementException.h> #include <decaf/util/concurrent/NotSynchronizable.h> #include <decaf/util/concurrent/Mutex.h> #include <decaf/util/Map.h>
```

Data Structures

- class **decaf::util::StlMap**< K, V, COMPARATOR >

***Map** (p. 1768) template that wraps around a `std::map` to provide a more user-friendly interface and to provide common functions that do not exist in `std::map`.*

Namespaces

- namespace **decaf**

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

- namespace **decaf::util**

7.587 src/main/decaf/util/StlQueue.h File Reference

```
#include <list> #include <vector> #include <decaf/util/Iterator.h> #include <decaf/util/concurrent/Mutex.h> #include <decaf/lang/Exception.h>
```

Data Structures

- class **decaf::util::StlQueue**< T >

*The **Queue** (p. 2222) class accepts messages with an `psuh(m)` command where `m` is the message to be queued.*

- class **decaf::util::StlQueue**< T >::QueueIterator

Namespaces

- namespace **decaf**

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

- namespace **decaf::util**

7.588 src/main/decaf/util/StlSet.h File Reference

```
#include <set>      #include <vector>      #include <memory> ×  
#include <decaf/util/NoSuchElementException.h>      #include  
<decaf/util/concurrent/Synchronizable.h> #include <decaf/util/concurrent/-  
Mutex.h> #include <decaf/util/Iterator.h> #include <decaf/util/-  
AbstractSet.h>
```

Data Structures

- class **decaf::util::StlSet< E >**
Set (p. 2397) template that wraps around a *std::set* to provide a more user-friendly interface and to provide common functions that do not exist in *std::set*.
- class **decaf::util::StlSet< E >::SetIterator**
- class **decaf::util::StlSet< E >::ConstSetIterator**

Namespaces

- namespace **decaf**
Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.
- namespace **decaf::util**

7.589 src/main/decaf/util/StringTokenizer.h File Reference

```
#include <decaf/util/NoSuchElementException.h>      #include  
<decaf/util/Config.h> #include <string>
```

Data Structures

- class **decaf::util::StringTokenizer**
Class that allows for parsing of string based on Tokens.

Namespaces

- namespace **decaf**
Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.
- namespace **decaf::util**

7.590 src/main/decaf/util/Timer.h File Reference

```
#include <memory> #include <decaf/util/Config.h> #include  
<decaf/util/Date.h> #include <decaf/lang/Pointer.h> ×  
#include <decaf/lang/exceptions/NullPointerException.h>  
#include <decaf/lang/exceptions/IllegalStateException.-  
h> #include <decaf/lang/exceptions/IllegalArgumentException.-  
h>
```

Data Structures

- class **decaf::util::Timer**

A facility for threads to schedule tasks for future execution in a background thread.

Namespaces

- namespace **decaf**

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

- namespace **decaf::util**

7.591 src/main/decaf/util/TimerTask.h File Reference

```
#include <decaf/util/Config.h> #include <decaf/lang/-  
Runnable.h> #include <decaf/util/concurrent/Mutex.h>
```

Data Structures

- class **decaf::util::TimerTask**

*A Base class for a task object that can be scheduled for one-time or repeated execution by a **Timer** (p. 2739).*

Namespaces

- namespace **decaf**

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

- namespace **decaf::internal**
- namespace **decaf::internal::util**
- namespace **decaf::util**

7.592 src/main/decaf/util/UUID.h File Reference

```
#include <decaf/util/Config.h>      #include <decaf/lang/-  
Comparable.h> #include <decaf/lang/exceptions/Unsupported-  
OperationException.h>      #include <decaf/lang/exceptions/-  
IllegalArgumentException.h> #include <apr_uuid.h> #include  
<string>
```

Data Structures

- class **decaf::util::UUID**

*A class that represents an immutable universally unique identifier (**UUID** (p. 2877)).*

Namespaces

- namespace **decaf**

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

- namespace **decaf::util**

7.593 src/main/decaf/util/zip/Adler32.h File Reference

```
#include <decaf/util/Config.h> #include <decaf/util/zip/-  
Checksum.h>
```

Data Structures

- class **decaf::util::zip::Adler32**

*Clas that can be used to compute an Adler-32 **Checksum** (p. 810) for a data stream.*

Namespaces

- namespace **decaf**

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

- namespace **decaf::util**
- namespace **decaf::util::zip**

7.594 src/main/decaf/util/zip/CheckedInputStream.h File Reference

```
#include <decaf/util/Config.h> #include <decaf/util/zip/-  
Checksum.h> #include <decaf/io/FilterInputStream.h>
```

Data Structures

- class **decaf::util::zip::CheckedInputStream**

*An implementation of a `FilterInputStream` that will maintain a **Checksum** (p. 810) of the bytes read, the **Checksum** (p. 810) can then be used to verify the integrity of the input stream.*

Namespaces

- namespace **decaf**

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

- namespace **decaf::util**
- namespace **decaf::util::zip**

7.595 src/main/decaf/util/zip/CheckedOutputStream.h File Reference

```
#include <decaf/util/Config.h> #include <decaf/util/zip/-  
Checksum.h> #include <decaf/io/FilterOutputStream.h>
```

Data Structures

- class **decaf::util::zip::CheckedOutputStream**

*An implementation of a `FilterOutputStream` that will maintain a **Checksum** (p. 810) of the bytes written, the **Checksum** (p. 810) can then be used to verify the integrity of the output stream.*

Namespaces

- namespace **decaf**

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

- namespace **decaf::util**
- namespace **decaf::util::zip**

7.596 src/main/decaf/util/zip/Checksum.h File Reference

```
#include <decaf/util/Config.h> #include <decaf/lang/exceptions/-  
NullPointerException.h> #include <decaf/lang/exceptions/-  
IndexOutOfBoundsException.h> #include <vector>
```

Data Structures

- class **decaf::util::zip::Checksum**

*An interface used to represent **Checksum** (p. 810) values in the Zip package.*

Namespaces

- namespace **decaf**

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

- namespace **decaf::util**
- namespace **decaf::util::zip**

7.597 src/main/decaf/util/zip/CRC32.h File Reference

```
#include <decaf/util/Config.h> #include <decaf/util/zip/-  
Checksum.h>
```

Data Structures

- class **decaf::util::zip::CRC32**

Class that can be used to compute a CRC-32 checksum for a data stream.

Namespaces

- namespace **decaf**

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

- namespace **decaf::util**
- namespace **decaf::util::zip**

7.598 src/main/decaf/util/zip/DataFormatException.h File Reference

```
#include <decaf/lang/Exception.h>
```

Data Structures

- class **decaf::util::zip::DataFormatException**

Namespaces

- namespace **decaf**

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

- namespace **decaf::util**
- namespace **decaf::util::zip**

7.599 src/main/decaf/util/zip/Deflater.h File Reference

```
#include <decaf/util/Config.h> #include <decaf/lang/exceptions/-  
NullPointerException.h> #include <decaf/lang/exceptions/-  
IllegalArgumentException.h> #include <decaf/lang/exceptions/-  
IllegalStateException.h> #include <decaf/lang/exceptions/-  
IndexOutOfBoundsException.h> #include <vector>
```

Data Structures

- class **decaf::util::zip::Deflater**

This class compresses data using the DEFLATE algorithm (see specification).

Namespaces

- namespace **decaf**

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

- namespace **decaf::util**
- namespace **decaf::util::zip**

7.600 src/main/decaf/util/zip/DeflaterOutputStream.h File Reference

```
#include <decaf/util/Config.h> #include <decaf/io/Filter-  
OutputStream.h> #include <decaf/io/IOException.h> #include  
<decaf/util/zip/Deflater.h> #include <vector>
```

Data Structures

- class **decaf::util::zip::DeflaterOutputStream**

Provides a FilterOutputStream instance that compresses the data before writing it to the wrapped OutputStream.

Namespaces

- namespace **decaf**
Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.
- namespace **decaf::util**
- namespace **decaf::util::zip**

7.601 src/main/decaf/util/zip/Inflater.h File Reference

```
#include <decaf/util/Config.h> #include <decaf/lang/exceptions/-  
NullPointerException.h> #include <decaf/lang/exceptions/-  
IllegalArgumentException.h> #include <decaf/lang/exceptions/-  
IllegalStateException.h> #include <decaf/lang/exceptions/-  
IndexOutOfBoundsException.h> #include <decaf/util/zip/-  
DataFormatException.h> #include <vector>
```

Data Structures

- class **decaf::util::zip::Inflater**
This class uncompresses data that was compressed using the DEFLATE algorithm (see specification).

Namespaces

- namespace **decaf**
Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.
- namespace **decaf::util**
- namespace **decaf::util::zip**

7.602 src/main/decaf/util/zip/InflaterInputStream.h File Reference

```
#include <decaf/util/Config.h> #include <decaf/io/Filter-  
InputStream.h> #include <decaf/io/IOException.h> #include  
<decaf/util/zip/Inflater.h> #include <vector>
```

Data Structures

- class **decaf::util::zip::InflaterInputStream**
A FilterInputStream that decompresses data read from the wrapped InputStream instance.

Namespaces

- namespace **decaf**
Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.
- namespace **decaf::util**
- namespace **decaf::util::zip**

7.603 src/main/decaf/util/zip/ZipException.h File Reference

```
#include <decaf/io/IOException.h>
```

Data Structures

- class **decaf::util::zip::ZipException**

Namespaces

- namespace **decaf**
Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.
- namespace **decaf::util**
- namespace **decaf::util::zip**

Index

- ~AbortPolicy
 - decaf::util::concurrent::ThreadPool-
Executor::AbortPolicy, 106
- ~AbstractCollection
 - decaf::util::AbstractCollection, 110
- ~AbstractExecutorService
 - decaf::util::concurrent::Abstract-
ExecutorService, 122
- ~AbstractList
 - decaf::util::AbstractList, 124
- ~AbstractMap
 - decaf::util::AbstractMap, 135
- ~AbstractOwnableSynchronizer
 - decaf::util::concurrent::locks::-
AbstractOwnableSynchronizer,
136
- ~AbstractQueue
 - decaf::util::AbstractQueue, 139
- ~AbstractSequentialList
 - decaf::util::AbstractSequentialList,
146
- ~AbstractSet
 - decaf::util::AbstractSet, 153
- ~AbstractTransportFactory
 - activemq::transport::AbstractTransport-
Factory, 155
- ~ActiveMQAckHandler
 - activemq::core::ActiveMQAck-
Handler, 156
- ~ActiveMQBlobMessage
 - activemq::commands::ActiveMQ-
BlobMessage, 158
- ~ActiveMQBlobMessageMarshaller
 - activemq::wireformat::openwire-
::marshal::generated::Active-
MQBlobMessageMarshaller,
162
- ~ActiveMQBytesMessage
 - activemq::commands::ActiveMQ-
BytesMessage, 168
- ~ActiveMQBytesMessageMarshaller
 - activemq::wireformat::openwire-
::marshal::generated::Active-
MQBytesMessageMarshaller,
184
- ~ActiveMQCPP
 - activemq::library::ActiveMQCPP, 245
- ~ActiveMQConnection
 - activemq::core::ActiveMQConnection,
193
- ~ActiveMQConnectionFactory
 - activemq::core::ActiveMQConnection-
Factory, 216
- ~ActiveMQConnectionMetaData
 - activemq::core::ActiveMQConnection-
MetaData, 228
- ~ActiveMQConsumer
 - activemq::core::ActiveMQConsumer,
236
- ~ActiveMQDestination
 - activemq::commands::ActiveMQ-
Destination, 249
- ~ActiveMQDestinationMarshaller
 - activemq::wireformat::openwire-
::marshal::generated::Active-
MQDestinationMarshaller, 259
- ~ActiveMQException
 - activemq::exceptions::ActiveMQ-
Exception, 263
- ~ActiveMQMapMessage
 - activemq::commands::ActiveMQ-
MapMessage, 270
- ~ActiveMQMapMessageMarshaller
 - activemq::wireformat::openwire-
::marshal::generated::Active-
MQMapMessageMarshaller,
285
- ~ActiveMQMessage
 - activemq::commands::ActiveMQ-
Message, 289
- ~ActiveMQMessageMarshaller
 - activemq::wireformat::openwire-

- ::marshal::generated::ActiveMQMessageMarshaller, 292
- ~ActiveMQMessageTemplate
 - activemq::commands::ActiveMQMessageTemplate, 296
- ~ActiveMQObjectMessage
 - activemq::commands::ActiveMQObjectMessage, 302
- ~ActiveMQObjectMessageMarshaller
 - activemq::wireformat::openwire::marshal::generated::ActiveMQObjectMessageMarshaller, 305
- ~ActiveMQProducer
 - activemq::core::ActiveMQProducer, 309
- ~ActiveMQProperties
 - activemq::util::ActiveMQProperties, 317
- ~ActiveMQQueue
 - activemq::commands::ActiveMQQueue, 322
- ~ActiveMQQueueBrowser
 - activemq::core::ActiveMQQueueBrowser, 326
- ~ActiveMQQueueMarshaller
 - activemq::wireformat::openwire::marshal::generated::ActiveMQQueueMarshaller, 330
- ~ActiveMQSession
 - activemq::core::ActiveMQSession, 337
- ~ActiveMQSessionExecutor
 - activemq::core::ActiveMQSessionExecutor, 356
- ~ActiveMQStreamMessage
 - activemq::commands::ActiveMQStreamMessage, 361
- ~ActiveMQStreamMessageMarshaller
 - activemq::wireformat::openwire::marshal::generated::ActiveMQStreamMessageMarshaller, 376
- ~ActiveMQTempDestination
 - activemq::commands::ActiveMQTempDestination, 380
- ~ActiveMQTempDestinationMarshaller
 - activemq::wireformat::openwire::marshal::generated::ActiveMQTempDestinationMarshaller, 383
- ~ActiveMQTempQueue
 - activemq::commands::ActiveMQTempQueue, 387
- ~ActiveMQTempQueueMarshaller
 - activemq::wireformat::openwire::marshal::generated::ActiveMQTempQueueMarshaller, 392
- ~ActiveMQTempTopic
 - activemq::commands::ActiveMQTempTopic, 397
- ~ActiveMQTempTopicMarshaller
 - activemq::wireformat::openwire::marshal::generated::ActiveMQTempTopicMarshaller, 402
- ~ActiveMQTextMessage
 - activemq::commands::ActiveMQTextMessage, 406
- ~ActiveMQTextMessageMarshaller
 - activemq::wireformat::openwire::marshal::generated::ActiveMQTextMessageMarshaller, 411
- ~ActiveMQTopic
 - activemq::commands::ActiveMQTopic, 415
- ~ActiveMQTopicMarshaller
 - activemq::wireformat::openwire::marshal::generated::ActiveMQTopicMarshaller, 420
- ~ActiveMQTransactionContext
 - activemq::core::ActiveMQTransactionContext, 425
- ~ActiveMQXACConnection
 - activemq::core::ActiveMQXACConnection, 432
- ~ActiveMQXACConnectionFactory
 - activemq::core::ActiveMQXACConnectionFactory, 435
- ~ActiveMQXASession
 - activemq::core::ActiveMQXASession, 437
- ~Adler32
 - decaf::util::zip::Adler32, 440
- ~Appendable
 - decaf::lang::Appendable, 442
- ~AprPool
 - decaf::internal::AprPool, 445
- ~ArrayList
 - decaf::util::ArrayList, 448
- ~ArrayListIterator

- decaf::util::concurrent::CopyOn-WriteArrayList::ArrayListIterator, 459
- ~ArrayPointer
 - decaf::lang::ArrayPointer, 466
- ~ArrayPointerComparator
 - decaf::lang::ArrayPointerComparator, 471
- ~Arrays
 - decaf::util::Arrays, 472
- ~AtomicBoolean
 - decaf::util::concurrent::atomic::AtomicBoolean, 474
- ~AtomicInteger
 - decaf::util::concurrent::atomic::AtomicInteger, 477
- ~AtomicRefCounter
 - decaf::util::concurrent::atomic::AtomicRefCounter, 482
- ~AtomicReference
 - decaf::util::concurrent::atomic::AtomicReference, 485
- ~BackupTransport
 - activemq::transport::failover::BackupTransport, 487
- ~BackupTransportPool
 - activemq::transport::failover::BackupTransportPool, 490
- ~BaseCommand
 - activemq::commands::BaseCommand, 493
- ~BaseCommandMarshaller
 - activemq::wireformat::openwire::marshal::generated::BaseCommandMarshaller, 500
- ~BaseDataStreamMarshaller
 - activemq::wireformat::openwire::marshal::BaseDataStreamMarshaller, 511
- ~BaseDataStructure
 - activemq::commands::BaseDataStructure, 529
- ~BindException
 - decaf::net::BindException, 534
- ~BlockingByteArrayInputStream
 - decaf::io::BlockingByteArrayInputStream, 536
- ~BlockingQueue
 - decaf::util::concurrent::BlockingQueue, 541
- ~Boolean
 - decaf::lang::Boolean, 546
- ~BooleanExpression
 - activemq::commands::BooleanExpression, 551
- ~BooleanStream
 - activemq::wireformat::openwire::utils::BooleanStream, 553
- ~BrokenBarrierException
 - decaf::util::concurrent::BrokenBarrierException, 558
- ~BrokerError
 - activemq::commands::BrokerError, 560
- ~BrokerException
 - activemq::exceptions::BrokerException, 564
- ~BrokerId
 - activemq::commands::BrokerId, 565
- ~BrokerIdMarshaller
 - activemq::wireformat::openwire::marshal::generated::BrokerIdMarshaller, 568
- ~BrokerInfo
 - activemq::commands::BrokerInfo, 573
- ~BrokerInfoMarshaller
 - activemq::wireformat::openwire::marshal::generated::BrokerInfoMarshaller, 579
- ~Buffer
 - decaf::nio::Buffer, 585
- ~BufferFactory
 - decaf::internal::nio::BufferFactory, 599
- ~BufferOverflowException
 - decaf::nio::BufferOverflowException, 611
- ~BufferUnderflowException
 - decaf::nio::BufferUnderflowException, 613
- ~BufferedInputStream
 - decaf::io::BufferedInputStream, 592
- ~BufferedOutputStream
 - decaf::io::BufferedOutputStream, 596
- ~Byte
 - decaf::lang::Byte, 616
- ~ByteArrayAdapter

- decaf::internal::util::ByteArray-Adapter, 630
- ~ByteArrayBuffer
 - decaf::internal::nio::ByteArrayBuffer, 660
- ~ByteArrayInputStream
 - decaf::io::ByteArrayInputStream, 683
- ~ByteArrayOutputStream
 - decaf::io::ByteArrayOutputStream, 688
- ~ByteBuffer
 - decaf::nio::ByteBuffer, 695
- ~BytesMessage
 - cms::BytesMessage, 720
- ~CMSException
 - cms::CMSException, 827
- ~CMSExceptionSupport
 - activemq::util::CMSException-Support, 829
- ~CMSProperties
 - cms::CMSProperties, 831
- ~CMSSecurityException
 - cms::CMSSecurityException, 836
- ~CRC32
 - decaf::util::zip::CRC32, 1066
- ~CachedConsumer
 - activemq::cmsutil::CachedConsumer, 735
- ~CachedProducer
 - activemq::cmsutil::CachedProducer, 740
- ~Callable
 - decaf::util::concurrent::Callable, 746
- ~CallerRunsPolicy
 - decaf::util::concurrent::ThreadPool-Executor::CallerRunsPolicy, 748
- ~CancellationException
 - decaf::util::concurrent::Cancellation-Exception, 750
- ~Certificate
 - decaf::security::cert::Certificate, 752
- ~CertificateEncodingException
 - decaf::security::cert::Certificate-EncodingException, 756
- ~CertificateException
 - decaf::security::cert::Certificate-Exception, 758
- ~CertificateExpiredException
 - decaf::security::cert::Certificate-ExpiredException, 760
- ~CertificateNotYetValidException
 - decaf::security::cert::CertificateNot-YetValidException, 762
- ~CertificateParsingException
 - decaf::security::cert::Certificate-ParsingException, 764
- ~CharArrayBuffer
 - decaf::internal::nio::CharArrayBuffer, 779
- ~CharBuffer
 - decaf::nio::CharBuffer, 788
- ~CharSequence
 - decaf::lang::CharSequence, 804
- ~CheckedInputStream
 - decaf::util::zip::CheckedInputStream, 807
- ~CheckedOutputStream
 - decaf::util::zip::CheckedOutput-Stream, 809
- ~Checksum
 - decaf::util::zip::Checksum, 811
- ~ClassCastException
 - decaf::lang::exceptions::ClassCast-Exception, 815
- ~CloseTransportsTask
 - activemq::transport::failover::Close-TransportsTask, 818
- ~Closeable
 - cms::Closeable, 816
 - decaf::io::Closeable, 817
- ~CmsAccessor
 - activemq::cmsutil::CmsAccessor, 820
- ~CmsDestinationAccessor
 - activemq::cmsutil::CmsDestination-Accessor, 824
- ~CmsTemplate
 - activemq::cmsutil::CmsTemplate, 839
- ~Collection
 - decaf::util::Collection, 853
- ~Command
 - activemq::commands::Command, 867
- ~CommandVisitor
 - activemq::state::CommandVisitor, 874
- ~CommandVisitorAdapter

- activemq::state::CommandVisitor-Adapter, 881
- ~Comparable
 - decaf::lang::Comparable, 886
- ~Comparator
 - decaf::util::Comparator, 889
- ~CompositeData
 - activemq::util::CompositeData, 891
- ~CompositeTask
 - activemq::threads::CompositeTask, 893
- ~CompositeTaskRunner
 - activemq::threads::CompositeTaskRunner, 894
- ~CompositeTransport
 - activemq::transport::CompositeTransport, 897
- ~ConcurrentMap
 - decaf::util::concurrent::ConcurrentMap, 898
- ~ConcurrentModificationException
 - decaf::util::ConcurrentModificationException, 904
- ~ConcurrentStlMap
 - decaf::util::concurrent::ConcurrentStlMap, 909
- ~Condition
 - decaf::util::concurrent::locks::Condition, 923
- ~ConditionHandle
 - decaf::util::concurrent::ConditionHandle, 928
- ~ConnectException
 - decaf::net::ConnectException, 933
- ~Connection
 - cms::Connection, 935
- ~ConnectionControl
 - activemq::commands::ConnectionControl, 940
- ~ConnectionControlMarshaller
 - activemq::wireformat::openwire::marshal::generated::ConnectionControlMarshaller, 944
- ~ConnectionError
 - activemq::commands::ConnectionError, 949
- ~ConnectionErrorMarshaller
 - activemq::wireformat::openwire::marshal::generated::ConnectionErrorMarshaller, 952
- ~ConnectionFactory
 - cms::ConnectionFactory, 956
- ~ConnectionFailedException
 - activemq::exceptions::ConnectionFailedException, 959
- ~ConnectionId
 - activemq::commands::ConnectionId, 961
- ~ConnectionIdMarshaller
 - activemq::wireformat::openwire::marshal::generated::ConnectionIdMarshaller, 964
- ~ConnectionInfo
 - activemq::commands::ConnectionInfo, 969
- ~ConnectionInfoMarshaller
 - activemq::wireformat::openwire::marshal::generated::ConnectionInfoMarshaller, 975
- ~ConnectionMetaData
 - cms::ConnectionMetaData, 979
- ~ConnectionState
 - activemq::state::ConnectionState, 983
- ~ConnectionStateTracker
 - activemq::state::ConnectionStateTracker, 986
- ~ConsoleHandler
 - decaf::util::logging::ConsoleHandler, 991
- ~ConsumerControl
 - activemq::commands::ConsumerControl, 993
- ~ConsumerControlMarshaller
 - activemq::wireformat::openwire::marshal::generated::ConsumerControlMarshaller, 997
- ~ConsumerId
 - activemq::commands::ConsumerId, 1002
- ~ConsumerIdMarshaller
 - activemq::wireformat::openwire::marshal::generated::ConsumerIdMarshaller, 1006
- ~ConsumerInfo
 - activemq::commands::ConsumerInfo, 1011
- ~ConsumerInfoMarshaller
 - activemq::wireformat::openwire::marshal::generated::Consumer-

- InfoMarshaller, 1018
- ~ConsumerState
 - activemq::state::ConsumerState, 1022
- ~ControlCommand
 - activemq::commands::ControlCommand, 1023
- ~ControlCommandMarshaller
 - activemq::wireformat::openwire::marshal::generated::ControlCommandMarshaller, 1026
- ~CopyOnWriteArrayList
 - decaf::util::concurrent::CopyOnWriteArrayList, 1032
- ~CopyOnWriteArraySet
 - decaf::util::concurrent::CopyOnWriteArraySet, 1054
- ~CountDownLatch
 - decaf::util::concurrent::CountDownLatch, 1063
- ~DataArrayResponse
 - activemq::commands::DataArrayResponse, 1070
- ~DataArrayResponseMarshaller
 - activemq::wireformat::openwire::marshal::generated::DataArrayResponseMarshaller, 1073
- ~DataFormatException
 - decaf::util::zip::DataFormatException, 1078
- ~DataInput
 - decaf::io::DataInput, 1087
- ~DataInputStream
 - decaf::io::DataInputStream, 1096
- ~DataOutput
 - decaf::io::DataOutput, 1104
- ~DataOutputStream
 - decaf::io::DataOutputStream, 1110
- ~DataResponse
 - activemq::commands::DataResponse, 1113
- ~DataResponseMarshaller
 - activemq::wireformat::openwire::marshal::generated::DataResponseMarshaller, 1116
- ~DataStreamMarshaller
 - activemq::wireformat::openwire::marshal::DataStreamMarshaller, 1120
- ~DataStructure
 - activemq::commands::DataStructure, 1133
- ~DatagramPacket
 - decaf::net::DatagramPacket, 1082
- ~Date
 - decaf::util::Date, 1140
- ~DecafRuntime
 - decaf::internal::DecafRuntime, 1143
- ~DedicatedTaskRunner
 - activemq::threads::DedicatedTaskRunner, 1145
- ~DefaultPrefetchPolicy
 - activemq::core::policies::DefaultPrefetchPolicy, 1147
- ~DefaultRedeliveryPolicy
 - activemq::core::policies::DefaultRedeliveryPolicy, 1151
- ~DefaultSSLContext
 - decaf::internal::net::ssl::DefaultSSLContext, 1164
- ~DefaultSSLServerSocketFactory
 - decaf::internal::net::ssl::DefaultSSLServerSocketFactory, 1167
- ~DefaultSSLSocketFactory
 - decaf::internal::net::ssl::DefaultSSLSocketFactory, 1173
- ~DefaultServerSocketFactory
 - decaf::internal::net::DefaultServerSocketFactory, 1157
- ~DefaultSocketFactory
 - decaf::internal::DefaultSocketFactory, 1161
- ~DefaultTransportListener
 - activemq::transport::DefaultTransportListener, 1178
- ~Deflater
 - decaf::util::zip::Deflater, 1182
- ~DeflaterOutputStream
 - decaf::util::zip::DeflaterOutputStream, 1192
- ~Delayed
 - decaf::util::concurrent::Delayed, 1194
- ~DeliveryMode
 - cms::DeliveryMode, 1196
- ~Deque
 - decaf::util::Deque, 1198
- ~Destination
 - cms::Destination, 1211

- ~DestinationInfo
 - activemq::commands::Destination-Info, 1215
- ~DestinationInfoMarshaller
 - activemq::wireformat::openwire-::marshal::generated::Destination-InfoMarshaller, 1219
- ~DestinationResolver
 - activemq::cmsutil::Destination-Resolver, 1222
- ~DiscardOldestPolicy
 - decaf::util::concurrent::ThreadPool-Executor::CallerRunsPolicy::-DiscardOldestPolicy, 1224
- ~DiscardPolicy
 - decaf::util::concurrent::ThreadPool-Executor::CallerRunsPolicy::-DiscardPolicy, 1226
- ~DiscoveryEvent
 - activemq::commands::Discovery-Event, 1227
- ~DiscoveryEventMarshaller
 - activemq::wireformat::openwire-::marshal::generated::Discovery-EventMarshaller, 1231
- ~Dispatcher
 - activemq::core::Dispatcher, 1235
- ~Double
 - decaf::lang::Double, 1238
- ~DoubleArrayBuffer
 - decaf::internal::nio::DoubleArray-Buffer, 1253
- ~DoubleBuffer
 - decaf::nio::DoubleBuffer, 1261
- ~DynamicDestinationResolver
 - activemq::cmsutil::DynamicDestination-Resolver, 1272
- ~EOFException
 - decaf::io::EOFException, 1277
- ~Entry
 - decaf::util::Map::Entry, 1274
- ~ErrorManager
 - decaf::util::logging::ErrorManager, 1278
- ~Exception
 - decaf::lang::Exception, 1282
- ~ExceptionListener
 - cms::ExceptionListener, 1287
- ~ExceptionResponse
 - activemq::commands::Exception-Response, 1288
- ~ExceptionResponseMarshaller
 - activemq::wireformat::openwire-::marshal::generated::Exception-ResponseMarshaller, 1291
- ~ExecutionException
 - decaf::util::concurrent::Execution-Exception, 1296
- ~Executor
 - decaf::util::concurrent::Executor, 1299
- ~ExecutorService
 - decaf::util::concurrent::Executor-Service, 1303
- ~Executors
 - decaf::util::concurrent::Executors, 1300
- ~FailoverTransport
 - activemq::transport::failover::Failover-Transport, 1308
- ~FailoverTransportFactory
 - activemq::transport::failover::Failover-TransportFactory, 1319
- ~FailoverTransportListener
 - activemq::transport::failover::Failover-TransportListener, 1321
- ~FifoMessageDispatchChannel
 - activemq::core::FifoMessageDispatch-Channel, 1324
- ~FileDescriptor
 - decaf::io::FileDescriptor, 1331
- ~Filter
 - decaf::util::logging::Filter, 1333
- ~FilterInputStream
 - decaf::io::FilterInputStream, 1336
- ~FilterOutputStream
 - decaf::io::FilterOutputStream, 1342
- ~Float
 - decaf::lang::Float, 1347
- ~FloatArrayBuffer
 - decaf::internal::nio::FloatArrayBuffer, 1362
- ~FloatBuffer
 - decaf::nio::FloatBuffer, 1370
- ~FlushCommand
 - activemq::commands::FlushCommand, 1381
- ~FlushCommandMarshaller

- activemq::wireformat::openwire-
::marshal::generated::Flush-
CommandMarshaller, 1384
- ~Flushable
 - decaf::io::Flushable, 1380
- ~Formatter
 - decaf::util::logging::Formatter, 1388
- ~Future
 - decaf::util::concurrent::Future, 1390
- ~FutureResponse
 - activemq::transport::correlator::-
FutureResponse, 1393
- ~GeneralSecurityException
 - decaf::security::GeneralSecurity-
Exception, 1397
- ~GenericResource
 - decaf::internal::util::GenericResource,
1398
- ~Handler
 - decaf::util::logging::Handler, 1403
- ~HexStringParser
 - decaf::internal::util::HexStringParser,
1406
- ~HexTable
 - activemq::wireformat::openwire-
::utils::HexTable, 1408
- ~HttpRequestException
 - decaf::net::HttpRequestException, 1410
- ~IOException
 - decaf::io::IOException, 1547
- ~IOTransport
 - activemq::transport::IOTransport,
1550
- ~IdGenerator
 - activemq::util::IdGenerator, 1412
- ~IllegalArgumentException
 - decaf::lang::exceptions::Illegal-
ArgumentException, 1415
- ~IllegalMonitorStateException
 - decaf::lang::exceptions::Illegal-
MonitorStateException, 1418
- ~IllegalStateException
 - cms::IllegalStateException, 1420
 - decaf::lang::exceptions::IllegalState-
Exception, 1422
- ~IllegalThreadStateException
 - decaf::lang::exceptions::Illegal-
ThreadStateException, 1425
- ~InactivityMonitor
 - activemq::transport::inactivity::-
InactivityMonitor, 1426
- ~IndexOutOfBoundsException
 - decaf::lang::exceptions::IndexOutOf-
BoundsException, 1431
- ~Inet4Address
 - decaf::net::Inet4Address, 1432
- ~Inet6Address
 - decaf::net::Inet6Address, 1436
- ~InetAddress
 - decaf::net::InetAddress, 1439
- ~InetSocketAddress
 - decaf::net::InetSocketAddress, 1445
- ~Inflater
 - decaf::util::zip::Inflater, 1450
- ~InflaterInputStream
 - decaf::util::zip::InflaterInputStream,
1460
- ~InputStream
 - decaf::io::InputStream, 1466
- ~InputStreamReader
 - decaf::io::InputStreamReader, 1476
- ~IntArrayBuffer
 - decaf::internal::nio::IntArrayBuffer,
1482
- ~IntBuffer
 - decaf::nio::IntBuffer, 1490
- ~Integer
 - decaf::lang::Integer, 1503
- ~IntegerResponse
 - activemq::commands::Integer-
Response, 1517
- ~IntegerResponseMarshaller
 - activemq::wireformat::openwire-
::marshal::generated::Integer-
ResponseMarshaller, 1520
- ~InternalCommandListener
 - activemq::transport::mock::Internal-
CommandListener, 1528
- ~InterruptedException
 - decaf::lang::exceptions::Interrupted-
Exception, 1531
- ~InterruptedIOException
 - decaf::io::InterruptedIOException,
1533
- ~InvalidClientIdException
 - cms::InvalidClientIdException, 1535
- ~InvalidDestinationException
 - cms::InvalidDestinationException,
1536

- ~InvalidKeyException
 - decaf::security::InvalidKeyException, 1538
- ~InvalidMarkException
 - decaf::nio::InvalidMarkException, 1541
- ~InvalidSelectorException
 - cms::InvalidSelectorException, 1542
- ~InvalidStateException
 - decaf::lang::exceptions::InvalidStateException, 1545
- ~Iterable
 - decaf::lang::Iterable, 1557
- ~Iterator
 - decaf::util::Iterator, 1559
- ~JournalQueueAck
 - activemq::commands::JournalQueueAck, 1562
- ~JournalQueueAckMarshaller
 - activemq::wireformat::openwire::marshal::generated::JournalQueueAckMarshaller, 1565
- ~JournalTopicAck
 - activemq::commands::JournalTopicAck, 1569
- ~JournalTopicAckMarshaller
 - activemq::wireformat::openwire::marshal::generated::JournalTopicAckMarshaller, 1573
- ~JournalTrace
 - activemq::commands::JournalTrace, 1577
- ~JournalTraceMarshaller
 - activemq::wireformat::openwire::marshal::generated::JournalTraceMarshaller, 1580
- ~JournalTransaction
 - activemq::commands::JournalTransaction, 1584
- ~JournalTransactionMarshaller
 - activemq::wireformat::openwire::marshal::generated::JournalTransactionMarshaller, 1588
- ~KeepAliveInfo
 - activemq::commands::KeepAliveInfo, 1592
- ~KeepAliveInfoMarshaller
 - activemq::wireformat::openwire::marshal::generated::KeepAliveInfoMarshaller, 1595
- ~Key
 - decaf::security::Key, 1599
- ~KeyException
 - decaf::security::KeyException, 1602
- ~KeyManagementException
 - decaf::security::KeyManagementException, 1605
- ~LastPartialCommand
 - activemq::commands::LastPartialCommand, 1606
- ~LastPartialCommandMarshaller
 - activemq::wireformat::openwire::marshal::generated::LastPartialCommandMarshaller, 1609
- ~Less
 - decaf::util::comparators::Less, 1613
- ~Level
 - decaf::util::logging::Level, 1618
- ~LinkedBlockingQueue
 - decaf::util::concurrent::LinkedBlockingQueue, 1624
- ~LinkedList
 - decaf::util::LinkedList, 1639
- ~List
 - decaf::util::List, 1660
- ~ListIterator
 - decaf::util::ListIterator, 1672
- ~LocalTransactionId
 - activemq::commands::LocalTransactionId, 1676
- ~LocalTransactionIdMarshaller
 - activemq::wireformat::openwire::marshal::generated::LocalTransactionIdMarshaller, 1679
- ~Lock
 - decaf::util::concurrent::Lock, 1683
 - decaf::util::concurrent::locks::Lock, 1686
- ~LockSupport
 - decaf::util::concurrent::locks::LockSupport, 1691
- ~LogManager
 - decaf::util::logging::LogManager, 1714
- ~LogRecord
 - decaf::util::logging::LogRecord, 1720
- ~LogWriter
 - decaf::util::logging::LogWriter, 1725
- ~Logger

- decaf::util::logging::Logger, 1696
- ~LoggerHierarchy
 - decaf::util::logging::LoggerHierarchy, 1706
- ~LoggingInputStream
 - activemq::io::LoggingInputStream, 1707
- ~LoggingOutputStream
 - activemq::io::LoggingOutputStream, 1708
- ~LoggingTransport
 - activemq::transport::logging::LoggingTransport, 1710
- ~Long
 - decaf::lang::Long, 1729
- ~LongArrayBuffer
 - decaf::internal::nio::LongArrayBuffer, 1747
- ~LongBuffer
 - decaf::nio::LongBuffer, 1755
- ~LongSequenceGenerator
 - activemq::util::LongSequenceGenerator, 1765
- ~MalformedURLException
 - decaf::net::MalformedURLException, 1768
- ~Map
 - decaf::util::Map, 1770
- ~MapMessage
 - cms::MapMessage, 1782
- ~MarkBlockLogger
 - decaf::util::logging::MarkBlockLogger, 1792
- ~MarshalAware
 - activemq::wireformat::MarshalAware, 1793
- ~MarshallerFactory
 - activemq::wireformat::openwire::marshal::generated::MarshallerFactory, 1796
- ~MarshallingSupport
 - activemq::util::MarshallingSupport, 1797
- ~Math
 - decaf::lang::Math, 1802
- ~MemoryUsage
 - activemq::util::MemoryUsage, 1819
- ~Message
 - activemq::commands::Message, 1826
- cms::Message, 1843
- ~MessageAck
 - activemq::commands::MessageAck, 1869
- ~MessageAckMarshaller
 - activemq::wireformat::openwire::marshal::generated::MessageAckMarshaller, 1874
- ~MessageConsumer
 - cms::MessageConsumer, 1878
- ~MessageCreator
 - activemq::cmsutil::MessageCreator, 1881
- ~MessageDispatch
 - activemq::commands::MessageDispatch, 1883
- ~MessageDispatchChannel
 - activemq::core::MessageDispatchChannel, 1887
- ~MessageDispatchMarshaller
 - activemq::wireformat::openwire::marshal::generated::MessageDispatchMarshaller, 1891
- ~MessageDispatchNotification
 - activemq::commands::MessageDispatchNotification, 1896
- ~MessageDispatchNotificationMarshaller
 - activemq::wireformat::openwire::marshal::generated::MessageDispatchNotificationMarshaller, 1900
- ~MessageEOFException
 - cms::MessageEOFException, 1906
- ~MessageEnumeration
 - cms::MessageEnumeration, 1904
- ~MessageFormatException
 - cms::MessageFormatException, 1907
- ~MessageId
 - activemq::commands::MessageId, 1909
- ~MessageIdMarshaller
 - activemq::wireformat::openwire::marshal::generated::MessageIdMarshaller, 1913
- ~MessageListener
 - cms::MessageListener, 1917
- ~MessageMarshaller
 - activemq::wireformat::openwire::marshal::generated::Message-

- Marshaller, 1918
- ~MessageNotReadableException
 - cms::MessageNotReadableException, 1923
- ~MessageNotWriteableException
 - cms::MessageNotWriteableException, 1924
- ~MessageProducer
 - cms::MessageProducer, 1926
- ~MessagePropertyInterceptor
 - activemq::wireformat::openwire::utils::MessagePropertyInterceptor, 1934
- ~MessagePull
 - activemq::commands::MessagePull, 1941
- ~MessagePullMarshaller
 - activemq::wireformat::openwire::marshal::generated::MessagePullMarshaller, 1945
- ~MockTransport
 - activemq::transport::mock::MockTransport, 1950
- ~MockTransportFactory
 - activemq::transport::mock::MockTransportFactory, 1959
- ~Mutex
 - decaf::util::concurrent::Mutex, 1961
- ~MutexHandle
 - decaf::util::concurrent::MutexHandle, 1966
- ~Network
 - decaf::internal::net::Network, 1969
- ~NetworkBridgeFilter
 - activemq::commands::NetworkBridgeFilter, 1972
- ~NetworkBridgeFilterMarshaller
 - activemq::wireformat::openwire::marshal::generated::NetworkBridgeFilterMarshaller, 1975
- ~NoRouteToHostException
 - decaf::net::NoRouteToHostException, 1980
- ~NoSuchAlgorithmException
 - decaf::security::NoSuchAlgorithmException, 1983
- ~NoSuchElementException
 - decaf::util::NoSuchElementException, 1986
- ~NoSuchProviderException
 - decaf::security::NoSuchProviderException, 1988
- ~NullPointerException
 - decaf::lang::exceptions::NullPointerException, 1991
- ~Number
 - decaf::lang::Number, 1992
- ~NumberFormatException
 - decaf::lang::exceptions::NumberFormatException, 1997
- ~ObjectMessage
 - cms::ObjectMessage, 1998
- ~OpenSSLContextSpi
 - decaf::internal::net::ssl::openssl::OpenSSLContextSpi, 1999
- ~OpenSSLParameters
 - decaf::internal::net::ssl::openssl::OpenSSLParameters, 2002
- ~OpenSSLServerSocket
 - decaf::internal::net::ssl::openssl::OpenSSLServerSocket, 2005
- ~OpenSSLServerSocketFactory
 - decaf::internal::net::ssl::openssl::OpenSSLServerSocketFactory, 2011
- ~OpenSSLSocket
 - decaf::internal::net::ssl::openssl::OpenSSLSocket, 2020
- ~OpenSSLSocketException
 - decaf::internal::net::ssl::openssl::OpenSSLSocketException, 2032
- ~OpenSSLSocketFactory
 - decaf::internal::net::ssl::openssl::OpenSSLSocketFactory, 2036
- ~OpenSSLSocketInputStream
 - decaf::internal::net::ssl::openssl::OpenSSLSocketInputStream, 2041
- ~OpenSSLSocketOutputStream
 - decaf::internal::net::ssl::openssl::OpenSSLSocketOutputStream, 2044
- ~OpenWireFormat
 - activemq::wireformat::openwire::OpenWireFormat, 2048
- ~OpenWireFormatFactory
 - activemq::wireformat::openwire::OpenWireFormatFactory, 2059
- ~OpenWireFormatNegotiator

- activemq::wireformat::openwire::-
OpenWireFormatNegotiator,
2061
- ~OpenWireResponseBuilder
 - activemq::wireformat::openwire::-
OpenWireResponseBuilder,
2065
- ~OutputStream
 - decaf::io::OutputStream, 2068
- ~OutputStreamWriter
 - decaf::io::OutputStreamWriter, 2075
- ~PartialCommand
 - activemq::commands::Partial-
Command, 2077
- ~PartialCommandMarshaller
 - activemq::wireformat::openwire-
::marshal::generated::Partial-
CommandMarshaller, 2080
- ~Pointer
 - decaf::lang::Pointer, 2087
- ~PointerComparator
 - decaf::lang::PointerComparator,
2092
- ~PooledSession
 - activemq::cmsutil::PooledSession,
2095
- ~PortUnreachableException
 - decaf::net::PortUnreachableException,
2109
- ~PrefetchPolicy
 - activemq::core::PrefetchPolicy, 2111
- ~PrimitiveList
 - activemq::util::PrimitiveList, 2116
- ~PrimitiveMap
 - activemq::util::PrimitiveMap, 2127
- ~PrimitiveTypesMarshaller
 - activemq::wireformat::openwire-
::marshal::PrimitiveTypes-
Marshaller, 2137
- ~PrimitiveValueConverter
 - activemq::util::PrimitiveValueConverter, 2144
- ~PrimitiveValueNode
 - activemq::util::PrimitiveValueNode,
2151
- ~Principal
 - decaf::security::Principal, 2160
- ~PriorityQueue
 - decaf::util::PriorityQueue, 2165
- ~ProducerAck
 - activemq::commands::ProducerAck,
2172
- ~ProducerAckMarshaller
 - activemq::wireformat::openwire-
::marshal::generated::Producer-
AckMarshaller, 2176
- ~ProducerCallback
 - activemq::cmsutil::ProducerCallback,
2179
- ~ProducerExecutor
 - activemq::cmsutil::CmsTemplate::-
ProducerExecutor, 2181
- ~ProducerId
 - activemq::commands::ProducerId,
2183
- ~ProducerIdMarshaller
 - activemq::wireformat::openwire-
::marshal::generated::Producer-
IdMarshaller, 2187
- ~ProducerInfo
 - activemq::commands::ProducerInfo,
2191
- ~ProducerInfoMarshaller
 - activemq::wireformat::openwire-
::marshal::generated::Producer-
InfoMarshaller, 2196
- ~ProducerState
 - activemq::state::ProducerState, 2199
- ~Properties
 - decaf::util::Properties, 2202
- ~PropertiesChangeListener
 - decaf::util::logging::Properties-
ChangeListener, 2210
- ~ProtocolException
 - decaf::net::ProtocolException, 2213
- ~PublicKey
 - decaf::security::PublicKey, 2214
- ~PushbackInputStream
 - decaf::io::PushbackInputStream,
2217
- ~Queue
 - cms::Queue, 2222
 - decaf::util::Queue, 2223
- ~QueueBrowser
 - cms::QueueBrowser, 2228
- ~Random
 - decaf::util::Random, 2231
- ~ReadChecker
 - activemq::transport::inactivity::Read-
Checker, 2237

- ~ReadOnlyBufferException
 - decaf::nio::ReadOnlyBufferException, 2246
- ~ReadWriteLock
 - decaf::util::concurrent::locks::ReadWriteLock, 2248
- ~Readable
 - decaf::lang::Readable, 2236
- ~Reader
 - decaf::io::Reader, 2239
- ~ReceiveExecutor
 - activemq::cmsutil::CmsTemplate::ReceiveExecutor, 2249
- ~RedeliveryPolicy
 - activemq::core::RedeliveryPolicy, 2252
- ~ReentrantLock
 - decaf::util::concurrent::locks::ReentrantLock, 2258
- ~RejectedExecutionException
 - decaf::util::concurrent::RejectedExecutionException, 2265
- ~RejectedExecutionHandler
 - decaf::util::concurrent::RejectedExecutionHandler, 2267
- ~RemoveInfo
 - activemq::commands::RemoveInfo, 2268
- ~RemoveInfoMarshaller
 - activemq::wireformat::openwire::marshal::generated::RemoveInfoMarshaller, 2272
- ~RemoveSubscriptionInfo
 - activemq::commands::RemoveSubscriptionInfo, 2276
- ~RemoveSubscriptionInfoMarshaller
 - activemq::wireformat::openwire::marshal::generated::RemoveSubscriptionInfoMarshaller, 2281
- ~ReplayCommand
 - activemq::commands::ReplayCommand, 2285
- ~ReplayCommandMarshaller
 - activemq::wireformat::openwire::marshal::generated::ReplayCommandMarshaller, 2288
- ~ResolveProducerExecutor
 - activemq::cmsutil::CmsTemplate::ResolveProducerExecutor, 2291
- ~ResolveReceiveExecutor
 - activemq::cmsutil::CmsTemplate::ResolveReceiveExecutor, 2292
- ~Resource
 - decaf::internal::util::Resource, 2293
- ~ResourceLifecycleManager
 - activemq::cmsutil::ResourceLifecycleManager, 2294
 - decaf::internal::util::ResourceLifecycleManager, 2297
- ~Response
 - activemq::commands::Response, 2299
- ~ResponseBuilder
 - activemq::transport::mock::ResponseBuilder, 2302
- ~ResponseCorrelator
 - activemq::transport::correlator::ResponseCorrelator, 2304
- ~ResponseMarshaller
 - activemq::wireformat::openwire::marshal::generated::ResponseMarshaller, 2308
- ~Runnable
 - decaf::lang::Runnable, 2312
- ~Runtime
 - decaf::lang::Runtime, 2313
- ~RuntimeException
 - decaf::lang::exceptions::RuntimeException, 2317
- ~SSLContext
 - decaf::net::ssl::SSLContext, 2496
- ~SSLContextSpi
 - decaf::net::ssl::SSLContextSpi, 2499
- ~SSLParameters
 - decaf::net::ssl::SSLParameters, 2503
- ~SSLServerSocket
 - decaf::net::ssl::SSLServerSocket, 2507
- ~SSLServerSocketFactory
 - decaf::net::ssl::SSLServerSocketFactory, 2511
- ~SSLSocket
 - decaf::net::ssl::SSLSocket, 2517
- ~SSLSocketFactory
 - decaf::net::ssl::SSLSocketFactory, 2523
- ~Scheduler

- activemq::threads::Scheduler, 2318
- ~SchedulerTimerTask
 - activemq::threads::SchedulerTimerTask, 2320
- ~SecureRandom
 - decaf::security::SecureRandom, 2323
- ~SecureRandomImpl
 - decaf::internal::security::SecureRandomImpl, 2327
- ~SecureRandomSpi
 - decaf::security::SecureRandomSpi, 2330
- ~Semaphore
 - decaf::util::concurrent::Semaphore, 2334
- ~SendExecutor
 - activemq::cmsutil::CmsTemplate::SendExecutor, 2342
- ~ServerSocket
 - decaf::net::ServerSocket, 2346
- ~ServerSocketFactory
 - decaf::net::ServerSocketFactory, 2353
- ~Service
 - activemq::util::Service, 2356
- ~ServiceListener
 - activemq::util::ServiceListener, 2357
- ~ServiceStopper
 - activemq::util::ServiceStopper, 2358
- ~ServiceSupport
 - activemq::util::ServiceSupport, 2360
- ~Session
 - cms::Session, 2365
- ~SessionCallback
 - activemq::cmsutil::SessionCallback, 2377
- ~SessionId
 - activemq::commands::SessionId, 2379
- ~SessionIdMarshaller
 - activemq::wireformat::openwire::marshal::generated::SessionIdMarshaller, 2383
- ~SessionInfo
 - activemq::commands::SessionInfo, 2387
- ~SessionInfoMarshaller
 - activemq::wireformat::openwire::marshal::generated::SessionInfoMarshaller, 2391
- ~SessionPool
 - activemq::cmsutil::SessionPool, 2394
- ~SessionState
 - activemq::state::SessionState, 2396
- ~Set
 - decaf::util::Set, 2398
- ~Short
 - decaf::lang::Short, 2400
- ~ShortArrayBuffer
 - decaf::internal::nio::ShortArrayBuffer, 2413
- ~ShortBuffer
 - decaf::nio::ShortBuffer, 2421
- ~ShutdownInfo
 - activemq::commands::ShutdownInfo, 2432
- ~ShutdownInfoMarshaller
 - activemq::wireformat::openwire::marshal::generated::ShutdownInfoMarshaller, 2435
- ~SignatureException
 - decaf::security::SignatureException, 2440
- ~SimpleFormatter
 - decaf::util::logging::SimpleFormatter, 2441
- ~SimpleLogger
 - decaf::util::logging::SimpleLogger, 2443
- ~SimplePriorityMessageDispatchChannel
 - activemq::core::SimplePriorityMessageDispatchChannel, 2445
- ~Socket
 - decaf::net::Socket, 2457
- ~SocketAddress
 - decaf::net::SocketAddress, 2470
- ~SocketException
 - decaf::net::SocketException, 2473
- ~SocketFactory
 - decaf::net::SocketFactory, 2475
- ~SocketFileDescriptor
 - decaf::internal::net::SocketFileDescriptor, 2479
- ~SocketImpl
 - decaf::net::SocketImpl, 2481
- ~SocketImplFactory
 - decaf::net::SocketImplFactory, 2488

- ~SocketOptions
 - decaf::net::SocketOptions, 2490
- ~SocketTimeoutException
 - decaf::net::SocketTimeoutException, 2495
- ~SslTransport
 - activemq::transport::tcp::SslTransport, 2526
- ~SslTransportFactory
 - activemq::transport::tcp::SslTransportFactory, 2527
- ~StandardErrorOutputStream
 - decaf::internal::io::StandardErrorOutputStream, 2529
- ~StandardInputStream
 - decaf::internal::io::StandardInputStream, 2531
- ~StandardOutputStream
 - decaf::internal::io::StandardOutputStream, 2532
- ~Startable
 - cms::Startable, 2534
- ~StaticInitializer
 - activemq::core::ActiveMQConstants::StaticInitializer, 2535
- ~StlMap
 - decaf::util::StlMap, 2556
- ~StlQueue
 - decaf::util::StlQueue, 2567
- ~StlSet
 - decaf::util::StlSet, 2577
- ~StompFrame
 - activemq::wireformat::stomp::StompFrame, 2589
- ~StompHelper
 - activemq::wireformat::stomp::StompHelper, 2594
- ~StompWireFormat
 - activemq::wireformat::stomp::StompWireFormat, 2598
- ~StompWireFormatFactory
 - activemq::wireformat::stomp::StompWireFormatFactory, 2601
- ~Stoppable
 - cms::Stoppable, 2602
- ~StreamHandler
 - decaf::util::logging::StreamHandler, 2604
- ~StreamMessage
 - cms::StreamMessage, 2608
- ~String
 - decaf::lang::String, 2624
- ~StringTokenizer
 - decaf::util::StringTokenizer, 2628
- ~SubscriptionInfo
 - activemq::commands::SubscriptionInfo, 2632
- ~SubscriptionInfoMarshaller
 - activemq::wireformat::openwire::marshal::generated::SubscriptionInfoMarshaller, 2636
- ~Synchronizable
 - decaf::util::concurrent::Synchronizable, 2640
- ~SynchronizableImpl
 - decaf::internal::util::concurrent::SynchronizableImpl, 2651
- ~Synchronization
 - activemq::core::Synchronization, 2654
- ~SynchronousQueue
 - decaf::util::concurrent::SynchronousQueue, 2657
- ~System
 - decaf::lang::System, 2667
- ~Task
 - activemq::threads::Task, 2676
- ~TaskRunner
 - activemq::threads::TaskRunner, 2677
- ~TcpSocket
 - decaf::internal::net::tcp::TcpSocket, 2681
- ~TcpSocketInputStream
 - decaf::internal::net::tcp::TcpSocketInputStream, 2689
- ~TcpSocketOutputStream
 - decaf::internal::net::tcp::TcpSocketOutputStream, 2692
- ~TcpTransport
 - activemq::transport::tcp::TcpTransport, 2694
- ~TcpTransportFactory
 - activemq::transport::tcp::TcpTransportFactory, 2697
- ~TemporaryQueue
 - cms::TemporaryQueue, 2699
- ~TemporaryTopic
 - cms::TemporaryTopic, 2700
- ~TextMessage

- cms::TextMessage, 2702
- ~Thread
 - decaf::lang::Thread, 2708
- ~ThreadFactory
 - decaf::util::concurrent::ThreadFactory, 2714
- ~ThreadGroup
 - decaf::lang::ThreadGroup, 2715
- ~ThreadPoolExecutor
 - decaf::util::concurrent::ThreadPoolExecutor, 2722
- ~Throwable
 - decaf::lang::Throwable, 2733
- ~TimeUnit
 - decaf::util::concurrent::TimeUnit, 2758
- ~TimeoutException
 - decaf::util::concurrent::TimeoutException, 2739
- ~Timer
 - decaf::util::Timer, 2741
- ~TimerTask
 - decaf::util::TimerTask, 2751
- ~TimerTaskHeap
 - decaf::internal::util::TimerTaskHeap, 2754
- ~Topic
 - cms::Topic, 2765
- ~Tracked
 - activemq::state::Tracked, 2766
- ~TransactionId
 - activemq::commands::TransactionId, 2767
- ~TransactionIdMarshaller
 - activemq::wireformat::openwire::marshal::generated::TransactionIdMarshaller, 2771
- ~TransactionInProgressException
 - cms::TransactionInProgressException, 2783
- ~TransactionInfo
 - activemq::commands::TransactionInfo, 2775
- ~TransactionInfoMarshaller
 - activemq::wireformat::openwire::marshal::generated::TransactionInfoMarshaller, 2779
- ~TransactionRolledBackException
 - cms::TransactionRolledBackException, 2784
- ~TransactionState
 - activemq::state::TransactionState, 2785
- ~TransferQueue
 - decaf::internal::util::concurrent::TransferQueue, 2787
- ~TransferStack
 - decaf::internal::util::concurrent::TransferStack, 2789
- ~Transport
 - activemq::transport::Transport, 2792
- ~TransportFactory
 - activemq::transport::TransportFactory, 2799
- ~TransportFilter
 - activemq::transport::TransportFilter, 2802
- ~TransportListener
 - activemq::transport::TransportListener, 2811
- ~TransportRegistry
 - activemq::transport::TransportRegistry, 2813
- ~URI
 - decaf::net::URI, 2833
- ~URIEncoderDecoder
 - decaf::internal::net::URIEncoderDecoder, 2842
- ~URIHelper
 - decaf::internal::net::URIHelper, 2846
- ~URIPool
 - activemq::transport::failover::URIPool, 2852
- ~URISyntaxException
 - decaf::net::URISyntaxException, 2859
- ~URIType
 - decaf::internal::net::URIType, 2862
- ~URL
 - decaf::net::URL, 2870
- ~URLDecoder
 - decaf::net::URLDecoder, 2870
- ~URLEncoder
 - decaf::net::URLEncoder, 2871
- ~UTFDataFormatException
 - decaf::io::UTFDataFormatException, 2876
- ~UUID
 - decaf::util::UUID, 2879
- ~UncaughtExceptionHandler

- decaf::lang::Thread::Uncaught-Exception
ExceptionHandler, 2816
- ~UnknownHostException
 - decaf::net::UnknownHostException, 2818
- ~UnknownServiceException
 - decaf::net::UnknownServiceException, 2821
- ~UnsupportedEncodingException
 - decaf::io::UnsupportedEncodingException, 2824
- ~UnsupportedOperationException
 - cms::UnsupportedOperationException, 2828
 - decaf::lang::exceptions::Unsupported-OperationException, 2826
- ~Usage
 - activemq::util::Usage, 2873
- ~WireFormat
 - activemq::wireformat::WireFormat, 2885
- ~WireFormatFactory
 - activemq::wireformat::WireFormat-Factory, 2889
- ~WireFormatInfo
 - activemq::commands::WireFormat-Info, 2892
- ~WireFormatInfoMarshaller
 - activemq::wireformat::openwire-::marshal::generated::Wire-FormatInfoMarshaller, 2901
- ~WireFormatNegotiator
 - activemq::wireformat::WireFormat-Negotiator, 2904
- ~WireFormatRegistry
 - activemq::wireformat::WireFormat-Registry, 2906
- ~WriteChecker
 - activemq::transport::inactivity::Write-Checker, 2908
- ~Writer
 - decaf::io::Writer, 2910
- ~X500Principal
 - decaf::security::auth::x500::X500-Principal, 2914
- ~X509Certificate
 - decaf::security::cert::X509Certificate, 2916
- ~XAConnection
 - cms::XAConnection, 2917
- ~XAConnectionFactory
 - cms::XAConnectionFactory, 2919
- ~XAException
 - cms::XAException, 2923
- ~XAResource
 - cms::XAResource, 2928
- ~XASession
 - cms::XASession, 2935
- ~XATransactionId
 - activemq::commands::XATransaction-Id, 2938
- ~XATransactionIdMarshaller
 - activemq::wireformat::openwire-::marshal::generated::XA-TransactionIdMarshaller, 2943
- ~XMLFormatter
 - decaf::util::logging::XMLFormatter, 2950
- ~Xid
 - cms::Xid, 2948
- ~ZipException
 - decaf::util::zip::ZipException, 2955
- ABORT
 - activemq::wireformat::stomp::Stomp-CommandConstants, 2583
- ACK
 - activemq::wireformat::stomp::Stomp-CommandConstants, 2583
- ACK_AUTO
 - activemq::wireformat::stomp::Stomp-CommandConstants, 2583
- ACK_CLIENT
 - activemq::wireformat::stomp::Stomp-CommandConstants, 2583
- ACK_INDIVIDUAL
 - activemq::wireformat::stomp::Stomp-CommandConstants, 2583
- ACK_TYPE_CONSUMED
 - activemq::core::ActiveMQConstants, 232
- ACK_TYPE_DELIVERED
 - activemq::core::ActiveMQConstants, 232
- ACK_TYPE_INDIVIDUAL
 - activemq::core::ActiveMQConstants, 232
- ACK_TYPE_POISON
 - activemq::core::ActiveMQConstants, 232
- ACK_TYPE_REDELIVERED

- activemq::core::ActiveMQConstants, 232
- ADVISORY_PREFIX
 - activemq::commands::ActiveMQ-Destination, 256
- ALL
 - decaf::util::logging::Level, 1619
- AMQCPP_API
 - activemq/util/Config.h, 3039
- AMQ_CATCHALL_NOTHROW
 - activemq/exceptions/Exception-Defines.h, 3009
- AMQ_CATCHALL_THROW
 - activemq/exceptions/Exception-Defines.h, 3009
- AMQ_CATCH_ALL_THROW_CMSEXCEPTION
 - CMSExcceptionSupport.h, 3038
- AMQ_CATCH_EXCEPTION_CONVERT
 - activemq/exceptions/Exception-Defines.h, 3008
- AMQ_CATCH_NOTHROW
 - activemq/exceptions/Exception-Defines.h, 3008
- AMQ_CATCH_RETHROW
 - activemq/exceptions/Exception-Defines.h, 3008
- ANY_CHILD
 - activemq::commands::ActiveMQ-Destination::DestinationFilter, 1213
- ANY_DESCENDENT
 - activemq::commands::ActiveMQ-Destination::DestinationFilter, 1213
- AUTO_ACKNOWLEDGE
 - cms::Session, 2365
- AbortPolicy
 - decaf::util::concurrent::ThreadPool-Executor::AbortPolicy, 105
- AbstractCollection
 - decaf::util::AbstractCollection, 110
- AbstractExecutorService
 - decaf::util::concurrent::Abstract-ExecutorService, 122
- AbstractList
 - decaf::util::AbstractList, 124
- AbstractOwnableSynchronizer
 - decaf::util::concurrent::locks:-AbstractOwnableSynchronizer, 136
- AbstractQueue
 - decaf::util::AbstractQueue, 139
- AckType
 - activemq::core::ActiveMQConstants, 232
- AcknowledgeMode
 - cms::Session, 2365
- ActiveMQBlobMessage
 - activemq::commands::ActiveMQ-BlobMessage, 158
- ActiveMQBlobMessageMarshaller
 - activemq::wireformat::openwire-::marshal::generated::Active-MQBlobMessageMarshaller, 162
- ActiveMQBytesMessage
 - activemq::commands::ActiveMQ-BytesMessage, 168
- ActiveMQBytesMessageMarshaller
 - activemq::wireformat::openwire-::marshal::generated::Active-MQBytesMessageMarshaller, 184
- ActiveMQCPP
 - activemq::library::ActiveMQCPP, 245
- ActiveMQConnection
 - activemq::core::ActiveMQConnection, 193
- ActiveMQConnectionFactory
 - activemq::core::ActiveMQConnection-Factory, 216
- ActiveMQConnectionMetaData
 - activemq::core::ActiveMQConnection-MetaData, 228
- ActiveMQConsumer
 - activemq::core::ActiveMQConsumer, 236
- ActiveMQDestination
 - activemq::commands::ActiveMQ-Destination, 249
- ActiveMQDestinationMarshaller
 - activemq::wireformat::openwire-::marshal::generated::Active-MQDestinationMarshaller, 258
- ActiveMQException
 - activemq::exceptions::ActiveMQ-Exception, 262, 263
- ActiveMQMapMessage

- activemq::commands::ActiveMQ-MapMessage, 270
- ActiveMQMapMessageMarshaller
 - activemq::wireformat::openwire::marshal::generated::ActiveMQMapMessageMarshaller, 285
- ActiveMQMessage
 - activemq::commands::ActiveMQ-Message, 289
- ActiveMQMessageMarshaller
 - activemq::wireformat::openwire::marshal::generated::ActiveMQMessageMarshaller, 292
- ActiveMQMessageTemplate
 - activemq::commands::ActiveMQ-MessageTemplate, 296
- ActiveMQObjectMessage
 - activemq::commands::ActiveMQ-ObjectMessage, 302
- ActiveMQObjectMessageMarshaller
 - activemq::wireformat::openwire::marshal::generated::ActiveMQObjectMessageMarshaller, 305
- ActiveMQProducer
 - activemq::core::ActiveMQProducer, 309
- ActiveMQProperties
 - activemq::util::ActiveMQProperties, 317
- ActiveMQQueue
 - activemq::commands::ActiveMQ-Queue, 322
- ActiveMQQueueBrowser
 - activemq::core::ActiveMQQueue-Browser, 326
- ActiveMQQueueMarshaller
 - activemq::wireformat::openwire::marshal::generated::ActiveMQQueueMarshaller, 330
- ActiveMQSession
 - activemq::core::ActiveMQSession, 337
- ActiveMQSessionExecutor
 - activemq::core::ActiveMQSession, 354
 - activemq::core::ActiveMQSession-Executor, 356
- ActiveMQStreamMessage
 - activemq::commands::ActiveMQ-StreamMessage, 361
- ActiveMQStreamMessageMarshaller
 - activemq::wireformat::openwire::marshal::generated::ActiveMQStreamMessageMarshaller, 376
- ActiveMQTempDestination
 - activemq::commands::ActiveMQ-TempDestination, 380
- ActiveMQTempDestinationMarshaller
 - activemq::wireformat::openwire::marshal::generated::ActiveMQTempDestinationMarshaller, 383
- ActiveMQTempQueue
 - activemq::commands::ActiveMQ-TempQueue, 387
- ActiveMQTempQueueMarshaller
 - activemq::wireformat::openwire::marshal::generated::ActiveMQTempQueueMarshaller, 392
- ActiveMQTempTopic
 - activemq::commands::ActiveMQ-TempTopic, 397
- ActiveMQTempTopicMarshaller
 - activemq::wireformat::openwire::marshal::generated::ActiveMQTempTopicMarshaller, 402
- ActiveMQTextMessage
 - activemq::commands::ActiveMQ-TextMessage, 406
- ActiveMQTextMessageMarshaller
 - activemq::wireformat::openwire::marshal::generated::ActiveMQTextMessageMarshaller, 411
- ActiveMQTopic
 - activemq::commands::ActiveMQ-Topic, 415
- ActiveMQTopicMarshaller
 - activemq::wireformat::openwire::marshal::generated::ActiveMQTopicMarshaller, 420
- ActiveMQTransactionContext
 - activemq::core::ActiveMQTransaction-Context, 425
- ActiveMQXAConnection
 - activemq::core::ActiveMQXAConnection, 432
- ActiveMQXAConnectionFactory

- activemq::core::ActiveMQXAConnection-Factory, 434
- ActiveMQXASession
 - activemq::core::ActiveMQXASession, 437
- Adler32
 - decaf::util::zip::Adler32, 440
- AprPool
 - decaf::internal::AprPool, 445
- ArrayList
 - decaf::util::ArrayList, 448
- ArrayListIterator
 - decaf::util::concurrent::CopyOnWriteArrayList::ArrayListIterator, 459
- ArrayPointer
 - decaf::lang::ArrayPointer, 464, 465
- Assert
 - zutil.h, 3162
- AsyncSignalReadErrorTask
 - activemq::transport::inactivity::InactivityMonitor, 1428
- AsyncWriteTask
 - activemq::transport::inactivity::InactivityMonitor, 1428
- AtomicBoolean
 - decaf::util::concurrent::atomic::AtomicBoolean, 473
- AtomicInteger
 - decaf::util::concurrent::atomic::AtomicInteger, 477
- AtomicRefCounter
 - decaf::util::concurrent::atomic::AtomicRefCounter, 482
- AtomicReference
 - decaf::util::concurrent::atomic::AtomicReference, 485
- BAD
 - inflate.h, 3149
- BEGIN
 - activemq::wireformat::stomp::StompCommandConstants, 2583
- BEST_COMPRESSION
 - decaf::util::zip::Deflater, 1188
- BEST_SPEED
 - decaf::util::zip::Deflater, 1188
- BIG_STRING_TYPE
 - activemq::util::PrimitiveValueNode, 2148
- BINARY_MIME_TYPE
 - activemq::commands::ActiveMQBlobMessage, 161
- BLOCKED
 - decaf::lang::Thread, 2707
- BL_CODES
 - deflate.h, 3144
- BOOLEAN_TYPE
 - activemq::util::PrimitiveValueNode, 2148
- BUSY_STATE
 - deflate.h, 3144
- BYTES
 - activemq::wireformat::stomp::StompCommandConstants, 2583
- BYTE_ARRAY_TYPE
 - activemq::util::PrimitiveValueNode, 2148
- BYTE_TYPE
 - activemq::util::PrimitiveValueNode, 2148
- BackupTransport
 - activemq::transport::failover::BackupTransport, 487
 - activemq::transport::failover::BackupTransportPool, 491
- BackupTransportPool
 - activemq::transport::failover::BackupTransportPool, 490
- BaseCommand
 - activemq::commands::BaseCommand, 493
- BaseCommandMarshaller
 - activemq::wireformat::openwire::marshal::generated::BaseCommandMarshaller, 500
- BindException
 - decaf::net::BindException, 533, 534
- BlockingByteArrayInputStream
 - decaf::io::BlockingByteArrayInputStream, 536
- Boolean
 - decaf::lang::Boolean, 546
- BooleanExpression
 - activemq::commands::BooleanExpression, 551
- BooleanStream
 - activemq::wireformat::openwire::utils::BooleanStream, 553
- BrokenBarrierException

- decaf::util::concurrent::BrokenBarrierException, 556, 557
- BrokerError
 - activemq::commands::BrokerError, 560
- BrokerException
 - activemq::exceptions::BrokerException, 563
- BrokerId
 - activemq::commands::BrokerId, 565
- BrokerIdMarshaller
 - activemq::wireformat::openwire::marshal::generated::BrokerIdMarshaller, 568
- BrokerInfo
 - activemq::commands::BrokerInfo, 573
- BrokerInfoMarshaller
 - activemq::wireformat::openwire::marshal::generated::BrokerInfoMarshaller, 579
- Browser
 - activemq::core::ActiveMQQueueBrowser, 329
- Buffer
 - decaf::nio::Buffer, 585
- BufferOverflowException
 - decaf::nio::BufferOverflowException, 610, 611
- BufferUnderflowException
 - decaf::nio::BufferUnderflowException, 612, 613
- BufferedInputStream
 - decaf::io::BufferedInputStream, 591
- BufferedOutputStream
 - decaf::io::BufferedOutputStream, 596
- Byte
 - decaf::lang::Byte, 616
 - zconf.h, 3153
- ByteArrayAdapter
 - decaf::internal::util::ByteArrayAdapter, 627–629
- ByteArrayBuffer
 - decaf::internal::nio::ByteArrayBuffer, 659, 660
- ByteArrayInputStream
 - decaf::io::ByteArrayInputStream, 682, 683
- ByteArrayOutputStream
 - decaf::io::ByteArrayOutputStream, 688
- ByteBuffer
 - decaf::nio::ByteBuffer, 694
- Bytef
 - zconf.h, 3153
- activemq::util::PrimitiveValueNode, 2148
- CHECK
 - inflate.h, 3149
- CLIENT_ACKNOWLEDGE
 - cms::Session, 2365
- CLOSE_FAILURE
 - decaf::util::logging::ErrorManager, 1278
- CMSException
 - cms::CMSException, 827
- CMSExceptionSupport.h
 - AMQ_CATCH_ALL_THROW_CMSEXCEPTION, 3038
- CMSSecurityException
 - cms::CMSSecurityException, 835, 836
- CMS_API
 - cms/Config.h, 3040
- CODELENS
 - inflate.h, 3149
- CODES
 - inftrees.h, 3150
- COMMENT
 - inflate.h, 3149
- COMMENT_STATE
 - deflate.h, 3144
- COMMIT
 - activemq::wireformat::stomp::StompCommandConstants, 2583
- COMPARATOR
 - activemq::commands::BrokerId, 565
 - activemq::commands::ConnectionId, 961
 - activemq::commands::ConsumerId, 1002
 - activemq::commands::LocalTransactionId, 1675
 - activemq::commands::MessageId, 1909
 - activemq::commands::ProducerId, 2183

- activemq::commands::SessionId, 2379
- activemq::commands::TransactionId, 2767
- activemq::commands::XATransactionId, 2938
- COMPOSITE_SEPARATOR
 - activemq::commands::ActiveMQ-Destination, 256
- CONFIG
 - decaf::util::logging::Level, 1619
- CONNECT
 - activemq::wireformat::stomp::Stomp-CommandConstants, 2583
- CONNECTED
 - activemq::wireformat::stomp::Stomp-CommandConstants, 2583
- CONNECTION_ADVISORY_PREFIX
 - activemq::commands::ActiveMQ-Destination, 256
- CONNECTION_ALWAYS_SYNC_SEND
 - activemq::core::ActiveMQConstants, 233
- CONNECTION_CLOSE_TIMEOUT
 - activemq::core::ActiveMQConstants, 233
- CONNECTION_DISPATCH_ASYNC
 - activemq::core::ActiveMQConstants, 233
- CONNECTION_PRODUCER_WINDOW_SIZE
 - activemq::core::ActiveMQConstants, 233
- CONNECTION_SEND_TIMEOUT
 - activemq::core::ActiveMQConstants, 233
- CONNECTION_USE_ASYNC_SEND
 - activemq::core::ActiveMQConstants, 233
- CONNECTION_USE_COMPRESSION
 - activemq::core::ActiveMQConstants, 233
- CONSUMER_ADVISORY_PREFIX
 - activemq::commands::ActiveMQ-Destination, 256
- CONSUMER_DISPATCH_ASYNC
 - activemq::core::ActiveMQConstants, 232
- CONSUMER_EXCLUSIVE
 - activemq::core::ActiveMQConstants, 232
- CONSUMER_NOLOCAL
 - activemq::core::ActiveMQConstants, 232
- CONSUMER_PREFETCH_SIZE
 - activemq::core::ActiveMQConstants, 232
- CONSUMER_PRIORITY
 - activemq::core::ActiveMQConstants, 232
- CONSUMER_RETROACTIVE
 - activemq::core::ActiveMQConstants, 232
- CONSUMER_SELECTOR
 - activemq::core::ActiveMQConstants, 232
- COPY
 - gzguts.h, 3147
 - inflate.h, 3149
- COPY_
 - inflate.h, 3149
- CRC32
 - decaf::util::zip::CRC32, 1066
- CUNSUMER_MAX_PENDING_MSG_LIMIT
 - activemq::core::ActiveMQConstants, 232
- CachedConsumer
 - activemq::cmsutil::CachedConsumer, 735
- CachedProducer
 - activemq::cmsutil::CachedProducer, 740
- CallerRunsPolicy
 - decaf::util::concurrent::ThreadPool-Executor::CallerRunsPolicy, 748
- CancellationException
 - decaf::util::concurrent::Cancellation-Exception, 749, 750
- CertificateEncodingException
 - decaf::security::cert::Certificate-EncodingException, 755, 756
- CertificateException
 - decaf::security::cert::Certificate-Exception, 757, 758
- CertificateExpiredException
 - decaf::security::cert::Certificate-ExpiredException, 759, 760
- CertificateNotYetValidException

- decaf::security::cert::CertificateNotYetValidException, 761, 762
- CertificateParsingException
 - decaf::security::cert::CertificateParsingException, 763, 764
- CharArrayBuffer
 - decaf::internal::nio::CharArrayBuffer, 777, 778
- CharBuffer
 - decaf::nio::CharBuffer, 788
- Character
 - decaf::lang::Character, 766
- CheckedInputStream
 - decaf::util::zip::CheckedInputStream, 806
- CheckedOutputStream
 - decaf::util::zip::CheckedOutputStream, 809
- ClassCastException
 - decaf::lang::exceptions::ClassCastException, 813, 814
- ClassName
 - activemq::commands::BrokerError::StackTraceElement, 2528
- CloseTransportsTask
 - activemq::transport::failover::CloseTransportsTask, 818
- CmsAccessor
 - activemq::cmsutil::CmsAccessor, 820
- CmsDestinationAccessor
 - activemq::cmsutil::CmsDestinationAccessor, 824
- CmsTemplate
 - activemq::cmsutil::CmsTemplate, 839
- Code
 - deflate.h, 3144
- CompositeData
 - activemq::util::CompositeData, 891
- CompositeTaskRunner
 - activemq::threads::CompositeTaskRunner, 894
- Concurrent.h
 - WAIT_INFINITE, 3231
 - synchronized, 3231
- ConcurrentModificationException
 - decaf::util::ConcurrentModificationException, 903, 904
- ConcurrentStlMap
 - decaf::util::concurrent::ConcurrentStlMap, 909
- ConditionHandle
 - decaf::util::concurrent::ConditionHandle, 928
- ConnectException
 - decaf::net::ConnectException, 932, 933
- ConnectionControl
 - activemq::commands::ConnectionControl, 940
- ConnectionControlMarshaller
 - activemq::wireformat::openwire::marshal::generated::ConnectionControlMarshaller, 944
- ConnectionError
 - activemq::commands::ConnectionError, 949
- ConnectionErrorMarshaller
 - activemq::wireformat::openwire::marshal::generated::ConnectionErrorMarshaller, 952
- ConnectionFailedException
 - activemq::exceptions::ConnectionFailedException, 959
- ConnectionId
 - activemq::commands::ConnectionId, 961
- ConnectionIdMarshaller
 - activemq::wireformat::openwire::marshal::generated::ConnectionIdMarshaller, 964
- ConnectionInfo
 - activemq::commands::ConnectionInfo, 969
- ConnectionInfoMarshaller
 - activemq::wireformat::openwire::marshal::generated::ConnectionInfoMarshaller, 975
- ConnectionState
 - activemq::state::ConnectionState, 983
- ConnectionStateTracker
 - activemq::state::ConnectionStateTracker, 986
- ConsoleHandler
 - decaf::util::logging::ConsoleHandler, 991
- ConstReferenceType
 - decaf::lang::ArrayPointer, 464

- ConsumerControl
 - activemq::commands::ConsumerControl, 993
- ConsumerControlMarshaller
 - activemq::wireformat::openwire::marshal::generated::ConsumerControlMarshaller, 997
- ConsumerId
 - activemq::commands::ConsumerId, 1002
- ConsumerIdMarshaller
 - activemq::wireformat::openwire::marshal::generated::ConsumerIdMarshaller, 1006
- ConsumerInfo
 - activemq::commands::ConsumerInfo, 1011
- ConsumerInfoMarshaller
 - activemq::wireformat::openwire::marshal::generated::ConsumerInfoMarshaller, 1018
- ConsumerState
 - activemq::state::ConsumerState, 1022
- ControlCommand
 - activemq::commands::ControlCommand, 1023
- ControlCommandMarshaller
 - activemq::wireformat::openwire::marshal::generated::ControlCommandMarshaller, 1026
- CopyOnWriteArrayList
 - decaf::util::concurrent::CopyOnWriteArrayList, 1032
- CopyOnWriteArraySet
 - decaf::util::concurrent::CopyOnWriteArrayList, 1049
 - decaf::util::concurrent::CopyOnWriteArraySet, 1053
- CountDownLatch
 - decaf::util::concurrent::CountDownLatch, 1063
- CounterType
 - decaf::lang::ArrayPointer, 464
 - decaf::lang::Pointer, 2085
- DAYS
 - decaf::util::concurrent::TimeUnit, 2764
- DEBUG
 - decaf::util::logging::Level, 1619
- DECAF_API
 - decaf/util/Config.h, 3040
- DECAF_CATCHALL_NOTHROW
 - decaf/lang/exceptions/ExceptionDefines.h, 3011
- DECAF_CATCHALL_THROW
 - decaf/lang/exceptions/ExceptionDefines.h, 3011
- DECAF_CATCH_EXCEPTION_CONVERT
 - decaf/lang/exceptions/ExceptionDefines.h, 3010
- DECAF_CATCH_NOTHROW
 - decaf/lang/exceptions/ExceptionDefines.h, 3010
- DECAF_CATCH_RETHROW
 - decaf/lang/exceptions/ExceptionDefines.h, 3010
- DECAF_UNUSED
 - decaf/util/Config.h, 3040
- DEFAULT_BUFFER_SIZE
 - decaf::util::zip::DeflaterOutputStream, 1193
 - decaf::util::zip::InflaterInputStream, 1464
- DEFAULT_COMPRESSION
 - decaf::util::zip::Deflater, 1188
- DEFAULT_DELIVERY_MODE
 - cms::Message, 1867
- DEFAULT_DURABLE_TOPIC_PREFETCH
 - activemq::core::policies::DefaultPrefetchPolicy, 1149
- DEFAULT_MESSAGE_SIZE
 - activemq::commands::Message, 1837
- DEFAULT_MSG_PRIORITY
 - cms::Message, 1867
- DEFAULT_ORDERED_TARGET
 - activemq::commands::ActiveMQDestination, 256
- DEFAULT_PRIORITY
 - activemq::cmsutil::CmsTemplate, 850
- DEFAULT_QUEUE_BROWSER_PREFETCH
 - activemq::core::policies::DefaultPrefetchPolicy, 1149
- DEFAULT_QUEUE_PREFETCH

- activemq::core::policies::Default-PrefetchPolicy, 1149
- DEFAULT_STRATEGY
 - decaf::util::zip::Deflater, 1189
- DEFAULT_TIME_TO_LIVE
 - activemq::cmsutil::CmsTemplate, 850
 - cms::Message, 1867
- DEFAULT_TOPIC_PREFETCH
 - activemq::core::policies::Default-PrefetchPolicy, 1149
- DEFAULT_URI
 - activemq::core::ActiveMQConnection-Factory, 226
- DEFAULT_VERSION
 - activemq::wireformat::openwire::OpenWireFormat, 2058
- DEFLATED
 - decaf::util::zip::Deflater, 1189
- DEF_MEM_LEVEL
 - zutil.h, 3162
- DELIVERY_MODE
 - cms::DeliveryMode, 1196
- DESTINATION_ADD_OPERATION
 - activemq::core::ActiveMQConstants, 232
- DESTINATION_REMOVE_OPERATION
 - activemq::core::ActiveMQConstants, 232
- DICT
 - inflate.h, 3149
- DICTID
 - inflate.h, 3149
- DISCONNECT
 - activemq::wireformat::stomp::Stomp-CommandConstants, 2583
- DIST
 - inflate.h, 3149
- DISTEXT
 - inflate.h, 3149
- DISTS
 - inftrees.h, 3150
- DONE
 - inflate.h, 3149
- DOUBLE_TYPE
 - activemq::util::PrimitiveValueNode, 2148
- DUPS_OK_ACKNOWLEDGE
 - cms::Session, 2365
- DYN_TREES
 - zutil.h, 3162
- D_CODES
 - deflate.h, 3144
- Dad
 - deflate.h, 3144
- dataArrayResponse
 - activemq::commands::DataArray-Response, 1070
- dataArrayResponseMarshaller
 - activemq::wireformat::openwire::marshal::generated::Data-ArrayResponseMarshaller, 1073
- DataFormatException
 - decaf::util::zip::DataFormatException, 1076, 1077
- DataInputStream
 - decaf::io::DataInputStream, 1096
- DataOutputStream
 - decaf::io::DataOutputStream, 1110
- DataResponse
 - activemq::commands::DataResponse, 1113
- DataResponseMarshaller
 - activemq::wireformat::openwire::marshal::generated::Data-ResponseMarshaller, 1116
- DatagramPacket
 - decaf::net::DatagramPacket, 1080–1082
- Date
 - decaf::util::Date, 1140
- Debug
 - decaf::util::logging, 102
- DecafRuntime
 - decaf::internal::DecafRuntime, 1143
- DedicatedTaskRunner
 - activemq::threads::DedicatedTask-Runner, 1145
- DefaultPrefetchPolicy
 - activemq::core::policies::Default-PrefetchPolicy, 1147
- DefaultRedeliveryPolicy
 - activemq::core::policies::Default-RedeliveryPolicy, 1151
- DefaultSSLContext
 - decaf::internal::net::ssl::DefaultSSL-Context, 1164
- DefaultSSLServerSocketFactory

- decaf::internal::net::ssl::DefaultSSL-
ServerSocketFactory, 1167
- DefaultSSLSocketFactory
 - decaf::internal::net::ssl::DefaultSSL-
SocketFactory, 1173
- DefaultServerSocketFactory
 - decaf::internal::net::DefaultServer-
SocketFactory, 1156
- DefaultSocketFactory
 - decaf::internal::net::DefaultSocket-
Factory, 1161
- Deflater
 - decaf::util::zip::Deflater, 1182
- DeflaterOutputStream
 - decaf::util::zip::DeflaterOutput-
Stream, 1191, 1192
- DestinationActions
 - activemq::core::ActiveMQConstants,
232
- DestinationInfo
 - activemq::commands::Destination-
Info, 1215
- DestinationInfoMarshaller
 - activemq::wireformat::openwire-
::marshal::generated::Destination-
InfoMarshaller, 1219
- DestinationOption
 - activemq::core::ActiveMQConstants,
232
- DestinationType
 - cms::Destination, 1211
- DiscardOldestPolicy
 - decaf::util::concurrent::ThreadPool-
Executor::CallerRunsPolicy::-
DiscardOldestPolicy, 1224
- DiscardPolicy
 - decaf::util::concurrent::ThreadPool-
Executor::CallerRunsPolicy::-
DiscardPolicy, 1226
- DiscoveryEvent
 - activemq::commands::Discovery-
Event, 1227
- DiscoveryEventMarshaller
 - activemq::wireformat::openwire-
::marshal::generated::Discovery-
EventMarshaller, 1231
- DispatchData
 - activemq::core::DispatchData, 1234
- Double
 - decaf::lang::Double, 1238
- DoubleArrayBuffer
 - decaf::internal::nio::DoubleArray-
Buffer, 1252, 1253
- DoubleBuffer
 - decaf::nio::DoubleBuffer, 1261
- DynamicDestinationResolver
 - activemq::cmsutil::DynamicDestination-
Resolver, 1272
- E
 - decaf::lang::Math, 1818
- ENOUGH
 - inftrees.h, 3150
- ENOUGH_DISTS
 - inftrees.h, 3150
- ENOUGH_LENS
 - inftrees.h, 3150
- EOFException
 - decaf::io::EOFException, 1275, 1276
- ERROR_CMD
 - activemq::wireformat::stomp::Stomp-
CommandConstants, 2584
- ERR_MSG
 - zutil.h, 3162
- ERR_RETURN
 - zutil.h, 3162
- EXLEN
 - inflate.h, 3149
- EXTRA
 - inflate.h, 3149
- EXTRA_STATE
 - deflate.h, 3144
- Entry
 - decaf::util::Map::Entry, 1274
- Error
 - decaf::util::logging, 102
- ErrorMessage
 - decaf::util::logging::ErrorMessage,
1278
- Exception
 - decaf::lang::Exception, 1281, 1282
- ExceptionResponse
 - activemq::commands::Exception-
Response, 1288
- ExceptionResponseMarshaller
 - activemq::wireformat::openwire-
::marshal::generated::Exception-
ResponseMarshaller, 1291
- ExecutionException
 - decaf::util::concurrent::Execution-
Exception, 1295, 1296

- ExecutorKernel
 - decaf::util::concurrent::ThreadPool-Executor, 2732
- FILTERED
 - decaf::util::zip::Deflater, 1189
- FINE
 - decaf::util::logging::Level, 1619
- FINER
 - decaf::util::logging::Level, 1620
- FINEST
 - decaf::util::logging::Level, 1620
- FINISH_STATE
 - deflate.h, 3144
- FLAGS
 - inflate.h, 3148
- FLOAT_TYPE
 - activemq::util::PrimitiveValueNode, 2148
- FLUSH_FAILURE
 - decaf::util::logging::ErrorManager, 1279
- FORMAT_FAILURE
 - decaf::util::logging::ErrorManager, 1279
- FailoverTransport
 - activemq::transport::failover::Failover-Transport, 1308
- FailoverTransportListener
 - activemq::transport::failover::Failover-Transport, 1318
 - activemq::transport::failover::Failover-TransportListener, 1321
- Fatal
 - decaf::util::logging, 102
- FifoMessageDispatchChannel
 - activemq::core::FifoMessageDispatch-Channel, 1324
- FileDescriptor
 - decaf::io::FileDescriptor, 1331
- FileName
 - activemq::commands::BrokerError::StackTraceElement, 2528
- FilterInputStream
 - decaf::io::FilterInputStream, 1336
- FilterOutputStream
 - decaf::io::FilterOutputStream, 1342
- Float
 - decaf::lang::Float, 1346, 1347
- FloatArrayBuffer
 - decaf::internal::nio::FloatArrayBuffer, 1360–1362
- FloatBuffer
 - decaf::nio::FloatBuffer, 1370
- FlushCommand
 - activemq::commands::FlushCommand, 1381
- FlushCommandMarshaller
 - activemq::wireformat::openwire-::marshal::generated::Flush-CommandMarshaller, 1384
- Freq
 - deflate.h, 3144
- FutureResponse
 - activemq::transport::correlator::FutureResponse, 1393
- GENERIC_FAILURE
 - decaf::util::logging::ErrorManager, 1279
- GT_OFF
 - gzguts.h, 3147
- GUNZIP
 - inflate.h, 3148
- GZBUFSIZE
 - gzguts.h, 3147
- GZIP
 - deflate.h, 3144
 - gzguts.h, 3147
- GZ_APPEND
 - gzguts.h, 3147
- GZ_NONE
 - gzguts.h, 3147
- GZ_READ
 - gzguts.h, 3147
- GZ_WRITE
 - gzguts.h, 3147
- GeneralSecurityException
 - decaf::security::GeneralSecurity-Exception, 1395, 1396
- GenericResource
 - decaf::internal::util::GenericResource, 1398
- HAVE_PTHREAD_H
 - activemq/util/Config.h, 3039
 - decaf/util/Config.h, 3040
- HAVE_UUID_T
 - activemq/util/Config.h, 3039
 - decaf/util/Config.h, 3040
- HAVE_UUID_UUID_H
 - activemq/util/Config.h, 3039

- decaf/util/Config.h, 3040
- HCRC
 - inflate.h, 3149
- HCRC_STATE
 - deflate.h, 3144
- HEAD
 - inflate.h, 3148
- HEADER_ACK
 - activemq::wireformat::stomp::Stomp-CommandConstants, 2584
- HEADER_CLIENT_ID
 - activemq::wireformat::stomp::Stomp-CommandConstants, 2584
- HEADER_CONSUMERPRIORITY
 - activemq::wireformat::stomp::Stomp-CommandConstants, 2584
- HEADER_CONTENTLENGTH
 - activemq::wireformat::stomp::Stomp-CommandConstants, 2584
- HEADER_CORRELATIONID
 - activemq::wireformat::stomp::Stomp-CommandConstants, 2584
- HEADER_DESTINATION
 - activemq::wireformat::stomp::Stomp-CommandConstants, 2584
- HEADER_DISPATCH_ASYNC
 - activemq::wireformat::stomp::Stomp-CommandConstants, 2584
- HEADER_EXCLUSIVE
 - activemq::wireformat::stomp::Stomp-CommandConstants, 2584
- HEADER_EXPIRES
 - activemq::wireformat::stomp::Stomp-CommandConstants, 2584
- HEADER_ID
 - activemq::wireformat::stomp::Stomp-CommandConstants, 2584
- HEADER_JMSPRIORITY
 - activemq::wireformat::stomp::Stomp-CommandConstants, 2584
- HEADER_LOGIN
 - activemq::wireformat::stomp::Stomp-CommandConstants, 2584
- HEADER_MAXPENDINGMSGLIMIT
 - activemq::wireformat::stomp::Stomp-CommandConstants, 2585
- HEADER_MESSAGE
 - activemq::wireformat::stomp::Stomp-CommandConstants, 2585
- HEADER_MESSAGEID
 - activemq::wireformat::stomp::Stomp-CommandConstants, 2585
- HEADER_NOLOCAL
 - activemq::wireformat::stomp::Stomp-CommandConstants, 2585
- HEADER_OLDSUBSCRIPTIONNAME
 - activemq::wireformat::stomp::Stomp-CommandConstants, 2585
- HEADER_PASSWORD
 - activemq::wireformat::stomp::Stomp-CommandConstants, 2585
- HEADER_PERSISTENT
 - activemq::wireformat::stomp::Stomp-CommandConstants, 2585
- HEADER_PREFETCHSIZE
 - activemq::wireformat::stomp::Stomp-CommandConstants, 2585
- HEADER_RECEIPTID
 - activemq::wireformat::stomp::Stomp-CommandConstants, 2585
- HEADER_RECEIPT_REQUIRED
 - activemq::wireformat::stomp::Stomp-CommandConstants, 2585
- HEADER_REDELIVERED
 - activemq::wireformat::stomp::Stomp-CommandConstants, 2585
- HEADER_REDELIVERYCOUNT
 - activemq::wireformat::stomp::Stomp-CommandConstants, 2585
- HEADER_REPLYTO
 - activemq::wireformat::stomp::Stomp-CommandConstants, 2586
- HEADER_REQUESTID
 - activemq::wireformat::stomp::Stomp-CommandConstants, 2586
- HEADER_RESPONSEID
 - activemq::wireformat::stomp::Stomp-CommandConstants, 2586
- HEADER_RETROACTIVE
 - activemq::wireformat::stomp::Stomp-CommandConstants, 2586
- HEADER_SELECTOR
 - activemq::wireformat::stomp::Stomp-CommandConstants, 2586
- HEADER_SESSIONID
 - activemq::wireformat::stomp::Stomp-CommandConstants, 2586
- HEADER_SUBSCRIPTION
 - activemq::wireformat::stomp::Stomp-CommandConstants, 2586

- HEADER_SUBSCRIPTIONNAME
 - activemq::wireformat::stomp::Stomp-CommandConstants, 2586
- HEADER_TIMESTAMP
 - activemq::wireformat::stomp::Stomp-CommandConstants, 2586
- HEADER_TRANSACTIONID
 - activemq::wireformat::stomp::Stomp-CommandConstants, 2586
- HEADER_TRANSFORMATION
 - activemq::wireformat::stomp::Stomp-CommandConstants, 2586
- HEADER_TRANSFORMATION_ERROR
 - activemq::wireformat::stomp::Stomp-CommandConstants, 2586
- HEADER_TYPE
 - activemq::wireformat::stomp::Stomp-CommandConstants, 2587
- HEAP_SIZE
 - deflate.h, 3145
- HOURS
 - decaf::util::concurrent::TimeUnit, 2764
- HUFFMAN_ONLY
 - decaf::util::zip::Deflater, 1189
- Handler
 - decaf::util::logging::Handler, 1403
- HexStringParser
 - decaf::internal::util::HexStringParser, 1406
- HexTable
 - activemq::wireformat::openwire::utils::HexTable, 1408
- HttpRetryException
 - decaf::net::HttpRetryException, 1409, 1410
- ID_ACTIVEMQBLOBBMESSAGE
 - activemq::commands::ActiveMQ-BlobMessage, 161
- ID_ACTIVEMQBYTESMESSAGE
 - activemq::commands::ActiveMQ-BytesMessage, 183
- ID_ACTIVEMQDESTINATION
 - activemq::commands::ActiveMQ-Destination, 256
- ID_ACTIVEMQMAPMESSAGE
 - activemq::commands::ActiveMQ-MapMessage, 283
- ID_ACTIVEMQMESSAGE
 - activemq::commands::ActiveMQ-Message, 290
- ID_ACTIVEMQOBJECTMESSAGE
 - activemq::commands::ActiveMQ-ObjectMessage, 303
- ID_ACTIVEMQQUEUE
 - activemq::commands::ActiveMQ-Queue, 325
- ID_ACTIVEMQSTREAMMESSAGE
 - activemq::commands::ActiveMQ-StreamMessage, 374
- ID_ACTIVEMQTEMPDESTINATION
 - activemq::commands::ActiveMQ-TempDestination, 382
- ID_ACTIVEMQTEMPQUEUE
 - activemq::commands::ActiveMQ-TempQueue, 391
- ID_ACTIVEMQTEMPTOPIC
 - activemq::commands::ActiveMQ-TempTopic, 400
- ID_ACTIVEMQTEXTMESSAGE
 - activemq::commands::ActiveMQ-TextMessage, 410
- ID_ACTIVEMQTOPIC
 - activemq::commands::ActiveMQ-Topic, 418
- ID_BROKERID
 - activemq::commands::BrokerId, 567
- ID_BROKERINFO
 - activemq::commands::BrokerInfo, 578
- ID_CONNECTIONCONTROL
 - activemq::commands::Connection-Control, 943
- ID_CONNECTIONERROR
 - activemq::commands::Connection-Error, 951
- ID_CONNECTIONID
 - activemq::commands::ConnectionId, 963
- ID_CONNECTIONINFO
 - activemq::commands::Connection-Info, 973
- ID_CONSUMERCONTROL
 - activemq::commands::Consumer-Control, 996
- ID_CONSUMERID
 - activemq::commands::ConsumerId, 1004
- ID_CONSUMERINFO

- activemq::commands::Consumer-Info, 1017
- ID_CONTROLCOMMAND
 - activemq::commands::Control-Command, 1025
- ID_DATAARRAYRESPONSE
 - activemq::commands::DataArray-Response, 1071
- ID_DATARESPONSE
 - activemq::commands::DataResponse, 1115
- ID_DESTINATIONINFO
 - activemq::commands::Destination-Info, 1217
- ID_DISCOVERYEVENT
 - activemq::commands::Discovery-Event, 1229
- ID_EXCEPTIONRESPONSE
 - activemq::commands::Exception-Response, 1290
- ID_FLUSHCOMMAND
 - activemq::commands::FlushCommand, 1383
- ID_INTEGERRESPONSE
 - activemq::commands::Integer-Response, 1519
- ID_JOURNALQUEUEACK
 - activemq::commands::Journal-QueueAck, 1563
- ID_JOURNALTOPICACK
 - activemq::commands::JournalTopic-Ack, 1572
- ID_JOURNALTRACE
 - activemq::commands::JournalTrace, 1579
- ID_JOURNALTRANSACTION
 - activemq::commands::Journal-Transaction, 1586
- ID_KEEPAALIVEINFO
 - activemq::commands::KeepAliveInfo, 1594
- ID_LASTPARTIALCOMMAND
 - activemq::commands::LastPartial-Command, 1608
- ID_LOCALTRANSACTIONID
 - activemq::commands::LocalTransaction-Id, 1678
- ID_MESSAGE
 - activemq::commands::Message, 1838
- ID_MESSAGEACK
 - activemq::commands::MessageAck, 1872
- ID_MESSAGEDISPATCH
 - activemq::commands::Message-Dispatch, 1885
- ID_MESSAGEDISPATCHNOTIFICATION
 - activemq::commands::Message-DispatchNotification, 1899
- ID_MESSAGEID
 - activemq::commands::MessageId, 1912
- ID_MESSAGEPULL
 - activemq::commands::MessagePull, 1943
- ID_NETWORKBRIDGEFILTER
 - activemq::commands::Network-BridgeFilter, 1974
- ID_PARTIALCOMMAND
 - activemq::commands::Partial-Command, 2079
- ID_PRODUCERACK
 - activemq::commands::ProducerAck, 2174
- ID_PRODUCERID
 - activemq::commands::ProducerId, 2185
- ID_PRODUCERINFO
 - activemq::commands::ProducerInfo, 2194
- ID_REMOVEINFO
 - activemq::commands::RemoveInfo, 2271
- ID_REMOVESUBSCRIPTIONINFO
 - activemq::commands::Remove-SubscriptionInfo, 2279
- ID_REPLAYCOMMAND
 - activemq::commands::Replay-Command, 2287
- ID_RESPONSE
 - activemq::commands::Response, 2301
- ID_SESSIONID
 - activemq::commands::SessionId, 2382
- ID_SESSIONINFO
 - activemq::commands::SessionInfo, 2389
- ID_SHUTDOWNINFO

- activemq::commands::ShutdownInfo, 2434
- ID_SUBSCRIPTIONINFO
 - activemq::commands::SubscriptionInfo, 2634
- ID_TRANSACTIONID
 - activemq::commands::TransactionId, 2770
- ID_TRANSACTIONINFO
 - activemq::commands::TransactionInfo, 2778
- ID_WIREFORMATINFO
 - activemq::commands::WireFormatInfo, 2899
- ID_XATransactionID
 - activemq::commands::XATransactionId, 2942
- INDIVIDUAL_ACKNOWLEDGE
 - cms::Session, 2365
- INFO
 - decaf::util::logging::Level, 1620
- INHERIT
 - decaf::util::logging::Level, 1620
- INIT_STATE
 - deflate.h, 3145
- INTEGER_TYPE
 - activemq::util::PrimitiveValueNode, 2148
- IOException
 - decaf::io::IOException, 1546, 1547
- IOTransport
 - activemq::transport::IOTransport, 1550
- IPos
 - deflate.h, 3145
- IdGenerator
 - activemq::util::IdGenerator, 1412
- IllegalArgumentException
 - decaf::lang::exceptions::IllegalArgumentException, 1414, 1415
- IllegalMonitorStateException
 - decaf::lang::exceptions::IllegalMonitorStateException, 1417, 1418
- IllegalStateException
 - cms::IllegalStateException, 1419, 1420
 - decaf::lang::exceptions::IllegalStateException, 1421, 1422
- IllegalThreadStateException
 - decaf::lang::exceptions::IllegalThreadStateException, 1423, 1424
- InactivityMonitor
 - activemq::transport::inactivity::InactivityMonitor, 1426
- IndexOutOfBoundsException
 - decaf::lang::exceptions::IndexOutOfBoundsException, 1429, 1430
- Inet4Address
 - decaf::net::Inet4Address, 1432
- Inet6Address
 - decaf::net::Inet6Address, 1436
- InetAddress
 - decaf::net::Inet4Address, 1435
 - decaf::net::Inet6Address, 1437
 - decaf::net::InetAddress, 1439
- InetSocketAddress
 - decaf::net::InetSocketAddress, 1445
- Inflater
 - decaf::util::zip::Inflater, 1449
- InflaterInputStream
 - decaf::util::zip::InflaterInputStream, 1459
- Info
 - decaf::util::logging, 102
- InputStream
 - decaf::io::InputStream, 1466
- InputStreamReader
 - decaf::io::InputStreamReader, 1476
- IntArrayBuffer
 - decaf::internal::nio::IntArrayBuffer, 1481, 1482
- IntBuffer
 - decaf::nio::IntBuffer, 1490
- Integer
 - decaf::lang::Integer, 1502
- IntegerResponse
 - activemq::commands::IntegerResponse, 1517
- IntegerResponseMarshaller
 - activemq::wireformat::openwire::marshal::generated::IntegerResponseMarshaller, 1520
- InternalCommandListener
 - activemq::transport::mock::InternalCommandListener, 1528
- InterruptedException

- decaf::lang::exceptions::Interrupted-Exception, 1529, 1530
- InterruptedIOException
 - decaf::io::InterruptedIOException, 1532, 1533
- InvalidClientIdException
 - cms::InvalidClientIdException, 1534, 1535
- InvalidDestinationException
 - cms::InvalidDestinationException, 1536
- InvalidKeyException
 - decaf::security::InvalidKeyException, 1537, 1538
- InvalidMarkException
 - decaf::nio::InvalidMarkException, 1539, 1540
- InvalidSelectorException
 - cms::InvalidSelectorException, 1542
- InvalidStateException
 - decaf::lang::exceptions::InvalidStateException, 1543, 1544
- JournalQueueAck
 - activemq::commands::JournalQueueAck, 1562
- JournalQueueAckMarshaller
 - activemq::wireformat::openwire-::marshal::generated::JournalQueueAckMarshaller, 1565
- JournalTopicAck
 - activemq::commands::JournalTopicAck, 1569
- JournalTopicAckMarshaller
 - activemq::wireformat::openwire-::marshal::generated::JournalTopicAckMarshaller, 1573
- JournalTrace
 - activemq::commands::JournalTrace, 1577
- JournalTraceMarshaller
 - activemq::wireformat::openwire-::marshal::generated::JournalTraceMarshaller, 1580
- JournalTransaction
 - activemq::commands::JournalTransaction, 1584
- JournalTransactionMarshaller
 - activemq::wireformat::openwire-::marshal::generated::JournalTransactionMarshaller, 1588
- KeepAliveInfo
 - activemq::commands::KeepAliveInfo, 1592
- KeepAliveInfoMarshaller
 - activemq::wireformat::openwire-::marshal::generated::KeepAliveInfoMarshaller, 1595
- KeyException
 - decaf::security::KeyException, 1601, 1602
- KeyManagementException
 - decaf::security::KeyManagementException, 1604, 1605
- LEN
 - inflate.h, 3149
- LENEXT
 - inflate.h, 3149
- LENGTH
 - inflate.h, 3149
- LENGTH_CODES
 - deflate.h, 3145
- LENLENS
 - inflate.h, 3149
- LENS
 - inftrees.h, 3150
- LEN_
 - inflate.h, 3149
- LIST_TYPE
 - activemq::util::PrimitiveValueNode, 2148
- LIT
 - inflate.h, 3149
- LITERALS
 - deflate.h, 3145
- LOGDECAF_DEBUG
 - LoggerDefines.h, 3255
- LOGDECAF_DEBUG_1
 - LoggerDefines.h, 3255
- LOGDECAF_DECLARE
 - LoggerDefines.h, 3255
- LOGDECAF_DECLARE_LOCAL
 - LoggerDefines.h, 3255
- LOGDECAF_ERROR
 - LoggerDefines.h, 3255
- LOGDECAF_FATAL
 - LoggerDefines.h, 3255
- LOGDECAF_INFO
 - LoggerDefines.h, 3255
- LOGDECAF_INITIALIZE
 - LoggerDefines.h, 3255

- LOGDECAF_WARN
 - LoggerDefines.h, 3255
- LONG_TYPE
 - activemq::util::PrimitiveValueNode, 2148
- LOOK
 - gzguts.h, 3147
- L_CODES
 - deflate.h, 3145
- LastPartialCommand
 - activemq::commands::LastPartialCommand, 1606
- LastPartialCommandMarshaller
 - activemq::wireformat::openwire::marshal::generated::LastPartialCommandMarshaller, 1609
- Len
 - deflate.h, 3145
- Less
 - decaf::util::comparators::Less, 1613
- Level
 - decaf::util::logging::Level, 1617
- Levels
 - decaf::util::logging, 102
- LineNumber
 - activemq::commands::BrokerError::StackTraceElement, 2528
- LinkedBlockingQueue
 - decaf::util::concurrent::LinkedBlockingQueue, 1623, 1624
- LinkedList
 - decaf::util::LinkedList, 1639
- List
 - decaf::util::List, 1660
- LocalTransactionId
 - activemq::commands::LocalTransactionMAP_TYPEId, 1676
- LocalTransactionIdMarshaller
 - activemq::wireformat::openwire::marshal::generated::LocalTransactionIdMarshaller, 1679
- Lock
 - decaf::util::concurrent::Lock, 1683
- LogManager
 - decaf::util::logging::LogManager, 1715
- LogRecord
 - decaf::util::logging::LogRecord, 1720
- LogWriter
 - decaf::util::logging::LogWriter, 1725
- Logger
 - decaf::util::logging::Logger, 1696
- LoggerDefines.h
 - LOGDECAF_DEBUG, 3255
 - LOGDECAF_DEBUG_1, 3255
 - LOGDECAF_DECLARE, 3255
 - LOGDECAF_DECLARE_LOCAL, 3255
 - LOGDECAF_ERROR, 3255
 - LOGDECAF_FATAL, 3255
 - LOGDECAF_INFO, 3255
 - LOGDECAF_INITIALIZE, 3255
 - LOGDECAF_WARN, 3255
- LoggerHierarchy
 - decaf::util::logging::LoggerHierarchy, 1706
- LoggingInputStream
 - activemq::io::LoggingInputStream, 1707
- LoggingOutputStream
 - activemq::io::LoggingOutputStream, 1708
- LoggingTransport
 - activemq::transport::logging::LoggingTransport, 1710
- Long
 - decaf::lang::Long, 1728
- LongArrayBuffer
 - decaf::internal::nio::LongArrayBuffer, 1745–1747
- LongBuffer
 - decaf::nio::LongBuffer, 1755
- LongSequenceGenerator
 - activemq::util::LongSequenceGenerator, 1765
- activemq::util::PrimitiveValueNode, 2148
- MATCH
 - inflate.h, 3149
- MAXBQUALSIZE
 - cms::Xid, 2949
- MAXGTRIDSIZE
 - cms::Xid, 2949
- MAX_BITS
 - deflate.h, 3145
- MAX_DIST
 - deflate.h, 3145
- MAX_MATCH

- zutil.h, 3162
- MAX_PREFETCH_SIZE
 - activemq::core::policies::Default-PrefetchPolicy, 1150
- MAX_PRIORITY
 - decaf::lang::Thread, 2713
- MAX_RADIX
 - decaf::lang::Character, 772
- MAX_SUPPORTED_VERSION
 - activemq::wireformat::openwire::OpenWireFormat, 2058
- MAX_VALUE
 - decaf::lang::Byte, 623
 - decaf::lang::Character, 772
 - decaf::lang::Double, 1248
 - decaf::lang::Float, 1356
 - decaf::lang::Integer, 1516
 - decaf::lang::Long, 1741
 - decaf::lang::Short, 2408
- MEM
 - inflate.h, 3149
- MESSAGE
 - activemq::wireformat::stomp::Stomp-CommandConstants, 2587
- MICROSECONDS
 - decaf::util::concurrent::TimeUnit, 2764
- MILLISECONDS
 - decaf::util::concurrent::TimeUnit, 2764
- MINUTES
 - decaf::util::concurrent::TimeUnit, 2764
- MIN_LOOKAHEAD
 - deflate.h, 3145
- MIN_MATCH
 - zutil.h, 3162
- MIN_PRIORITY
 - decaf::lang::Thread, 2713
- MIN_RADIX
 - decaf::lang::Character, 772
- MIN_VALUE
 - decaf::lang::Byte, 623
 - decaf::lang::Character, 773
 - decaf::lang::Double, 1248
 - decaf::lang::Float, 1356
 - decaf::lang::Integer, 1516
 - decaf::lang::Long, 1741
 - decaf::lang::Short, 2408
- MalformedURLException
 - decaf::net::MalformedURLException, 1766, 1767
- Map
 - decaf::util::Map, 1770
- MarkBlockLogger
 - decaf::util::logging::MarkBlock-Logger, 1792
- Markblock
 - decaf::util::logging, 102
- MarshallingSupport
 - activemq::util::MarshallingSupport, 1797
- Math
 - decaf::lang::Math, 1802
- MemoryUsage
 - activemq::util::MemoryUsage, 1819
- Message
 - activemq::commands::Message, 1826
- MessageAck
 - activemq::commands::MessageAck, 1869
- MessageAckMarshaller
 - activemq::wireformat::openwire::marshal::generated::Message-AckMarshaller, 1874
- MessageDispatch
 - activemq::commands::Message-Dispatch, 1883
- MessageDispatchMarshaller
 - activemq::wireformat::openwire::marshal::generated::Message-DispatchMarshaller, 1891
- MessageDispatchNotification
 - activemq::commands::Message-DispatchNotification, 1896
- MessageDispatchNotificationMarshaller
 - activemq::wireformat::openwire::marshal::generated::Message-DispatchNotificationMarshaller, 1900
- MessageEOFException
 - cms::MessageEOFException, 1906
- MessageFormatException
 - cms::MessageFormatException, 1907
- MessageId
 - activemq::commands::MessageId, 1909
- MessageIdMarshaller

- activemq::wireformat::openwire-
::marshal::generated::Message-
IdMarshaller, 1913
- MessageMarshaller
 - activemq::wireformat::openwire-
::marshal::generated::Message-
Marshaller, 1918
- MessageNotReadableException
 - cms::MessageNotReadableException,
1923
- MessageNotWriteableException
 - cms::MessageNotWriteableException,
1924
- MessagePropertyInterceptor
 - activemq::wireformat::openwire-
::utils::MessageProperty-
Interceptor, 1934
- MessagePull
 - activemq::commands::MessagePull,
1941
- MessagePullMarshaller
 - activemq::wireformat::openwire-
::marshal::generated::Message-
PullMarshaller, 1945
- MethodName
 - activemq::commands::BrokerError::-
StackTraceElement, 2528
- MockTransport
 - activemq::transport::mock::Mock-
Transport, 1950
- Mutex
 - decaf::util::concurrent::Mutex, 1961
- MutexHandle
 - decaf::util::concurrent::MutexHandle,
1966
- NAME
 - inflate.h, 3149
- NAME_STATE
 - deflate.h, 3145
- NANOSECONDS
 - decaf::util::concurrent::TimeUnit,
2764
- NEGATIVE_INFINITY
 - decaf::lang::Double, 1248
 - decaf::lang::Float, 1356
- NEW
 - decaf::lang::Thread, 2707
- NON_PERSISTENT
 - cms::DeliveryMode, 1196
- NORM_PRIORITY
 - decaf::lang::Thread, 2713
- NO_COMPRESSION
 - decaf::util::zip::Deflater, 1189
- NO_MAXIMUM_REDELIVERIES
 - activemq::core::RedeliveryPolicy,
2256
- NULL_TYPE
 - activemq::wireformat::openwire::-
OpenWireFormat, 2059
 - activemq::util::PrimitiveValueNode,
2148
- NUM_OPTIONS
 - activemq::core::ActiveMQConstants,
232
- NUM_PARAMS
 - activemq::core::ActiveMQConstants,
233
- NaN
 - decaf::lang::Double, 1248
 - decaf::lang::Float, 1356
- Network
 - decaf::internal::net::Network, 1969
- NetworkBridgeFilter
 - activemq::commands::Network-
BridgeFilter, 1972
- NetworkBridgeFilterMarshaller
 - activemq::wireformat::openwire-
::marshal::generated::Network-
BridgeFilterMarshaller, 1975
- NoRouteToHostException
 - decaf::net::NoRouteToHostException,
1979, 1980
- NoSuchAlgorithmException
 - decaf::security::NoSuchAlgorithm-
Exception, 1982, 1983
- NoSuchElementException
 - decaf::util::NoSuchElementException,
1984, 1985
- NoSuchProviderException
 - decaf::security::NoSuchProvider-
Exception, 1987, 1988
- Null
 - decaf::util::logging, 102
- NullPointerException
 - decaf::lang::exceptions::NullPointerException,
1990, 1991
- NumberFormatException
 - decaf::lang::exceptions::Number-
FormatException, 1995, 1996
- OF

- deflate.h, 3145, 3146
- gzguts.h, 3147, 3148
- inffast.h, 3148
- infrees.h, 3150
- zlib.h, 3158–3161
- zutil.h, 3163
- OFF
 - decaf::util::logging::Level, 1620
- OPEN_FAILURE
 - decaf::util::logging::ErrorManager, 1279
- OS
 - inflate.h, 3149
- Off
 - decaf::util::logging, 102
- OpenSSLContextSpi
 - decaf::internal::net::ssl::openssl::OpenSSLContextSpi, 1999
- OpenSSLServerSocket
 - decaf::internal::net::ssl::openssl::OpenSSLServerSocket, 2005
- OpenSSLServerSocketFactory
 - decaf::internal::net::ssl::openssl::OpenSSLServerSocketFactory, 2011
- OpenSSLSocket
 - decaf::internal::net::ssl::openssl::OpenSSLContextSpi, 2001
 - decaf::internal::net::ssl::openssl::OpenSSLSocket, 2019, 2020
- OpenSSLSocketException
 - decaf::internal::net::ssl::openssl::OpenSSLSocketException, 2030, 2031
- OpenSSLSocketFactory
 - decaf::internal::net::ssl::openssl::OpenSSLContextSpi, 2001
 - decaf::internal::net::ssl::openssl::OpenSSLSocketFactory, 2036
- OpenSSLSocketInputStream
 - decaf::internal::net::ssl::openssl::OpenSSLSocketInputStream, 2041
- OpenSSLSocketOutputStream
 - decaf::internal::net::ssl::openssl::OpenSSLSocketOutputStream, 2044
- OpenWireFormat
 - activemq::wireformat::openwire::OpenWireFormat, 2048
- OpenWireFormatFactory
 - activemq::wireformat::openwire::OpenWireFormatFactory, 2059
- OpenWireFormatNegotiator
 - activemq::wireformat::openwire::OpenWireFormatNegotiator, 2061
- OpenWireResponseBuilder
 - activemq::wireformat::openwire::OpenWireResponseBuilder, 2065
- OutputStream
 - decaf::io::OutputStream, 2068
- OutputStreamWriter
 - decaf::io::OutputStreamWriter, 2075
- PARAM_CLIENTID
 - activemq::core::ActiveMQConstants, 233
- PARAM_PASSWORD
 - activemq::core::ActiveMQConstants, 233
- PARAM_USERNAME
 - activemq::core::ActiveMQConstants, 233
- PERSISTENT
 - cms::DeliveryMode, 1196
- PI
 - decaf::lang::Math, 1818
- POSITIVE_INFINITY
 - decaf::lang::Double, 1248
 - decaf::lang::Float, 1356
- PRESET_DICT
 - zutil.h, 3162
- PRODUCER_ADVISORY_PREFIX
 - activemq::commands::ActiveMQ-Destination, 257
- PartialCommand
 - activemq::commands::Partial-Command, 2077
- PartialCommandMarshaller
 - activemq::wireformat::openwire::marshal::generated::Partial-CommandMarshaller, 2080
- Pointer
 - decaf::lang::Pointer, 2086, 2087
- PointerType
 - decaf::lang::ArrayPointer, 464
 - decaf::lang::Pointer, 2085
- PooledSession

- activemq::cmsutil::PooledSession, 2095
- PortUnreachableException
 - decaf::net::PortUnreachableException, 2108, 2109
- Pos
 - deflate.h, 3145
- Posf
 - deflate.h, 3145
- PrefetchPolicy
 - activemq::core::PrefetchPolicy, 2111
- PrimitiveList
 - activemq::util::PrimitiveList, 2116
- PrimitiveMap
 - activemq::util::PrimitiveMap, 2127
- PrimitiveType
 - activemq::util::PrimitiveValueNode, 2148
- PrimitiveTypesMarshaller
 - activemq::wireformat::openwire::marshal::PrimitiveTypesMarshaller, 2137
- PrimitiveValueConverter
 - activemq::util::PrimitiveValueConverter, 2144
- PrimitiveValueNode
 - activemq::util::PrimitiveValueNode, 2149–2151
- PriorityQueue
 - decaf::util::PriorityQueue, 2164, 2165
- PriorityQueueIterator
 - decaf::util::PriorityQueue, 2171
- ProducerAck
 - activemq::commands::ProducerAck, 2172
- ProducerAckMarshaller
 - activemq::wireformat::openwire::marshal::generated::ProducerAckMarshaller, 2176
- ProducerExecutor
 - activemq::cmsutil::CmsTemplate, 850
 - activemq::cmsutil::CmsTemplate::ProducerExecutor, 2180
- ProducerId
 - activemq::commands::ProducerId, 2183
- ProducerIdMarshaller
 - activemq::wireformat::openwire::marshal::generated::ProducerIdMarshaller, 2187
- activemq::commands::ProducerInfo, 2191
- ProducerInfoMarshaller
 - activemq::wireformat::openwire::marshal::generated::ProducerInfoMarshaller, 2196
- ProducerState
 - activemq::state::ProducerState, 2199
- Properties
 - decaf::util::Properties, 2201
- ProtocolException
 - decaf::net::ProtocolException, 2212, 2213
- PushbackInputStream
 - decaf::io::PushbackInputStream, 2216
- QUEUE
 - cms::Destination, 1211
- QUEUE_PREFIX
 - activemq::wireformat::stomp::StompCommandConstants, 2587
- QUEUE_QUALIFIED_PREFIX
 - activemq::commands::ActiveMQDestination, 257
- RECEIPT
 - activemq::wireformat::stomp::StompCommandConstants, 2587
- RECEIVE_TIMEOUT_INDEFINITE_WAIT
 - activemq::cmsutil::CmsTemplate, 850
- RECEIVE_TIMEOUT_NO_WAIT
 - activemq::cmsutil::CmsTemplate, 850
- RUNNABLE
 - decaf::lang::Thread, 2707
- Random
 - decaf::util::Random, 2230, 2231
- ReadChecker
 - activemq::transport::inactivity::InactivityMonitor, 1428
 - activemq::transport::inactivity::ReadChecker, 2237
- ReadOnlyBufferException
 - decaf::nio::ReadOnlyBufferException, 2244, 2245

- Reader
 - decaf::io::Reader, 2239
- ReceiveExecutor
 - activemq::cmsutil::CmsTemplate, 850
 - activemq::cmsutil::CmsTemplate::ReceiveExecutor, 2249
- RedeliveryPolicy
 - activemq::core::RedeliveryPolicy, 2252
- ReentrantLock
 - decaf::util::concurrent::locks::ReentrantLock, 2258
- ReferenceType
 - decaf::lang::ArrayPointer, 464
 - decaf::lang::Pointer, 2086
- RejectedExecutionException
 - decaf::util::concurrent::RejectedExecutionException, 2264, 2265
- RejectedExecutionHandler
 - decaf::util::concurrent::RejectedExecutionHandler, 2267
- RemoveInfo
 - activemq::commands::RemoveInfo, 2268
- RemoveInfoMarshaller
 - activemq::wireformat::openwire::marshal::generated::RemoveInfoMarshaller, 2272
- RemoveSubscriptionInfo
 - activemq::commands::RemoveSubscriptionInfo, 2276
- RemoveSubscriptionInfoMarshaller
 - activemq::wireformat::openwire::marshal::generated::RemoveSubscriptionInfoMarshaller, 2281
- RemoveTransactionAction
 - activemq::state::ConnectionStateTracker, 990
- ReplayCommand
 - activemq::commands::ReplayCommand, 2285
- ReplayCommandMarshaller
 - activemq::wireformat::openwire::marshal::generated::ReplayCommandMarshaller, 2288
- ResolveProducerExecutor
 - activemq::cmsutil::CmsTemplate, 850
 - activemq::cmsutil::CmsTemplate::ResolveProducerExecutor, 2291
- ResolveReceiveExecutor
 - activemq::cmsutil::CmsTemplate, 850
 - activemq::cmsutil::CmsTemplate::ResolveReceiveExecutor, 2292
- ResourceLifecycleManager
 - activemq::cmsutil::ResourceLifecycleManager, 2294
 - decaf::internal::util::ResourceLifecycleManager, 2297
- Response
 - activemq::commands::Response, 2299
- ResponseCorrelator
 - activemq::transport::correlator::ResponseCorrelator, 2304
- ResponseMarshaller
 - activemq::wireformat::openwire::marshal::generated::ResponseMarshaller, 2308
- RuntimeException
 - decaf::lang::exceptions::RuntimeException, 2315, 2316
- SECONDS
 - decaf::util::concurrent::TimeUnit, 2764
- SEND
 - activemq::wireformat::stomp::StompCommandConstants, 2587
- SERVICESUPPORT_H_
 - ServiceSupport.h, 3046
- SESSION_TRANSACTED
 - cms::Session, 2365
- SEVERE
 - decaf::util::logging::Level, 1620
- SHORT_TYPE
 - activemq::util::PrimitiveValueNode, 2148
- SIZE
 - decaf::lang::Byte, 623
 - decaf::lang::Character, 773
 - decaf::lang::Double, 1248
 - decaf::lang::Float, 1357
 - decaf::lang::Integer, 1516
 - decaf::lang::Long, 1741

- decaf::lang::Short, 2408
- SLEEPING
 - decaf::lang::Thread, 2707
- SOCKET_OPTION_BINDADDR
 - decaf::net::SocketOptions, 2490
- SOCKET_OPTION_BROADCAST
 - decaf::net::SocketOptions, 2490
- SOCKET_OPTION_IP_MULTICAST_IF
 - decaf::net::SocketOptions, 2490
- SOCKET_OPTION_IP_MULTICAST_IF2
 - decaf::net::SocketOptions, 2490
- SOCKET_OPTION_IP_MULTICAST_LOOP
 - decaf::net::SocketOptions, 2491
- SOCKET_OPTION_IP_TOS
 - decaf::net::SocketOptions, 2491
- SOCKET_OPTION_KEEPAIVE
 - decaf::net::SocketOptions, 2491
- SOCKET_OPTION_LINGER
 - decaf::net::SocketOptions, 2491
- SOCKET_OPTION_OOINLINE
 - decaf::net::SocketOptions, 2491
- SOCKET_OPTION_RCVBUF
 - decaf::net::SocketOptions, 2492
- SOCKET_OPTION_REUSEADDR
 - decaf::net::SocketOptions, 2492
- SOCKET_OPTION_SNDBUF
 - decaf::net::SocketOptions, 2492
- SOCKET_OPTION_TCP_NODELAY
 - decaf::net::SocketOptions, 2492
- SOCKET_OPTION_TIMEOUT
 - decaf::net::SocketOptions, 2492
- SSLContext
 - decaf::net::ssl::SSLContext, 2496
- SSLParameters
 - decaf::net::ssl::SSLParameters, 2502
- SSLServerSocket
 - decaf::net::ssl::SSLServerSocket, 2506, 2507
- SSLServerSocketFactory
 - decaf::net::ssl::SSLServerSocketFactory, 2511
- SSLSocket
 - decaf::net::ssl::SSLSocket, 2515, 2516
- SSLSocketFactory
 - decaf::net::ssl::SSLSocketFactory, 2523
- STATIC_TREES
 - zutil.h, 3162
- STORED
 - inflate.h, 3149
- STORED_BLOCK
 - zutil.h, 3162
- STRING_TYPE
 - activemq::util::PrimitiveValueNode, 2148
- SUBSCRIBE
 - activemq::wireformat::stomp::StompCommandConstants, 2587
- SYNC
 - inflate.h, 3149
- Scheduler
 - activemq::threads::Scheduler, 2318
- SchedulerTimerTask
 - activemq::threads::SchedulerTimerTask, 2320
- SecureRandom
 - decaf::security::SecureRandom, 2322
- SecureRandomImpl
 - decaf::internal::security::SecureRandomImpl, 2327
- SecureRandomSpi
 - decaf::security::SecureRandomSpi, 2330
- Semaphore
 - decaf::util::concurrent::Semaphore, 2333, 2334
- SendExecutor
 - activemq::cmsutil::CmsTemplate, 850
 - activemq::cmsutil::CmsTemplate::SendExecutor, 2342
- ServerSocket
 - decaf::net::ServerSocket, 2344–2346
 - decaf::net::Socket, 2469
- ServerSocketFactory
 - decaf::net::ServerSocketFactory, 2353
- ServiceStopper
 - activemq::util::ServiceStopper, 2358
- ServiceSupport
 - activemq::util::ServiceSupport, 2360
- ServiceSupport.h
 - SERVICESUPPORT_H_, 3046
- SessionId

- activemq::commands::SessionId, 2379
- SessionIdMarshaller
 - activemq::wireformat::openwire-
::marshal::generated::Session-
IdMarshaller, 2383
- SessionInfo
 - activemq::commands::SessionInfo, 2387
- SessionInfoMarshaller
 - activemq::wireformat::openwire-
::marshal::generated::Session-
InfoMarshaller, 2391
- SessionPool
 - activemq::cmsutil::SessionPool, 2394
- SessionState
 - activemq::state::SessionState, 2396
- Short
 - decaf::lang::Short, 2400
- ShortArrayBuffer
 - decaf::internal::nio::ShortArray-
Buffer, 2412, 2413
- ShortBuffer
 - decaf::nio::ShortBuffer, 2421
- ShutdownInfo
 - activemq::commands::ShutdownInfo, 2432
- ShutdownInfoMarshaller
 - activemq::wireformat::openwire-
::marshal::generated::Shutdown-
InfoMarshaller, 2435
- SignatureException
 - decaf::security::SignatureException, 2439, 2440
- SimpleFormatter
 - decaf::util::logging::SimpleFormatter, 2441
- SimpleLogger
 - decaf::util::logging::SimpleLogger, 2443
- SimplePriorityMessageDispatchChannel
 - activemq::core::SimplePriority-
MessageDispatchChannel, 2445
- Socket
 - decaf::net::Socket, 2455–2457
- SocketException
 - decaf::net::SocketException, 2472
- SocketFactory
 - decaf::net::SocketFactory, 2475
- SocketFileDescriptor
 - decaf::internal::net::SocketFile-
Descriptor, 2479
- SocketImpl
 - decaf::net::SocketImpl, 2481
- SocketTimeoutException
 - decaf::net::SocketTimeoutException, 2493, 2494
- SslTransport
 - activemq::transport::tcp::SslTransport, 2526
- StackTraceElement
 - activemq::commands::BrokerError::-
StackTraceElement, 2528
- StandardErrorOutputStream
 - decaf::internal::io::StandardError-
OutputStream, 2529
- StandardInputStream
 - decaf::internal::io::StandardInput-
Stream, 2531
- StandardOutputStream
 - decaf::internal::io::StandardOutput-
Stream, 2532
- State
 - decaf::lang::Thread, 2707
- StaticInitializer
 - activemq::core::ActiveMQConstants-
::StaticInitializer, 2535
- StlList
 - decaf::util::StlList, 2541
- StlMap
 - decaf::util::StlMap, 2556
- StlQueue
 - decaf::util::StlQueue, 2567
- StlSet
 - decaf::util::StlSet, 2576, 2577
- StompFrame
 - activemq::wireformat::stomp::Stomp-
Frame, 2589
- StompHelper
 - activemq::wireformat::stomp::Stomp-
Helper, 2594
- StompWireFormat
 - activemq::wireformat::stomp::Stomp-
WireFormat, 2598
- StompWireFormatFactory
 - activemq::wireformat::stomp::Stomp-
WireFormatFactory, 2601
- StreamHandler

- decaf::util::logging::StreamHandler, 2604
- String
 - decaf::lang::String, 2622, 2623
- StringTokenizer
 - decaf::util::StringTokenizer, 2628
- SubscriptionInfo
 - activemq::commands::SubscriptionInfo, 2632
- SubscriptionInfoMarshaller
 - activemq::wireformat::openwire::marshal::generated::SubscriptionInfoMarshaller, 2636
- SynchronizableImpl
 - decaf::internal::util::concurrent::SynchronizableImpl, 2651
- SynchronousQueue
 - decaf::util::concurrent::SynchronousQueue, 2657
- System
 - decaf::lang::System, 2667
- TABLE
 - inflate.h, 3149
- TEMPORARY_QUEUE
 - cms::Destination, 1211
- TEMPORARY_TOPIC
 - cms::Destination, 1211
- TEMPQUEUE_PREFIX
 - activemq::wireformat::stomp::StompCommandConstants, 2587
- TEMPTOPIC_PREFIX
 - activemq::wireformat::stomp::StompCommandConstants, 2587
- TEMP_POSTFIX
 - activemq::commands::ActiveMQDestination, 257
- TEMP_PREFIX
 - activemq::commands::ActiveMQDestination, 257
- TEMP_QUEUE_QUALIFED_PREFIX
 - activemq::commands::ActiveMQDestination, 257
- TEMP_TOPIC_QUALIFED_PREFIX
 - activemq::commands::ActiveMQDestination, 257
- TERMINATED
 - decaf::lang::Thread, 2707
- TEXT
 - activemq::wireformat::stomp::StompCommandConstants, 2587
- TIME
 - inflate.h, 3149
- TIMED_WAITING
 - decaf::lang::Thread, 2707
- TMENDRSCAN
 - cms::XAResource, 2933
- TMFAIL
 - cms::XAResource, 2933
- TMJOIN
 - cms::XAResource, 2933
- TMNOFLAGS
 - cms::XAResource, 2933
- TMONEPHASE
 - cms::XAResource, 2934
- TMRESUME
 - cms::XAResource, 2934
- TMSTARTRSCAN
 - cms::XAResource, 2934
- TMSUCCESS
 - cms::XAResource, 2934
- TMSUSPEND
 - cms::XAResource, 2934
- TOPIC
 - cms::Destination, 1211
- TOPIC_PREFIX
 - activemq::wireformat::stomp::StompCommandConstants, 2587
- TOPIC_QUALIFIED_PREFIX
 - activemq::commands::ActiveMQDestination, 257
- TRANSACTION_STATE_BEGIN
 - activemq::core::ActiveMQConstants, 233
- TRANSACTION_STATE_COMMITONEPHASE
 - activemq::core::ActiveMQConstants, 233
- TRANSACTION_STATE_COMMITTWOPHASE
 - activemq::core::ActiveMQConstants, 233
- TRANSACTION_STATE_END
 - activemq::core::ActiveMQConstants, 233
- TRANSACTION_STATE_FORGET
 - activemq::core::ActiveMQConstants, 233
- TRANSACTION_STATE_PREPARE
 - activemq::core::ActiveMQConstants, 233

- TRANSACTION_STATE_RECOVER
 - activemq::core::ActiveMQConstants, 233
- TRANSACTION_STATE_ROLLBACK
 - activemq::core::ActiveMQConstants, 233
- TRY_FREE
 - zutil.h, 3163
- TYPE
 - inflate.h, 3149
- TYPEDO
 - inflate.h, 3149
- TcpSocket
 - decaf::internal::net::tcp::TcpSocket, 2681
- TcpSocketInputStream
 - decaf::internal::net::tcp::TcpSocket-InputStream, 2689
- TcpSocketOutputStream
 - decaf::internal::net::tcp::TcpSocket-OutputStream, 2692
- TcpTransport
 - activemq::transport::tcp::TcpTransport, 2694
- Thread
 - decaf::lang::Thread, 2707, 2708
- ThreadGroup
 - decaf::lang::ThreadGroup, 2715
- ThreadPoolExecutor
 - decaf::util::concurrent::ThreadPool-Executor, 2719–2721
- Throwable
 - decaf::lang::Throwable, 2733
- Throwing
 - decaf::util::logging, 102
- TimeUnit
 - decaf::util::concurrent::TimeUnit, 2758
- TimeoutException
 - decaf::util::concurrent::Timeout-Exception, 2737, 2738
- Timer
 - decaf::util::Timer, 2741
 - decaf::util::TimerTask, 2753
- TimerImpl
 - decaf::util::TimerTask, 2753
- TimerTask
 - decaf::util::TimerTask, 2751
- TimerTaskHeap
 - decaf::internal::util::TimerTaskHeap, 2754
- Trace
 - zutil.h, 3162
- Tracec
 - zutil.h, 3162
- Tracecv
 - zutil.h, 3162
- Tracev
 - zutil.h, 3162
- Tracevv
 - zutil.h, 3162
- Tracked
 - activemq::state::Tracked, 2766
- TransactionId
 - activemq::commands::TransactionId, 2767
- TransactionIdMarshaller
 - activemq::wireformat::openwire-::marshal::generated::Transaction-IdMarshaller, 2771
- TransactionInProgressException
 - cms::TransactionInProgressException, 2783
- TransactionInfo
 - activemq::commands::Transaction-Info, 2775
- TransactionInfoMarshaller
 - activemq::wireformat::openwire-::marshal::generated::Transaction-InfoMarshaller, 2779
- TransactionRolledBackException
 - cms::TransactionRolledBackException, 2784
- TransactionState
 - activemq::core::ActiveMQConstants, 232
 - activemq::state::TransactionState, 2785
- TransferQueue
 - decaf::internal::util::concurrent::-TransferQueue, 2787
- TransferStack
 - decaf::internal::util::concurrent::-TransferStack, 2789
- TransportFilter
 - activemq::transport::TransportFilter, 2802
- UNSUBSCRIBE

- activemq::wireformat::stomp::Stomp-CommandConstants, 2587
- URI
 - decaf::net::URI, 2830–2832
- URLEncoderDecoder
 - decaf::internal::net::URLEncoder-Decoder, 2842
- URIHelper
 - decaf::internal::net::URIHelper, 2845, 2846
- URIParam
 - activemq::core::ActiveMQConstants, 233
- URIPool
 - activemq::transport::failover::URI-Pool, 2852
- URISyntaxException
 - decaf::net::URISyntaxException, 2857–2859
- URIType
 - decaf::internal::net::URIType, 2862
- URL
 - decaf::net::URL, 2870
- UTFDataFormatException
 - decaf::io::UTFDataFormatException, 2875, 2876
- UUID
 - decaf::util::UUID, 2879
- UnknownHostException
 - decaf::net::UnknownHostException, 2817, 2818
- UnknownServiceException
 - decaf::net::UnknownServiceException, 2819, 2820
- UnsupportedEncodingException
 - decaf::io::UnsupportedEncoding-Exception, 2822, 2823
- UnsupportedOperationException
 - cms::UnsupportedOperationException, 2828
 - decaf::lang::exceptions::Unsupported-OperationException, 2825, 2826
- WAITING
 - decaf::lang::Thread, 2707
- WAIT_INFINITE
 - Concurrent.h, 3231
- WARNING
 - decaf::util::logging::Level, 1620
- WIN_INIT
 - deflate.h, 3145
- WRITE_FAILURE
 - decaf::util::logging::ErrorManager, 1279
- Warn
 - decaf::util::logging, 102
- WireFormatInfo
 - activemq::commands::WireFormat-Info, 2892
- WireFormatInfoMarshaller
 - activemq::wireformat::openwire-::marshal::generated::Wire-FormatInfoMarshaller, 2901
- WireFormatNegotiator
 - activemq::wireformat::WireFormat-Negotiator, 2904
- WriteChecker
 - activemq::transport::inactivity::InactivityMonitor, 1428
 - activemq::transport::inactivity::Write-Checker, 2908
- Writer
 - decaf::io::Writer, 2910
- XAER_ASYNC
 - cms::XAException, 2925
- XAER_DUPID
 - cms::XAException, 2926
- XAER_INVALID
 - cms::XAException, 2926
- XAER_NOTA
 - cms::XAException, 2926
- XAER_OUTSIDE
 - cms::XAException, 2926
- XAER_PROTO
 - cms::XAException, 2926
- XAER_RMERR
 - cms::XAException, 2926
- XAER_RMFAIL
 - cms::XAException, 2926
- XAException
 - cms::XAException, 2923
- XATransactionId
 - activemq::commands::XATransaction-Id, 2938
- XATransactionIdMarshaller
 - activemq::wireformat::openwire-::marshal::generated::XA-TransactionIdMarshaller, 2943
- XA_HEURCOM
 - cms::XAException, 2924

- XA_HEURHAZ
 - cms::XAException, 2924
- XA_HEURMIX
 - cms::XAException, 2924
- XA_HEURRB
 - cms::XAException, 2924
- XA_NOMIGRATE
 - cms::XAException, 2924
- XA_OK
 - cms::XAResource, 2934
- XA_RBBASE
 - cms::XAException, 2924
- XA_RBCOMMFAIL
 - cms::XAException, 2924
- XA_RBDEADLOCK
 - cms::XAException, 2924
- XA_RBEND
 - cms::XAException, 2925
- XA_RBINTEGRITY
 - cms::XAException, 2925
- XA_RBOTHER
 - cms::XAException, 2925
- XA_RBPROTO
 - cms::XAException, 2925
- XA_RBROLLBACK
 - cms::XAException, 2925
- XA_RBTIMEOUT
 - cms::XAException, 2925
- XA_RBTRANSIENT
 - cms::XAException, 2925
- XA_RDONLY
 - cms::XAException, 2925
 - cms::XAResource, 2934
- XA_RETRY
 - cms::XAException, 2925
- XMLFormatter
 - decaf::util::logging::XMLFormatter, 2950
- Xid
 - cms::Xid, 2948
- ZALLOC
 - zutil.h, 3163
- ZFREE
 - zutil.h, 3163
- ZLIB_INTERNAL
 - gzguts.h, 3147
 - zutil.h, 3163
- ZLIB_VERNUM
 - zlib.h, 3158
- ZLIB_VERSION
 - zlib.h, 3158
- ZLIB_VER_MAJOR
 - zlib.h, 3158
- ZLIB_VER_MINOR
 - zlib.h, 3158
- ZLIB_VER_REVISION
 - zlib.h, 3158
- ZLIB_VER_SUBREVISION
 - zlib.h, 3158
- Z_ASCII
 - zlib.h, 3157
- Z_BEST_COMPRESSION
 - zlib.h, 3157
- Z_BEST_SPEED
 - zlib.h, 3157
- Z_BINARY
 - zlib.h, 3157
- Z_BLOCK
 - zlib.h, 3157
- Z_BUF_ERROR
 - zlib.h, 3157
- Z_DATA_ERROR
 - zlib.h, 3157
- Z_DEFAULT_COMPRESSION
 - zlib.h, 3157
- Z_DEFAULT_STRATEGY
 - zlib.h, 3157
- Z_DEFLATED
 - zlib.h, 3157
- Z_ERRNO
 - zlib.h, 3157
- Z_FILTERED
 - zlib.h, 3157
- Z_FINISH
 - zlib.h, 3157
- Z_FIXED
 - zlib.h, 3157
- Z_FULL_FLUSH
 - zlib.h, 3157
- Z_HUFFMAN_ONLY
 - zlib.h, 3157
- Z_MEM_ERROR
 - zlib.h, 3157
- Z_NEED_DICT
 - zlib.h, 3157
- Z_NO_COMPRESSION
 - zlib.h, 3157
- Z_NO_FLUSH
 - zlib.h, 3157
- Z_NULL
 - zlib.h, 3157

- zlib.h, 3157
- Z_OK
 - zlib.h, 3157
- Z_PARTIAL_FLUSH
 - zlib.h, 3158
- Z_RLE
 - zlib.h, 3158
- Z_STREAM_END
 - zlib.h, 3158
- Z_STREAM_ERROR
 - zlib.h, 3158
- Z_SYNC_FLUSH
 - zlib.h, 3158
- Z_TEXT
 - zlib.h, 3158
- Z_TREES
 - zlib.h, 3158
- Z_UNKNOWN
 - zlib.h, 3158
- Z_VERSION_ERROR
 - zlib.h, 3158
- ZipException
 - decaf::util::zip::ZipException, 2953, 2954
- _FALSE
 - decaf::lang::Boolean, 550
- _TRUE
 - decaf::lang::Boolean, 550
- _array
 - decaf::internal::nio::CharArrayBuffer, 785
- _capacity
 - decaf::nio::Buffer, 589
- _dist_code
 - deflate.h, 3146
 - trees.h, 3151
- _length_code
 - deflate.h, 3146
 - trees.h, 3151
- _limit
 - decaf::nio::Buffer, 589
- _mark
 - decaf::nio::Buffer, 589
- _markSet
 - decaf::nio::Buffer, 589
- _position
 - decaf::nio::Buffer, 589
- _tr_tally_dist
 - deflate.h, 3144
- _tr_tally_lit
 - deflate.h, 3144
- abs
 - decaf::lang::Math, 1802, 1803
- accept
 - decaf::internal::net::ssl::openssl:-
OpenSSLServerSocket, 2006
 - decaf::internal::net::tcp::TcpSocket, 2682
 - decaf::net::ServerSocket, 2346
 - decaf::net::SocketImpl, 2481
- accepted
 - decaf::net::Socket, 2457
- ackMode
 - activemq::core::ActiveMQSession, 354
- ackType
 - activemq::commands::MessageAck, 1872
- acknowledge
 - activemq::commands::ActiveMQ-
MessageTemplate, 296
 - activemq::core::ActiveMQConsumer, 236
 - activemq::core::ActiveMQSession, 337
 - cms::Message, 1843
- acknowledgeMessage
 - activemq::core::ActiveMQAck-
Handler, 156
- acquire
 - decaf::util::concurrent::Semaphore, 2334, 2335
- acquireUninterruptibly
 - decaf::util::concurrent::Semaphore, 2335, 2336
- action
 - activemq::cmsutil::CmsTemplate:-
ProducerExecutor, 2181
- activemq, 65
- activemq/exceptions/ExceptionDefines.h
 - AMQ_CATCHALL_NOTHROW, 3009
 - AMQ_CATCHALL_THROW, 3009
 - AMQ_CATCH_EXCEPTION_CON-
VERT, 3008
 - AMQ_CATCH_NOTHROW, 3008
 - AMQ_CATCH_RETHROW, 3008
- activemq/util/Config.h
 - AMQCPP_API, 3039

- HAVE_PTHREAD_H, 3039
- HAVE_UUID_T, 3039
- HAVE_UUID_UUID_H, 3039
- activemq::cmsutil, 66
- activemq::cmsutil::CachedConsumer, 734
 - ~CachedConsumer, 735
 - CachedConsumer, 735
 - close, 735
 - getMessageListener, 736
 - getMessageSelector, 736
 - receive, 736, 737
 - receiveNoWait, 737
 - setMessageListener, 737
 - start, 738
 - stop, 738
- activemq::cmsutil::CachedProducer, 738
 - ~CachedProducer, 740
 - CachedProducer, 740
 - close, 740
 - getDeliveryMode, 740
 - getDisableMessageID, 740
 - getDisableMessageTimeStamp, 740
 - getPriority, 741
 - getTimeToLive, 741
 - send, 741–743
 - setDeliveryMode, 744
 - setDisableMessageID, 744
 - setDisableMessageTimeStamp, 744
 - setPriority, 745
 - setTimeToLive, 745
- activemq::cmsutil::CmsAccessor, 819
 - ~CmsAccessor, 820
 - CmsAccessor, 820
 - checkConnectionFactory, 820
 - createConnection, 821
 - createSession, 821
 - destroy, 821
 - getConnectionFactory, 822
 - getResourceLifecycleManager, 822
 - getSessionAcknowledgeMode, 822
 - init, 822
 - operator=, 822
 - setConnectionFactory, 823
 - setSessionAcknowledgeMode, 823
- activemq::cmsutil::CmsDestination-
Accessor, 823
 - ~CmsDestinationAccessor, 824
 - CmsDestinationAccessor, 824
 - checkDestinationResolver, 824
 - destroy, 824
 - getDestinationResolver, 824, 825
 - init, 825
 - isPubSubDomain, 825
 - resolveDestinationName, 825
 - setDestinationResolver, 825
 - setPubSubDomain, 825
- activemq::cmsutil::CmsTemplate, 836
 - ~CmsTemplate, 839
 - CmsTemplate, 839
 - DEFAULT_PRIORITY, 850
 - DEFAULT_TIME_TO_LIVE, 850
 - ProducerExecutor, 850
 - RECEIVE_TIMEOUT_INDEFINITE-
_WAIT, 850
 - RECEIVE_TIMEOUT_NO_WAIT,
850
 - ReceiveExecutor, 850
 - ResolveProducerExecutor, 850
 - ResolveReceiveExecutor, 850
 - SendExecutor, 850
 - destroy, 840
 - execute, 840, 841
 - getDefaultDestination, 841, 842
 - getDefaultDestinationName, 842
 - getDeliveryMode, 842
 - getPriority, 842
 - getReceiveTimeout, 842
 - getTimeToLive, 842
 - init, 842
 - isExplicitQosEnabled, 843
 - isMessageIdEnabled, 843
 - isMessageTimestampEnabled, 843
 - isNoLocal, 843
 - receive, 843, 844
 - receiveSelected, 845
 - send, 846, 847
 - setDefaultDestination, 847
 - setDefaultDestinationName, 847
 - setDeliveryMode, 848
 - setDeliveryPersistent, 848
 - setExplicitQosEnabled, 848
 - setMessageIdEnabled, 849
 - setMessageTimestampEnabled, 849
 - setNoLocal, 849
 - setPriority, 849
 - setPubSubDomain, 849
 - setReceiveTimeout, 849
 - setTimeToLive, 850
- activemq::cmsutil::CmsTemplate::Producer-
Executor, 2180

- ~ProducerExecutor, 2181
- ProducerExecutor, 2180
- action, 2181
- destination, 2181
- doInCms, 2181
- getDestination, 2181
- parent, 2181
- activemq::cmsutil::CmsTemplate::Receive-
Executor, 2248
 - ~ReceiveExecutor, 2249
 - ReceiveExecutor, 2249
 - destination, 2250
 - doInCms, 2249
 - getDestination, 2249
 - getMessage, 2249
 - message, 2250
 - noLocal, 2250
 - parent, 2250
 - selector, 2250
- activemq::cmsutil::CmsTemplate::Resolve-
ProducerExecutor, 2291
 - ~ResolveProducerExecutor, 2291
 - ResolveProducerExecutor, 2291
 - getDestination, 2291
- activemq::cmsutil::CmsTemplate::Resolve-
ReceiveExecutor, 2292
 - ~ResolveReceiveExecutor, 2292
 - ResolveReceiveExecutor, 2292
 - getDestination, 2292
- activemq::cmsutil::CmsTemplate::Send-
Executor, 2341
 - ~SendExecutor, 2342
 - SendExecutor, 2342
 - doInCms, 2342
- activemq::cmsutil::DestinationResolver, 1222
 - ~DestinationResolver, 1222
 - destroy, 1222
 - init, 1223
 - resolveDestinationName, 1223
- activemq::cmsutil::DynamicDestination-
Resolver, 1272
 - ~DynamicDestinationResolver, 1272
 - DynamicDestinationResolver, 1272
 - destroy, 1272
 - init, 1273
 - resolveDestinationName, 1273
- activemq::cmsutil::MessageCreator, 1880
 - ~MessageCreator, 1881
 - createMessage, 1881
- activemq::cmsutil::PooledSession, 2092
 - ~PooledSession, 2095
 - PooledSession, 2095
 - close, 2095
 - commit, 2095
 - createBrowser, 2095, 2096
 - createBytesMessage, 2096, 2097
 - createCachedConsumer, 2097
 - createCachedProducer, 2098
 - createConsumer, 2098, 2099
 - createDurableConsumer, 2100
 - createMapMessage, 2101
 - createMessage, 2101
 - createProducer, 2101
 - createQueue, 2102
 - createStreamMessage, 2102
 - createTemporaryQueue, 2102
 - createTemporaryTopic, 2103
 - createTextMessage, 2103
 - createTopic, 2104
 - getAcknowledgeMode, 2104
 - getSession, 2104, 2105
 - isTransacted, 2105
 - recover, 2105
 - rollback, 2106
 - start, 2106
 - stop, 2106
 - unsubscribe, 2106
- activemq::cmsutil::ProducerCallback, 2179
 - ~ProducerCallback, 2179
 - doInCms, 2179
- activemq::cmsutil::ResourceLifecycle-
Manager, 2293
 - ~ResourceLifecycleManager, 2294
 - ResourceLifecycleManager, 2294
 - addConnection, 2295
 - addDestination, 2295
 - addMessageConsumer, 2295
 - addMessageProducer, 2295
 - addSession, 2296
 - destroy, 2296
 - operator=, 2296
 - releaseAll, 2296
- activemq::cmsutil::SessionCallback, 2377
 - ~SessionCallback, 2377
 - doInCms, 2377
- activemq::cmsutil::SessionPool, 2394
 - ~SessionPool, 2394

- SessionPool, 2394
- getResourceLifecycleManager, 2395
- returnSession, 2395
- takeSession, 2395
- activemq::commands, 67
- activemq::commands::ActiveMQBlob-
Message, 157
 - ~ActiveMQBlobMessage, 158
 - ActiveMQBlobMessage, 158
 - BINARY_MIME_TYPE, 161
 - ID_ACTIVEMQBLOBMESAGE, 161
 - clone, 158
 - cloneDataStructure, 158
 - copyDataStructure, 158
 - equals, 159
 - getDataStructureType, 159
 - getMimeType, 159
 - getName, 159
 - getRemoteBlobUrl, 160
 - isDeletedByBroker, 160
 - setDeletedByBroker, 160
 - setMimeType, 160
 - setName, 160
 - setRemoteBlobUrl, 161
 - toString, 161
- activemq::commands::ActiveMQBytes-
Message, 166
 - ~ActiveMQBytesMessage, 168
 - ActiveMQBytesMessage, 168
 - ID_ACTIVEMQBYTESMESSAGE, 183
 - clearBody, 168
 - clone, 168
 - cloneDataStructure, 169
 - copyDataStructure, 169
 - equals, 169
 - getBodyBytes, 169
 - getBodyLength, 170
 - getDataStructureType, 170
 - onSend, 170
 - readBoolean, 171
 - readByte, 171
 - readBytes, 171, 172
 - readChar, 173
 - readDouble, 173
 - readFloat, 174
 - readInt, 174
 - readLong, 175
 - readShort, 175
 - readString, 175
 - readUTF, 176
 - readUnsignedShort, 176
 - reset, 177
 - setBodyBytes, 177
 - toString, 177
 - writeBoolean, 178
 - writeByte, 178
 - writeBytes, 179
 - writeChar, 179
 - writeDouble, 180
 - writeFloat, 180
 - writeInt, 180
 - writeLong, 181
 - writeShort, 181
 - writeString, 182
 - writeUTF, 182
 - writeUnsignedShort, 182
- activemq::commands::ActiveMQDestination, 246
 - ~ActiveMQDestination, 249
 - ADVISORY_PREFIX, 256
 - ActiveMQDestination, 249
 - COMPOSITE_SEPARATOR, 256
 - CONNECTION_ADVISORY_PREFIX, 256
 - CONSUMER_ADVISORY_PREFIX, 256
 - DEFAULT_ORDERED_TARGET, 256
 - ID_ACTIVEMQDESTINATION, 256
 - PRODUCER_ADVISORY_PREFIX, 257
 - QUEUE_QUALIFIED_PREFIX, 257
 - TEMP_POSTFIX, 257
 - TEMP_PREFIX, 257
 - TEMP_QUEUE_QUALIFIED_PREFIX, 257
 - TEMP_TOPIC_QUALIFIED_PREFIX, 257
 - TOPIC_QUALIFIED_PREFIX, 257
 - advisory, 256
 - cloneDataStructure, 249
 - copyDataStructure, 249
 - createDestination, 249
 - createTemporaryName, 250
 - equals, 250
 - exclusive, 256
 - getCMSDestination, 251
 - getClientId, 250

- getDataStructureType, 251
- getDestinationType, 251
- getOptions, 252
- getOrderedTarget, 252
- getPhysicalName, 252
- isAdvisory, 252
- isComposite, 252
- isConnectionAdvisory, 253
- isConsumerAdvisory, 253
- isExclusive, 253
- isOrdered, 253
- isProducerAdvisory, 253
- isQueue, 253
- isTemporary, 254
- isTopic, 254
- isWildcard, 254
- options, 256
- ordered, 257
- orderedTarget, 257
- physicalName, 257
- setAdvisory, 254
- setExclusive, 254
- setOrdered, 255
- setOrderedTarget, 255
- setPhysicalName, 255
- toString, 255
- activemq::commands::ActiveMQDestination-
::DestinationFilter, 1213
 - ANY_CHILD, 1213
 - ANY_DESCENDENT, 1213
- activemq::commands::ActiveMQMap-
Message, 264
 - ~ActiveMQMapMessage, 270
 - ActiveMQMapMessage, 270
 - ID_ACTIVEMQMAPMESSAGE, 283
 - beforeMarshal, 271
 - checkMapsUnmarshalled, 271
 - clearBody, 271
 - clone, 271
 - cloneDataStructure, 272
 - copyDataStructure, 272
 - equals, 272
 - getBoolean, 273
 - getByte, 273
 - getBytes, 273
 - getChar, 274
 - getDataStructureType, 274
 - getDouble, 275
 - getFloat, 275
 - getInt, 275
 - getLong, 276
 - getMap, 276, 277
 - getMapNames, 277
 - getShort, 277
 - getString, 277
 - isEmpty, 278
 - isMarshalAware, 278
 - itemExists, 278
 - setBoolean, 279
 - setByte, 279
 - setBytes, 280
 - setChar, 280
 - setDouble, 281
 - setFloat, 281
 - setInt, 281
 - setLong, 282
 - setShort, 282
 - setString, 283
 - toString, 283
- activemq::commands::ActiveMQMessage,
288
 - ~ActiveMQMessage, 289
 - ActiveMQMessage, 289
 - ID_ACTIVEMQMESSAGE, 290
 - clone, 289
 - cloneDataStructure, 289
 - copyDataStructure, 289
 - equals, 289
 - getDataStructureType, 290
 - toString, 290
- activemq::commands::ActiveMQMessage-
Template
 - ~ActiveMQMessageTemplate, 296
 - ActiveMQMessageTemplate, 296
 - acknowledge, 296
 - clearBody, 296
 - clearProperties, 296
 - equals, 297
 - failIfReadOnlyBody, 297
 - failIfReadOnlyProperties, 297
 - failIfWriteOnlyBody, 297
 - getBooleanProperty, 297
 - getByteProperty, 297
 - getCMSCorrelationID, 297
 - getCMSDeliveryMode, 297
 - getCMSDestination, 298
 - getCMSExpiration, 298
 - getCMSMessageID, 298
 - getCMSPriority, 298
 - getCMSRedelivered, 298

- getCMSReplyTo, 298
- getCMSTimestamp, 298
- getCMSType, 298
- getDoubleProperty, 298
- getFloatProperty, 298
- getIntProperty, 298
- getLongProperty, 298
- getPropertyNames, 299
- getShortProperty, 299
- getStringProperty, 299
- onSend, 299
- propertyExists, 299
- setBooleanProperty, 299
- setByteProperty, 299
- setCMSCorrelationID, 299
- setCMSDeliveryMode, 299
- setCMSDestination, 299
- setCMSExpiration, 300
- setCMSMessageID, 300
- setCMSPriority, 300
- setCMSRedelivered, 300
- setCMSReplyTo, 300
- setCMSTimestamp, 300
- setCMSType, 300
- setDoubleProperty, 300
- setFloatProperty, 300
- setIntProperty, 300
- setLongProperty, 300
- setShortProperty, 300
- setStringProperty, 301
- activemq::commands::ActiveMQMessage-
Template< T >, 295
- activemq::commands::ActiveMQObject-
Message, 301
 - ~ActiveMQObjectMessage, 302
 - ActiveMQObjectMessage, 302
 - ID_ACTIVEMQOBJECTMESSAGE,
303
 - clone, 302
 - cloneDataStructure, 302
 - copyDataStructure, 302
 - equals, 302
 - getDataStructureType, 303
 - toString, 303
- activemq::commands::ActiveMQQueue,
321
 - ~ActiveMQQueue, 322
 - ActiveMQQueue, 322
 - ID_ACTIVEMQQUEUE, 325
 - clone, 322
 - cloneDataStructure, 322
 - copy, 323
 - copyDataStructure, 323
 - equals, 323
 - getCMSDestination, 324
 - getCMSProperties, 324
 - getDataStructureType, 324
 - getDestinationType, 324
 - getQueueName, 325
 - toString, 325
- activemq::commands::ActiveMQStream-
Message, 359
 - ~ActiveMQStreamMessage, 361
 - ActiveMQStreamMessage, 361
 - ID_ACTIVEMQSTREAMMESSAGE,
374
 - clearBody, 361
 - clone, 361
 - cloneDataStructure, 362
 - copyDataStructure, 362
 - equals, 362
 - getDataStructureType, 362
 - onSend, 363
 - readBoolean, 363
 - readByte, 363
 - readBytes, 364, 365
 - readChar, 365
 - readDouble, 366
 - readFloat, 366
 - readInt, 367
 - readLong, 367
 - readShort, 368
 - readString, 368
 - readUnsignedShort, 369
 - reset, 369
 - toString, 369
 - writeBoolean, 369
 - writeByte, 370
 - writeBytes, 370, 371
 - writeChar, 371
 - writeDouble, 371
 - writeFloat, 372
 - writeInt, 372
 - writeLong, 373
 - writeShort, 373
 - writeString, 373
 - writeUnsignedShort, 374
- activemq::commands::ActiveMQTemp-
Destination, 379
 - ~ActiveMQTempDestination, 380

- ActiveMQTempDestination, 380
- ID_ACTIVEMQTEMPDESTINATION, 382
- cloneDataStructure, 380
- close, 380
- connection, 382
- copyDataStructure, 380
- equals, 381
- getDataStructureType, 381
- setConnection, 381
- toString, 382
- activemq::commands::ActiveMQTempQueue, 386
 - ~ActiveMQTempQueue, 387
- ActiveMQTempQueue, 387
- ID_ACTIVEMQTEMPQUEUE, 391
- clone, 387
- cloneDataStructure, 388
- copy, 388
- copyDataStructure, 388
- destroy, 388
- equals, 389
- getCMSDestination, 389
- getCMSProperties, 390
- getDataStructureType, 390
- getDestinationType, 390
- getQueueName, 390
- toString, 391
- activemq::commands::ActiveMQTempTopic, 396
 - ~ActiveMQTempTopic, 397
- ActiveMQTempTopic, 397
- ID_ACTIVEMQTEMPTOPIC, 400
- clone, 397
- cloneDataStructure, 397
- copy, 397
- copyDataStructure, 398
- destroy, 398
- equals, 398
- getCMSDestination, 399
- getCMSProperties, 399
- getDataStructureType, 399
- getDestinationType, 399
- getTopicName, 400
- toString, 400
- activemq::commands::ActiveMQTextMessage, 405
 - ~ActiveMQTextMessage, 406
- ActiveMQTextMessage, 406
- ID_ACTIVEMQTEXTMESSAGE, 410
 - beforeMarshal, 406
 - clearBody, 406
 - clone, 407
 - cloneDataStructure, 407
 - copyDataStructure, 407
 - equals, 407
 - getDataStructureType, 408
 - getSize, 408
 - getText, 408
 - setText, 409
 - text, 410
 - toString, 409
- activemq::commands::ActiveMQTopic, 414
 - ~ActiveMQTopic, 415
- ActiveMQTopic, 415
- ID_ACTIVEMQTOPIC, 418
- clone, 415
- cloneDataStructure, 415
- copy, 416
- copyDataStructure, 416
- equals, 416
- getCMSDestination, 417
- getCMSProperties, 417
- getDataStructureType, 417
- getDestinationType, 417
- getTopicName, 418
- toString, 418
- activemq::commands::BaseCommand, 491
 - ~BaseCommand, 493
- BaseCommand, 493
- copyDataStructure, 493
- equals, 493
- getCommandId, 494
- isBrokerInfo, 495
- isConnectionControl, 495
- isConnectionInfo, 495
- isConsumerInfo, 495
- isKeepAliveInfo, 495
- isMessage, 495
- isMessageAck, 496
- isMessageDispatch, 496
- isMessageDispatchNotification, 496
- isProducerAck, 496
- isProducerInfo, 496
- isRemoveInfo, 496
- isRemoveSubscriptionInfo, 496

- isResponse, 497
- isResponseRequired, 497
- isShutdownInfo, 497
- isTransactionInfo, 497
- isWireFormatInfo, 497
- setCommandId, 497
- setResponseRequired, 498
- toString, 498
- activemq::commands::BaseDataStructure, 528
 - ~BaseDataStructure, 529
 - afterMarshal, 529
 - afterUnmarshal, 529
 - beforeMarshal, 529
 - beforeUnmarshal, 530
 - copyDataStructure, 530
 - equals, 530
 - getMarshaledForm, 530
 - isMarshalAware, 530
 - setMarshaledForm, 530
 - toString, 531
- activemq::commands::BooleanExpression, 551
 - ~BooleanExpression, 551
 - BooleanExpression, 551
 - cloneDataStructure, 551
 - copyDataStructure, 551
 - equals, 552
 - toString, 552
- activemq::commands::BrokerError, 558
 - ~BrokerError, 560
 - BrokerError, 560
 - cloneDataStructure, 560
 - copyDataStructure, 560
 - getCause, 560
 - getDataStructureType, 560
 - getExceptionClass, 561
 - getMessage, 561
 - getStackTraceElements, 561
 - setCause, 561
 - setExceptionClass, 562
 - setMessage, 562
 - setStackTraceElements, 562
 - visit, 562
- activemq::commands::BrokerError::Stack-TraceElement, 2527
 - ClassName, 2528
 - FileName, 2528
 - LineNumber, 2528
 - MethodName, 2528
 - StackTraceElement, 2528
- activemq::commands::BrokerId, 564
 - ~BrokerId, 565
 - BrokerId, 565
 - COMPARATOR, 565
 - ID_BROKERID, 567
 - cloneDataStructure, 565
 - compareTo, 566
 - copyDataStructure, 566
 - equals, 566
 - getDataStructureType, 566
 - getValue, 566, 567
 - operator<, 567
 - operator=, 567
 - operator==, 567
 - setValue, 567
 - toString, 567
 - value, 567
- activemq::commands::BrokerInfo, 572
 - ~BrokerInfo, 573
 - BrokerInfo, 573
 - ID_BROKERINFO, 578
 - brokerId, 577
 - brokerName, 577
 - brokerURL, 577
 - brokerUploadUrl, 577
 - cloneDataStructure, 574
 - connectionId, 577
 - copyDataStructure, 574
 - duplexConnection, 578
 - equals, 574
 - faultTolerantConfiguration, 578
 - getBrokerId, 574
 - getBrokerName, 574, 575
 - getBrokerURL, 575
 - getBrokerUploadUrl, 575
 - getConnectionId, 575
 - getDataStructureType, 575
 - getNetworkProperties, 575
 - getPeerBrokerInfos, 575
 - isBrokerInfo, 575
 - isDuplexConnection, 576
 - isFaultTolerantConfiguration, 576
 - isMasterBroker, 576
 - isNetworkConnection, 576
 - isSlaveBroker, 576
 - masterBroker, 578
 - networkConnection, 578
 - networkProperties, 578
 - peerBrokerInfos, 578

- setBrokerId, 576
- setBrokerName, 576
- setBrokerURL, 576
- setBrokerUploadUrl, 576
- setConnectionId, 576
- setDuplexConnection, 576
- setFaultTolerantConfiguration, 576
- setMasterBroker, 576
- setNetworkConnection, 576
- setNetworkProperties, 576
- setPeerBrokerInfos, 577
- setSlaveBroker, 577
- slaveBroker, 578
- toString, 577
- visit, 577
- activemq::commands::Command, 866
 - ~Command, 867
 - getCommandId, 867
 - isBrokerInfo, 867
 - isConnectionControl, 867
 - isConnectionInfo, 867
 - isConsumerInfo, 867
 - isKeepAliveInfo, 868
 - isMessage, 868
 - isMessageAck, 868
 - isMessageDispatch, 868
 - isMessageDispatchNotification, 868
 - isProducerAck, 868
 - isProducerInfo, 868
 - isRemoveInfo, 869
 - isRemoveSubscriptionInfo, 869
 - isResponse, 869
 - isResponseRequired, 869
 - isShutdownInfo, 869
 - isTransactionInfo, 869
 - isWireFormatInfo, 869
 - setCommandId, 870
 - setResponseRequired, 870
 - toString, 870
 - visit, 871
- activemq::commands::ConnectionControl, 938
 - ~ConnectionControl, 940
 - ConnectionControl, 940
 - ID_CONNECTIONCONTROL, 943
 - cloneDataStructure, 940
 - close, 943
 - connectedBrokers, 943
 - copyDataStructure, 940
 - equals, 940
 - exit, 943
 - faultTolerant, 943
 - getConnectedBrokers, 940, 941
 - getDataStructureType, 941
 - getReconnectTo, 941
 - isClose, 941
 - isConnectionControl, 941
 - isExit, 941
 - isFaultTolerant, 941
 - isRebalanceConnection, 941
 - isResume, 941
 - isSuspend, 941
 - rebalanceConnection, 943
 - reconnectTo, 943
 - resume, 943
 - setClose, 942
 - setConnectedBrokers, 942
 - setExit, 942
 - setFaultTolerant, 942
 - setRebalanceConnection, 942
 - setReconnectTo, 942
 - setResume, 942
 - setSuspend, 942
 - suspend, 943
 - toString, 942
 - visit, 942
- activemq::commands::ConnectionError, 948
 - ~ConnectionError, 949
 - ConnectionError, 949
 - ID_CONNECTIONERROR, 951
 - cloneDataStructure, 949
 - connectionId, 951
 - copyDataStructure, 949
 - equals, 949
 - exception, 951
 - getConnectionId, 949, 950
 - getDataStructureType, 950
 - getException, 950
 - setConnectionId, 950
 - setException, 950
 - toString, 950
 - visit, 950
- activemq::commands::ConnectionId, 960
 - ~ConnectionId, 961
 - COMPARATOR, 961
 - ConnectionId, 961
 - ID_CONNECTIONID, 963
 - cloneDataStructure, 961
 - compareTo, 961

- copyDataStructure, 962
- equals, 962
- getDataStructureType, 962
- getValue, 962
- operator<, 963
- operator=, 963
- operator==, 963
- setValue, 963
- toString, 963
- value, 963
- activemq::commands::ConnectionInfo, 968
 - ~ConnectionInfo, 969
 - ConnectionInfo, 969
 - ID_CONNECTIONINFO, 973
 - brokerMasterConnector, 973
 - brokerPath, 973
 - clientId, 973
 - clientMaster, 973
 - cloneDataStructure, 969
 - connectionId, 973
 - copyDataStructure, 970
 - createRemoveCommand, 970
 - equals, 970
 - failoverReconnect, 973
 - faultTolerant, 973
 - getBrokerPath, 970
 - getClientId, 970
 - getConnectionId, 970, 971
 - getDataStructureType, 971
 - getPassword, 971
 - getUserName, 971
 - isBrokerMasterConnector, 971
 - isClientMaster, 971
 - isConnectionInfo, 971
 - isFailoverReconnect, 971
 - isFaultTolerant, 971
 - isManageable, 972
 - manageable, 973
 - password, 973
 - setBrokerMasterConnector, 972
 - setBrokerPath, 972
 - setClientId, 972
 - setClientMaster, 972
 - setConnectionId, 972
 - setFailoverReconnect, 972
 - setFaultTolerant, 972
 - setManageable, 972
 - setPassword, 972
 - setUserName, 972
 - toString, 972
 - userName, 973
 - visit, 972
- activemq::commands::ConsumerControl, 992
 - ~ConsumerControl, 993
 - ConsumerControl, 993
 - ID_CONSUMERCONTROL, 996
 - cloneDataStructure, 993
 - close, 996
 - consumerId, 996
 - copyDataStructure, 993
 - destination, 996
 - equals, 993
 - flush, 996
 - getConsumerId, 994
 - getDataStructureType, 994
 - getDestination, 994
 - getPrefetch, 994
 - isClose, 994
 - isFlush, 994
 - isStart, 994
 - isStop, 995
 - prefetch, 996
 - setClose, 995
 - setConsumerId, 995
 - setDestination, 995
 - setFlush, 995
 - setPrefetch, 995
 - setStart, 995
 - setStop, 995
 - start, 996
 - stop, 996
 - toString, 995
 - visit, 995
- activemq::commands::ConsumerId, 1000
 - ~ConsumerId, 1002
 - COMPARATOR, 1002
 - ConsumerId, 1002
 - ID_CONSUMERID, 1004
 - cloneDataStructure, 1002
 - compareTo, 1002
 - connectionId, 1004
 - copyDataStructure, 1002
 - equals, 1002, 1003
 - getConnectionId, 1003
 - getDataStructureType, 1003
 - getParentId, 1003
 - getSessionId, 1003
 - getValue, 1003

- operator<, 1003
- operator=, 1003
- operator==, 1003
- sessionId, 1004
- setConnectionId, 1004
- setSessionId, 1004
- setValue, 1004
- toString, 1004
- value, 1004
- activemq::commands::ConsumerInfo, 1009
 - ~ConsumerInfo, 1011
 - ConsumerInfo, 1011
 - ID_CONSUMERINFO, 1017
 - additionalPredicate, 1016
 - brokerPath, 1016
 - browser, 1016
 - cloneDataStructure, 1011
 - consumerId, 1016
 - copyDataStructure, 1011
 - createRemoveCommand, 1012
 - destination, 1016
 - dispatchAsync, 1016
 - equals, 1012
 - exclusive, 1016
 - getAdditionalPredicate, 1012
 - getBrokerPath, 1012
 - getConsumerId, 1012
 - getDataStructureType, 1013
 - getDestination, 1013
 - getMaximumPendingMessageLimit, 1013
 - getNetworkConsumerPath, 1013
 - getPrefetchSize, 1013
 - getPriority, 1013
 - getSelector, 1013
 - getSubscriptionName, 1013, 1014
 - isBrowser, 1014
 - isConsumerInfo, 1014
 - isDispatchAsync, 1014
 - isExclusive, 1014
 - isNetworkSubscription, 1014
 - isNoLocal, 1014
 - isNoRangeAcks, 1014
 - isOptimizedAcknowledge, 1014
 - isRetroactive, 1014
 - maximumPendingMessageLimit, 1017
 - networkConsumerPath, 1017
 - networkSubscription, 1017
 - noLocal, 1017
 - noRangeAcks, 1017
 - optimizedAcknowledge, 1017
 - prefetchSize, 1017
 - priority, 1017
 - retroactive, 1017
 - selector, 1017
 - setAdditionalPredicate, 1014
 - setBrokerPath, 1014
 - setBrowser, 1014
 - setConsumerId, 1015
 - setDestination, 1015
 - setDispatchAsync, 1015
 - setExclusive, 1015
 - setMaximumPendingMessageLimit, 1015
 - setNetworkConsumerPath, 1015
 - setNetworkSubscription, 1015
 - setNoLocal, 1015
 - setNoRangeAcks, 1015
 - setOptimizedAcknowledge, 1015
 - setPrefetchSize, 1015
 - setPriority, 1015
 - setRetroactive, 1015
 - setSelector, 1015
 - setSubscriptionName, 1015
 - subscriptionName, 1017
 - toString, 1016
 - visit, 1016
- activemq::commands::ControlCommand, 1022
 - ~ControlCommand, 1023
 - ControlCommand, 1023
 - ID_CONTROLCOMMAND, 1025
 - cloneDataStructure, 1023
 - command, 1025
 - copyDataStructure, 1024
 - equals, 1024
 - getCommand, 1024
 - getDataStructureType, 1024
 - setCommand, 1024
 - toString, 1025
 - visit, 1025
- activemq::commands::DataArrayResponse, 1069
 - ~DataArrayResponse, 1070
 - DataArrayResponse, 1070
 - ID_DATAARRAYRESPONSE, 1071
 - cloneDataStructure, 1070
 - copyDataStructure, 1070

- data, 1071
- equals, 1070
- getData, 1070, 1071
- getDataStructureType, 1071
- setData, 1071
- toString, 1071
- activemq::commands::DataResponse, 1112
 - ~DataResponse, 1113
 - DataResponse, 1113
 - ID_DATARESPONSE, 1115
 - cloneDataStructure, 1113
 - copyDataStructure, 1113
 - data, 1115
 - equals, 1114
 - getData, 1114
 - getDataStructureType, 1114
 - setData, 1114
 - toString, 1114
- activemq::commands::DataStructure, 1133
 - ~DataStructure, 1133
 - cloneDataStructure, 1133
 - copyDataStructure, 1134
 - equals, 1135
 - getDataStructureType, 1137
 - toString, 1138
- activemq::commands::DestinationInfo, 1213
 - ~DestinationInfo, 1215
 - DestinationInfo, 1215
 - ID_DESTINATIONINFO, 1217
 - brokerPath, 1217
 - cloneDataStructure, 1215
 - connectionId, 1217
 - copyDataStructure, 1215
 - destination, 1217
 - equals, 1215
 - getBrokerPath, 1215, 1216
 - getConnectionId, 1216
 - getDataStructureType, 1216
 - getDestination, 1216
 - getOperationType, 1216
 - getTimeout, 1216
 - operationType, 1217
 - setBrokerPath, 1216
 - setConnectionId, 1216
 - setDestination, 1216
 - setOperationType, 1217
 - setTimeout, 1217
- timeout, 1218
- toString, 1217
- visit, 1217
- activemq::commands::DiscoveryEvent, 1226
 - ~DiscoveryEvent, 1227
 - DiscoveryEvent, 1227
 - ID_DISCOVERYEVENT, 1229
 - brokerName, 1229
 - cloneDataStructure, 1227
 - copyDataStructure, 1228
 - equals, 1228
 - getBrokerName, 1228
 - getDataStructureType, 1228
 - getServiceName, 1229
 - serviceName, 1229
 - setBrokerName, 1229
 - setServiceName, 1229
 - toString, 1229
- activemq::commands::ExceptionResponse, 1287
 - ~ExceptionResponse, 1288
 - ExceptionResponse, 1288
 - ID_EXCEPTIONRESPONSE, 1290
 - cloneDataStructure, 1288
 - copyDataStructure, 1289
 - equals, 1289
 - exception, 1290
 - getDataStructureType, 1289
 - getException, 1289, 1290
 - setException, 1290
 - toString, 1290
- activemq::commands::FlushCommand, 1381
 - ~FlushCommand, 1381
 - FlushCommand, 1381
 - ID_FLUSHCOMMAND, 1383
 - cloneDataStructure, 1381
 - copyDataStructure, 1382
 - equals, 1382
 - getDataStructureType, 1382
 - toString, 1382
 - visit, 1383
- activemq::commands::IntegerResponse, 1516
 - ~IntegerResponse, 1517
 - ID_INTEGERRESPONSE, 1519
 - IntegerResponse, 1517
 - cloneDataStructure, 1517
 - copyDataStructure, 1517

- equals, 1518
- getDataStructureType, 1518
- getResult, 1518
- result, 1519
- setResult, 1518
- toString, 1518
- activemq::commands::JournalQueueAck, 1561
 - ~JournalQueueAck, 1562
 - ID_JOURNALQUEUEACK, 1563
 - JournalQueueAck, 1562
 - cloneDataStructure, 1562
 - copyDataStructure, 1562
 - destination, 1563
 - equals, 1562
 - getDataStructureType, 1562
 - getDestination, 1563
 - getMessageAck, 1563
 - messageAck, 1564
 - setDestination, 1563
 - setMessageAck, 1563
 - toString, 1563
- activemq::commands::JournalTopicAck, 1568
 - ~JournalTopicAck, 1569
 - ID_JOURNALTOPICACK, 1572
 - JournalTopicAck, 1569
 - clientId, 1572
 - cloneDataStructure, 1569
 - copyDataStructure, 1569
 - destination, 1572
 - equals, 1570
 - getClientId, 1570
 - getDataStructureType, 1570
 - getDestination, 1570
 - getMessageId, 1571
 - getMessageSequenceId, 1571
 - getSubscriptionName, 1571
 - getTransactionId, 1571
 - messageId, 1572
 - messageSequenceId, 1572
 - setClientId, 1571
 - setDestination, 1571
 - setMessageId, 1571
 - setMessageSequenceId, 1571
 - setSubscriptionName, 1571
 - setTransactionId, 1571
 - subscriptionName, 1572
 - toString, 1571
 - transactionId, 1572
- activemq::commands::JournalTrace, 1577
 - ~JournalTrace, 1577
 - ID_JOURNALTRACE, 1579
 - JournalTrace, 1577
 - cloneDataStructure, 1578
 - copyDataStructure, 1578
 - equals, 1578
 - getDataStructureType, 1578
 - getMessage, 1578, 1579
 - message, 1579
 - setMessage, 1579
 - toString, 1579
- activemq::commands::JournalTransaction, 1583
 - ~JournalTransaction, 1584
 - ID_JOURNALTRANSACTION, 1586
 - JournalTransaction, 1584
 - cloneDataStructure, 1585
 - copyDataStructure, 1585
 - equals, 1585
 - getDataStructureType, 1585
 - getTransactionId, 1586
 - getType, 1586
 - getWasPrepared, 1586
 - setTransactionId, 1586
 - setType, 1586
 - setWasPrepared, 1586
 - toString, 1586
 - transactionId, 1586
 - type, 1587
 - wasPrepared, 1587
- activemq::commands::KeepAliveInfo, 1591
 - ~KeepAliveInfo, 1592
 - ID_KEEPAIVEINFO, 1594
 - KeepAliveInfo, 1592
 - cloneDataStructure, 1592
 - copyDataStructure, 1592
 - equals, 1592
 - getDataStructureType, 1593
 - isKeepAliveInfo, 1593
 - toString, 1593
 - visit, 1593
- activemq::commands::LastPartialCommand, 1606
 - ~LastPartialCommand, 1606
 - ID_LASTPARTIALCOMMAND, 1608
 - LastPartialCommand, 1606
 - cloneDataStructure, 1606
 - copyDataStructure, 1607

- equals, 1607
- getDataStructureType, 1607
- toString, 1607
- activemq::commands::LocalTransactionId, 1674
 - ~LocalTransactionId, 1676
 - COMPARATOR, 1675
 - ID_LOCALTRANSACTIONID, 1678
 - LocalTransactionId, 1676
 - cloneDataStructure, 1676
 - compareTo, 1676
 - connectionId, 1678
 - copyDataStructure, 1676
 - equals, 1676, 1677
 - getConnectionId, 1677
 - getDataStructureType, 1677
 - getValue, 1677
 - isLocalTransactionId, 1677
 - operator<, 1677
 - operator=, 1677
 - operator==, 1677
 - setConnectionId, 1677
 - setValue, 1677
 - toString, 1678
 - value, 1678
- activemq::commands::Message, 1821
 - ~Message, 1826
 - DEFAULT_MESSAGE_SIZE, 1837
 - ID_MESSAGE, 1838
 - Message, 1826
 - afterUnmarshal, 1826
 - arrival, 1837
 - beforeMarshal, 1826
 - brokerInTime, 1837
 - brokerOutTime, 1837
 - brokerPath, 1837
 - cloneDataStructure, 1826
 - cluster, 1837
 - compressed, 1837
 - connection, 1837
 - content, 1837
 - copyDataStructure, 1827
 - correlationId, 1837
 - dataStructure, 1837
 - destination, 1838
 - droppable, 1838
 - equals, 1827
 - expiration, 1838
 - getAckHandler, 1828
 - getArrival, 1828
 - getBrokerInTime, 1828
 - getBrokerOutTime, 1828
 - getBrokerPath, 1828
 - getCluster, 1828
 - getConnection, 1828
 - getContent, 1828, 1829
 - getCorrelationId, 1829
 - getDataStructure, 1829
 - getDataStructureType, 1829
 - getDestination, 1829
 - getExpiration, 1829
 - getGroupId, 1829, 1830
 - getGroupSequence, 1830
 - getMarshaledProperties, 1830
 - getMessageId, 1830
 - getMessageProperties, 1830
 - getOriginalDestination, 1830
 - getOriginalTransactionId, 1830
 - getPriority, 1831
 - getProducerId, 1831
 - getRedeliveryCounter, 1831
 - getReplyTo, 1831
 - getSize, 1831
 - getTargetConsumerId, 1831
 - getTimestamp, 1831
 - getTransactionId, 1831
 - getType, 1832
 - getUserId, 1832
 - groupId, 1838
 - groupSequence, 1838
 - isCompressed, 1832
 - isDroppable, 1832
 - isExpired, 1832
 - isMarshalAware, 1832
 - isMessage, 1832
 - isPersistent, 1833
 - isReadOnlyBody, 1833
 - isReadOnlyProperties, 1833
 - isRecievedByDFBridge, 1833
 - marshalledProperties, 1838
 - messageId, 1838
 - onSend, 1833
 - originalDestination, 1838
 - originalTransactionId, 1838
 - persistent, 1838
 - priority, 1838
 - producerId, 1838
 - recievedByDFBridge, 1838
 - redeliveryCounter, 1838
 - replyTo, 1838

- setAckHandler, 1833
- setArrival, 1834
- setBrokerInTime, 1834
- setBrokerOutTime, 1834
- setBrokerPath, 1834
- setCluster, 1834
- setCompressed, 1834
- setConnection, 1834
- setContent, 1834
- setCorrelationId, 1834
- setDataStructure, 1834
- setDestination, 1835
- setDroppable, 1835
- setExpiration, 1835
- setGroupId, 1835
- setGroupSequence, 1835
- setMarshaledProperties, 1835
- setMessageId, 1835
- setOriginalDestination, 1835
- setOriginalTransactionId, 1835
- setPersistent, 1835
- setPriority, 1835
- setProducerId, 1835
- setReadOnlyBody, 1835
- setReadOnlyProperties, 1835
- setReceivedByDFBridge, 1836
- setRedeliveryCounter, 1836
- setReplyTo, 1836
- setTargetConsumerId, 1836
- setTimestamp, 1836
- setTransactionId, 1836
- setType, 1836
- setUserId, 1836
- targetConsumerId, 1838
- timestamp, 1838
- toString, 1836
- transactionId, 1839
- type, 1839
- userId, 1839
- visit, 1837
- activemq::commands::MessageAck, 1867
 - ~MessageAck, 1869
 - ID_MESSAGEACK, 1872
 - MessageAck, 1869
 - ackType, 1872
 - cloneDataStructure, 1869
 - consumerId, 1872
 - copyDataStructure, 1869
 - destination, 1872
 - equals, 1869
 - firstMessageId, 1872
 - getAckType, 1870
 - getConsumerId, 1870
 - getDataStructureType, 1870
 - getDestination, 1870
 - getFirstMessageId, 1870
 - getLastMessageId, 1870
 - getMessageCount, 1871
 - getTransactionId, 1871
 - isMessageAck, 1871
 - lastMessageId, 1872
 - messageCount, 1872
 - setAckType, 1871
 - setConsumerId, 1871
 - setDestination, 1871
 - setFirstMessageId, 1871
 - setLastMessageId, 1871
 - setMessageCount, 1871
 - setTransactionId, 1871
 - toString, 1871
 - transactionId, 1872
 - visit, 1872
- activemq::commands::MessageDispatch, 1881
 - ~MessageDispatch, 1883
 - ID_MESSAGEDISPATCH, 1885
 - MessageDispatch, 1883
 - cloneDataStructure, 1883
 - consumerId, 1885
 - copyDataStructure, 1883
 - destination, 1885
 - equals, 1883
 - getConsumerId, 1883, 1884
 - getDataStructureType, 1884
 - getDestination, 1884
 - getMessage, 1884
 - getRedeliveryCounter, 1884
 - isMessageDispatch, 1884
 - message, 1885
 - redeliveryCounter, 1886
 - setConsumerId, 1884
 - setDestination, 1884
 - setMessage, 1885
 - setRedeliveryCounter, 1885
 - toString, 1885
 - visit, 1885
- activemq::commands::MessageDispatch-Notification, 1894
 - ~MessageDispatchNotification, 1896

- ID_MESSAGE_DISPATCH_NOTIFICATION, 1899
- MessageDispatchNotification, 1896
- cloneDataStructure, 1896
- consumerId, 1899
- copyDataStructure, 1896
- deliverySequenceId, 1899
- destination, 1899
- equals, 1896
- getConsumerId, 1897
- getDataStructureType, 1897
- getDeliverySequenceId, 1897
- getDestination, 1897
- getMessageId, 1897
- isMessageDispatchNotification, 1897
- messageId, 1899
- setConsumerId, 1898
- setDeliverySequenceId, 1898
- setDestination, 1898
- setMessageId, 1898
- toString, 1898
- visit, 1898
- activemq::commands::MessageId, 1908
 - ~MessageId, 1909
 - COMPARATOR, 1909
 - ID_MESSAGEID, 1912
 - MessageId, 1909
 - brokerSequenceId, 1912
 - cloneDataStructure, 1909
 - compareTo, 1910
 - copyDataStructure, 1910
 - equals, 1910
 - getBrokerSequenceId, 1910
 - getDataStructureType, 1910
 - getProducerId, 1911
 - getProducerSequenceId, 1911
 - operator<, 1911
 - operator=, 1911
 - operator==, 1911
 - producerId, 1912
 - producerSequenceId, 1912
 - setBrokerSequenceId, 1911
 - setProducerId, 1911
 - setProducerSequenceId, 1911
 - setTextView, 1911
 - setValue, 1911
 - toString, 1911
- activemq::commands::MessagePull, 1939
 - ~MessagePull, 1941
 - ID_MESSAGEPULL, 1943
 - MessagePull, 1941
 - cloneDataStructure, 1941
 - consumerId, 1943
 - copyDataStructure, 1941
 - correlationId, 1943
 - destination, 1943
 - equals, 1941
 - getConsumerId, 1941, 1942
 - getCorrelationId, 1942
 - getDataStructureType, 1942
 - getDestination, 1942
 - getMessageId, 1942
 - getTimeout, 1942
 - messageId, 1943
 - setConsumerId, 1942
 - setCorrelationId, 1942
 - setDestination, 1942
 - setMessageId, 1943
 - setTimeout, 1943
 - timeout, 1944
 - toString, 1943
 - visit, 1943
- activemq::commands::NetworkBridgeFilter, 1971
 - ~NetworkBridgeFilter, 1972
 - ID_NETWORKBRIDGEFILTER, 1974
 - NetworkBridgeFilter, 1972
 - cloneDataStructure, 1972
 - copyDataStructure, 1972
 - equals, 1973
 - getDataStructureType, 1973
 - getNetworkBrokerId, 1973
 - getNetworkTTL, 1973
 - networkBrokerId, 1974
 - networkTTL, 1974
 - setNetworkBrokerId, 1973
 - setNetworkTTL, 1973
 - toString, 1974
- activemq::commands::PartialCommand, 2076
 - ~PartialCommand, 2077
 - ID_PARTIALCOMMAND, 2079
 - PartialCommand, 2077
 - cloneDataStructure, 2077
 - commandId, 2079
 - copyDataStructure, 2077
 - data, 2079
 - equals, 2077
 - getCommandId, 2078

- getData, 2078
- getDataStructureType, 2078
- setCommandId, 2078
- setData, 2078
- toString, 2078
- activemq::commands::ProducerAck, 2171
 - ~ProducerAck, 2172
 - ID_PRODUCERACK, 2174
 - ProducerAck, 2172
 - cloneDataStructure, 2172
 - copyDataStructure, 2172
 - equals, 2173
 - getDataStructureType, 2173
 - getProducerId, 2173
 - getSize, 2173
 - isProducerAck, 2173
 - producerId, 2174
 - setProducerId, 2174
 - setSize, 2174
 - size, 2174
 - toString, 2174
 - visit, 2174
- activemq::commands::ProducerId, 2182
 - ~ProducerId, 2183
 - COMPARATOR, 2183
 - ID_PRODUCERID, 2185
 - ProducerId, 2183
 - cloneDataStructure, 2183
 - compareTo, 2183
 - connectionId, 2185
 - copyDataStructure, 2183
 - equals, 2184
 - getConnectionId, 2184
 - getDataStructureType, 2184
 - getParentId, 2184
 - getSessionId, 2185
 - getValue, 2185
 - operator<, 2185
 - operator=, 2185
 - operator==, 2185
 - sessionId, 2186
 - setConnectionId, 2185
 - setProducerSessionKey, 2185
 - setSessionId, 2185
 - setValue, 2185
 - toString, 2185
 - value, 2186
- activemq::commands::ProducerInfo, 2190
 - ~ProducerInfo, 2191
 - ID_PRODUCERINFO, 2194
 - ProducerInfo, 2191
 - brokerPath, 2194
 - cloneDataStructure, 2191
 - copyDataStructure, 2192
 - createRemoveCommand, 2192
 - destination, 2194
 - dispatchAsync, 2194
 - equals, 2192
 - getBrokerPath, 2192
 - getDataStructureType, 2192
 - getDestination, 2193
 - getProducerId, 2193
 - getWindowSize, 2193
 - isDispatchAsync, 2193
 - isProducerInfo, 2193
 - producerId, 2194
 - setBrokerPath, 2193
 - setDestination, 2193
 - setDispatchAsync, 2193
 - setProducerId, 2193
 - setWindowSize, 2193
 - toString, 2194
 - visit, 2194
 - windowSize, 2194
- activemq::commands::RemoveInfo, 2267
 - ~RemoveInfo, 2268
 - ID_REMOVEINFO, 2271
 - RemoveInfo, 2268
 - cloneDataStructure, 2269
 - copyDataStructure, 2269
 - equals, 2269
 - getDataStructureType, 2269
 - getLastDeliveredSequenceId, 2269
 - getObjectId, 2270
 - isRemoveInfo, 2270
 - lastDeliveredSequenceId, 2271
 - objectId, 2271
 - setLastDeliveredSequenceId, 2270
 - setObjectId, 2270
 - toString, 2270
 - visit, 2270
- activemq::commands::RemoveSubscriptionInfo, 2275
 - ~RemoveSubscriptionInfo, 2276
 - ID_REMOVESUBSCRIPTIONINFO, 2279
 - RemoveSubscriptionInfo, 2276
 - clientId, 2279
 - cloneDataStructure, 2276
 - connectionId, 2279

- copyDataStructure, 2277
- equals, 2277
- getClientId, 2277
- getConnectionId, 2277, 2278
- getDataStructureType, 2278
- getSubscriptionName, 2278
- isRemoveSubscriptionInfo, 2278
- setClientId, 2278
- setConnectionId, 2278
- setSubscriptionName, 2278
- subscriptionName, 2279
- toString, 2278
- visit, 2279
- activemq::commands::ReplayCommand, 2284
 - ~ReplayCommand, 2285
 - ID_REPLAYCOMMAND, 2287
 - ReplayCommand, 2285
 - cloneDataStructure, 2285
 - copyDataStructure, 2285
 - equals, 2285
 - firstNakNumber, 2287
 - getDataStructureType, 2285
 - getFirstNakNumber, 2286
 - getLastNakNumber, 2286
 - lastNakNumber, 2287
 - setFirstNakNumber, 2286
 - setLastNakNumber, 2286
 - toString, 2286
 - visit, 2286
- activemq::commands::Response, 2298
 - ~Response, 2299
 - ID_RESPONSE, 2301
 - Response, 2299
 - cloneDataStructure, 2299
 - copyDataStructure, 2299
 - correlationId, 2301
 - equals, 2299
 - getCorrelationId, 2300
 - getDataStructureType, 2300
 - isResponse, 2300
 - setCorrelationId, 2300
 - toString, 2300
 - visit, 2301
- activemq::commands::SessionId, 2378
 - ~SessionId, 2379
 - COMPARATOR, 2379
 - ID_SESSIONID, 2382
 - SessionId, 2379
 - cloneDataStructure, 2379
 - compareTo, 2380
 - connectionId, 2382
 - copyDataStructure, 2380
 - equals, 2380
 - getConnectionId, 2380
 - getDataStructureType, 2381
 - getParentId, 2381
 - getValue, 2381
 - operator<, 2381
 - operator=, 2381
 - operator==, 2381
 - setConnectionId, 2381
 - setValue, 2381
 - toString, 2381
 - value, 2382
- activemq::commands::SessionInfo, 2386
 - ~SessionInfo, 2387
 - ID_SESSIONINFO, 2389
 - SessionInfo, 2387
 - cloneDataStructure, 2387
 - copyDataStructure, 2387
 - createRemoveCommand, 2388
 - equals, 2388
 - getAckMode, 2388
 - getDataStructureType, 2388
 - getSessionId, 2388
 - sessionId, 2389
 - setAckMode, 2389
 - setSessionId, 2389
 - toString, 2389
 - visit, 2389
- activemq::commands::ShutdownInfo, 2431
 - ~ShutdownInfo, 2432
 - ID_SHUTDOWNINFO, 2434
 - ShutdownInfo, 2432
 - cloneDataStructure, 2432
 - copyDataStructure, 2432
 - equals, 2433
 - getDataStructureType, 2433
 - isShutdownInfo, 2433
 - toString, 2433
 - visit, 2433
- activemq::commands::SubscriptionInfo, 2631
 - ~SubscriptionInfo, 2632
 - ID_SUBSCRIPTIONINFO, 2634
 - SubscriptionInfo, 2632
 - clientId, 2634
 - cloneDataStructure, 2632

- copyDataStructure, 2632
- destination, 2634
- equals, 2632
- getClientId, 2633
- getDataStructureType, 2633
- getDestination, 2633
- getSelector, 2633
- getSubscriptionName, 2633
- getSubscribedDestination, 2634
- selector, 2635
- setClientId, 2634
- setDestination, 2634
- setSelector, 2634
- setSubscriptionName, 2634
- setSubscribedDestination, 2634
- subscriptionName, 2635
- subscribedDestination, 2635
- toString, 2634
- activemq::commands::TransactionId, 2766
 - ~TransactionId, 2767
 - COMPARATOR, 2767
 - ID_TRANSACTIONID, 2770
 - TransactionId, 2767
 - cloneDataStructure, 2768
 - compareTo, 2768
 - copyDataStructure, 2768
 - equals, 2768, 2769
 - getDataStructureType, 2769
 - isLocalTransactionId, 2769
 - isXATransactionId, 2769
 - operator<, 2769
 - operator=, 2769
 - operator==, 2769
 - toString, 2769
- activemq::commands::TransactionInfo, 2774
 - ~TransactionInfo, 2775
 - ID_TRANSACTIONINFO, 2778
 - TransactionInfo, 2775
 - cloneDataStructure, 2775
 - connectionId, 2777
 - copyDataStructure, 2775
 - equals, 2776
 - getConnectionId, 2776
 - getDataStructureType, 2776
 - getTransactionId, 2776
 - getType, 2776
 - isTransactionInfo, 2776
 - setConnectionId, 2777
 - setTransactionId, 2777
 - setType, 2777
 - toString, 2777
 - transactionId, 2778
 - type, 2778
 - visit, 2777
- activemq::commands::WireFormatInfo, 2889
 - ~WireFormatInfo, 2892
 - ID_WIREFORMATINFO, 2899
 - WireFormatInfo, 2892
 - afterUnmarshal, 2892
 - beforeMarshal, 2892
 - cloneDataStructure, 2892
 - copyDataStructure, 2892
 - equals, 2892
 - getCacheSize, 2893
 - getDataStructureType, 2893
 - getMagic, 2893
 - getMarshaledProperties, 2893
 - getMaxInactivityDuration, 2893
 - getMaxInactivityDurationInitialDelay, 2894
 - getProperties, 2894
 - getVersion, 2894
 - isCacheEnabled, 2894
 - isMarshalAware, 2895
 - isSizePrefixDisabled, 2895
 - isStackTraceEnabled, 2895
 - isTcpNoDelayEnabled, 2895
 - isTightEncodingEnabled, 2896
 - isValid, 2896
 - isWireFormatInfo, 2896
 - setCacheEnabled, 2896
 - setCacheSize, 2896
 - setMagic, 2897
 - setMarshaledProperties, 2897
 - setMaxInactivityDuration, 2897
 - setMaxInactivityDurationInitialDelay, 2897
 - setProperties, 2898
 - setSizePrefixDisabled, 2898
 - setStackTraceEnabled, 2898
 - setTcpNoDelayEnabled, 2898
 - setTightEncodingEnabled, 2898
 - setVersion, 2899
 - toString, 2899
 - visit, 2899
- activemq::commands::XATransactionId, 2936

- ~XATransactionId, 2938
- COMPARATOR, 2938
- ID_XATRANSACTIONID, 2942
- XATransactionId, 2938
- branchQualifier, 2942
- clone, 2938
- cloneDataStructure, 2938
- compareTo, 2938
- copyDataStructure, 2939
- equals, 2939
- formatId, 2942
- getBranchQualifier, 2939, 2940
- getDataStructureType, 2940
- getFormatId, 2940
- getGlobalTransactionId, 2940, 2941
- globalTransactionId, 2942
- isXATransactionId, 2941
- operator<, 2941
- operator=, 2941
- operator==, 2941
- setBranchQualifier, 2941
- setFormatId, 2941
- setGlobalTransactionId, 2941
- toString, 2942
- activemq::core, 68
- activemq::core::ActiveMQAckHandler, 156
 - ~ActiveMQAckHandler, 156
 - acknowledgeMessage, 156
- activemq::core::ActiveMQConnection, 187
 - ~ActiveMQConnection, 193
 - ActiveMQConnection, 193
 - addDispatcher, 193
 - addProducer, 194
 - addSession, 194
 - addTransportListener, 194
 - checkClosed, 195
 - checkClosedOrFailed, 195
 - cleanup, 195
 - close, 195
 - createSession, 196
 - destroyDestination, 196, 197
 - disconnect, 197
 - ensureConnectionInfoSent, 197
 - fire, 197
 - getBrokerURL, 198
 - getClientId, 198
 - getCloseTimeout, 198
 - getCompressionLevel, 198
 - getConnectionId, 198
 - getConnectionInfo, 199
 - getExceptionListener, 199
 - getFirstFailureError, 199
 - getMetaData, 199
 - getNextLocalTransactionId, 200
 - getNextSessionId, 200
 - getNextTempDestinationId, 200
 - getPassword, 201
 - getPrefetchPolicy, 201
 - getProducerWindowSize, 201
 - getProperties, 201
 - getRedeliveryPolicy, 201
 - getResourceManagerId, 202
 - getScheduler, 202
 - getSendTimeout, 202
 - getTransport, 202
 - getUsername, 202
 - isAlwaysSyncSend, 203
 - isClosed, 203
 - isDispatchAsync, 203
 - isMessagePrioritySupported, 203
 - isStarted, 203
 - isTransportFailed, 203
 - isUseAsyncSend, 204
 - isUseCompression, 204
 - onAsyncException, 204
 - onCommand, 204
 - onException, 205
 - oneway, 204
 - removeDispatcher, 205
 - removeProducer, 205
 - removeSession, 206
 - removeTransportListener, 206
 - sendPullRequest, 206
 - setAlwaysSyncSend, 207
 - setBrokerURL, 207
 - setClientId, 207
 - setCloseTimeout, 208
 - setCompressionLevel, 208
 - setDefaultClientId, 208
 - setDispatchAsync, 208
 - setExceptionListener, 209
 - setMessagePrioritySupported, 209
 - setPassword, 209
 - setPrefetchPolicy, 209
 - setProducerWindowSize, 210
 - setRedeliveryPolicy, 210
 - setSendTimeout, 210
 - setTransportInterruptProcessing-Complete, 211

- setUseAsyncSend, 211
- setUseCompression, 211
- setUsername, 211
- signalInterruptionProcessingComplete, 211
- start, 211
- stop, 212
- syncRequest, 212
- transportInterrupted, 212
- transportResumed, 212
- waitForTransportInterruptionProcessingToComplete, 213
- activemq::core::ActiveMQConnectionFactory, 213
 - ~ActiveMQConnectionFactory, 216
 - ActiveMQConnectionFactory, 216
 - DEFAULT_URI, 226
 - createActiveMQConnection, 216
 - createConnection, 217, 218
 - getBrokerURI, 219
 - getClientId, 219
 - getCloseTimeout, 219
 - getCompressionLevel, 219
 - getExceptionListener, 220
 - getPassword, 220
 - getPrefetchPolicy, 220
 - getProducerWindowSize, 220
 - getRedeliveryPolicy, 221
 - getSendTimeout, 221
 - getUsername, 221
 - isAlwaysSyncSend, 221
 - isDispatchAsync, 221
 - isMessagePrioritySupported, 222
 - isUseAsyncSend, 222
 - isUseCompression, 222
 - setAlwaysSyncSend, 222
 - setBrokerURI, 222, 223
 - setClientId, 223
 - setCloseTimeout, 223
 - setCompressionLevel, 223
 - setDispatchAsync, 224
 - setExceptionListener, 224
 - setMessagePrioritySupported, 224
 - setPassword, 224
 - setPrefetchPolicy, 225
 - setProducerWindowSize, 225
 - setRedeliveryPolicy, 225
 - setSendTimeout, 225
 - setUseAsyncSend, 226
 - setUseCompression, 226
 - setUsername, 226
- activemq::core::ActiveMQConnectionMetaData, 227
 - ~ActiveMQConnectionMetaData, 228
 - ActiveMQConnectionMetaData, 228
 - getCMSMajorVersion, 228
 - getCMSMinorVersion, 228
 - getCMSProviderName, 228
 - getCMSVersion, 229
 - getCMSXPropertyNames, 229
 - getProviderMajorVersion, 229
 - getProviderMinorVersion, 230
 - getProviderVersion, 230
- activemq::core::ActiveMQConstants, 231
 - ACK_TYPE_CONSUMED, 232
 - ACK_TYPE_DELIVERED, 232
 - ACK_TYPE_INDIVIDUAL, 232
 - ACK_TYPE_POISON, 232
 - ACK_TYPE_REDELIVERED, 232
 - AckType, 232
 - CONNECTION_ALWAYS_SYNCSEND, 233
 - CONNECTION_CLOSE_TIMEOUT, 233
 - CONNECTION_DISPATCH_ASYNC, 233
 - CONNECTION_PRODUCER_WINDOW_SIZE, 233
 - CONNECTION_SEND_TIMEOUT, 233
 - CONNECTION_USE_ASYNC_SEND, 233
 - CONNECTION_USE_COMPRESSION, 233
 - CONSUMER_DISPATCH_ASYNC, 232
 - CONSUMER_EXCLUSIVE, 232
 - CONSUMER_NOLOCAL, 232
 - CONSUMER_PREFETCH_SIZE, 232
 - CONSUMER_PRIORITY, 232
 - CONSUMER_RETROACTIVE, 232
 - CONSUMER_SELECTOR, 232
 - CUNSUMER_MAX_PENDING_MSG_LIMIT, 232
 - DESTINATION_ADD_OPERATION, 232
 - DESTINATION_REMOVE_OPERATION, 232

- DestinationActions, 232
- DestinationOption, 232
- NUM_OPTIONS, 232
- NUM_PARAMS, 233
- PARAM_CLIENTID, 233
- PARAM_PASSWORD, 233
- PARAM_USERNAME, 233
- TRANSACTION_STATE_BEGIN, 233
- TRANSACTION_STATE_COMMIT-ONEPHASE, 233
- TRANSACTION_STATE_COMMIT-TWOPHASE, 233
- TRANSACTION_STATE_END, 233
- TRANSACTION_STATE_FORGET, 233
- TRANSACTION_STATE_PREPARE, 233
- TRANSACTION_STATE_RECOVER, 233
- TRANSACTION_STATE_ROLLBACK, 233
- TransactionState, 232
- URIParam, 233
- toDestinationOption, 233
- toString, 233
- toURIOption, 233
- activemq::core::ActiveMQConstants:-
 - StaticInitializer, 2535
 - ~StaticInitializer, 2535
 - StaticInitializer, 2535
 - destOptionMap, 2535
 - destOptions, 2535
 - uriParams, 2536
 - uriParamsMap, 2536
- activemq::core::ActiveMQConsumer, 234
 - ~ActiveMQConsumer, 236
 - ActiveMQConsumer, 236
 - acknowledge, 236
 - afterMessagesConsumed, 236
 - beforeMessagesConsumed, 237
 - clearMessagesInProgress, 237
 - close, 237
 - commit, 237
 - deliverAcks, 237
 - dequeue, 238
 - dispatch, 238
 - dispose, 238
 - doClose, 239
 - getConsumerId, 239
 - getConsumerInfo, 239
 - getFailureError, 239
 - getLastDeliveredSequenceId, 239
 - getMessageAvailableCount, 240
 - getMessageListener, 240
 - getMessageSelector, 240
 - getRedeliveryPolicy, 240
 - inProgressClearRequired, 241
 - isClosed, 241
 - isSynchronizationRegistered, 241
 - iterate, 241
 - receive, 241, 242
 - receiveNoWait, 242
 - rollback, 242
 - setFailureError, 242
 - setLastDeliveredSequenceId, 243
 - setMessageListener, 243
 - setRedeliveryPolicy, 243
 - setSynchronizationRegistered, 243
 - start, 244
 - stop, 244
- activemq::core::ActiveMQProducer, 308
 - ~ActiveMQProducer, 309
 - ActiveMQProducer, 309
 - close, 309
 - dispose, 310
 - getDeliveryMode, 310
 - getDisableMessageID, 310
 - getDisableMessageTimeStamp, 310
 - getPriority, 311
 - getProducerId, 311
 - getProducerInfo, 311
 - getSendTimeout, 311
 - getTimeToLive, 311
 - isClosed, 312
 - onProducerAck, 312
 - send, 312–314
 - setDeliveryMode, 314
 - setDisableMessageID, 314
 - setDisableMessageTimeStamp, 315
 - setPriority, 315
 - setSendTimeout, 315
 - setTimeToLive, 315
- activemq::core::ActiveMQQueueBrowser, 326
 - ~ActiveMQQueueBrowser, 326
 - ActiveMQQueueBrowser, 326
 - Browser, 329
 - close, 327
 - getEnumeration, 327

- getMessageSelector, 327
- getQueue, 327
- hasMoreMessages, 328
- nextMessage, 328
- activemq::core::ActiveMQSession, 333
 - ~ActiveMQSession, 337
 - ActiveMQSession, 337
 - ActiveMQSessionExecutor, 354
 - ackMode, 354
 - acknowledge, 337
 - addConsumer, 337
 - addProducer, 338
 - clearMessagesInProgress, 338
 - close, 338
 - closed, 354
 - commit, 338
 - config, 354
 - connection, 354
 - consumerIds, 354
 - consumers, 354
 - createBrowser, 338, 339
 - createBytesMessage, 339, 340
 - createConsumer, 340, 341
 - createDurableConsumer, 342
 - createMapMessage, 342
 - createMessage, 343
 - createProducer, 343
 - createQueue, 343
 - createStreamMessage, 344
 - createTemporaryQueue, 344
 - createTemporaryTopic, 344
 - createTextMessage, 344, 345
 - createTopic, 345
 - deliverAcks, 345
 - dispatch, 346
 - dispose, 346
 - doClose, 346
 - doStartTransaction, 346
 - executor, 354
 - fire, 346
 - getAcknowledgeMode, 347
 - getConnection, 347
 - getExceptionListener, 347
 - getLastDeliveredSequenceId, 347
 - getNextConsumerId, 347
 - getNextProducerId, 348
 - getScheduler, 348
 - getSessionId, 348
 - getSessionInfo, 348
 - getTransactionContext, 348
 - isAutoAcknowledge, 349
 - isClientAcknowledge, 349
 - isDupsOkAcknowledge, 349
 - isIndividualAcknowledge, 349
 - isStarted, 349
 - isTransacted, 349
 - lastDeliveredSequenceId, 354
 - oneway, 350
 - producerIds, 354
 - producerSequenceIds, 355
 - recover, 350
 - redispatch, 350
 - removeConsumer, 351
 - removeProducer, 351
 - rollback, 351
 - send, 352
 - sessionInfo, 355
 - setLastDeliveredSequenceId, 352
 - start, 352
 - stop, 352
 - syncRequest, 353
 - transaction, 355
 - unsubscribe, 353
 - wakeup, 353
- activemq::core::ActiveMQSessionExecutor, 355
 - ~ActiveMQSessionExecutor, 356
 - ActiveMQSessionExecutor, 356
 - clear, 356
 - clearMessagesInProgress, 356
 - close, 357
 - execute, 357
 - executeFirst, 357
 - getUnconsumedMessages, 357
 - hasUnconsumedMessages, 358
 - isEmpty, 358
 - isRunning, 358
 - iterate, 358
 - start, 358
 - stop, 358
 - wakeup, 359
- activemq::core::ActiveMQTransactionContext, 423
 - ~ActiveMQTransactionContext, 425
 - ActiveMQTransactionContext, 425
 - addSynchronization, 425
 - begin, 425
 - commit, 425
 - end, 426
 - forget, 427

- getTransactionId, 427
- getTransactionTimeout, 427
- isInLocalTransaction, 428
- isInTransaction, 428
- isInXATransaction, 428
- isSameRM, 428
- prepare, 429
- recover, 429
- removeSynchronization, 430
- rollback, 430
- setTransactionTimeout, 431
- start, 431
- activemq::core::ActiveMQXAConnection, 432
 - ~ActiveMQXAConnection, 432
 - ActiveMQXAConnection, 432
 - createSession, 433
 - createXASession, 433
- activemq::core::ActiveMQXAConnectionFactory, 433
 - ~ActiveMQXAConnectionFactory, 435
 - ActiveMQXAConnectionFactory, 434
 - createActiveMQConnection, 435
 - createXAConnection, 435, 436
- activemq::core::ActiveMQXASession, 436
 - ~ActiveMQXASession, 437
 - ActiveMQXASession, 437
 - commit, 437
 - doStartTransaction, 437
 - getXAResource, 438
 - isAutoAcknowledge, 438
 - isTransacted, 438
 - rollback, 438
- activemq::core::DispatchData, 1234
 - DispatchData, 1234
 - getConsumerId, 1234
 - getMessage, 1234
- activemq::core::Dispatcher, 1234
 - ~Dispatcher, 1235
 - dispatch, 1235
- activemq::core::FifoMessageDispatchChannel, 1322
 - ~FifoMessageDispatchChannel, 1324
 - FifoMessageDispatchChannel, 1324
 - clear, 1324
 - close, 1324
 - dequeue, 1324
 - dequeueNoWait, 1325
 - enqueue, 1325
 - enqueueFirst, 1325
 - isClosed, 1325
 - isEmpty, 1326
 - isRunning, 1326
 - lock, 1326
 - notify, 1326
 - notifyAll, 1327
 - peek, 1327
 - removeAll, 1327
 - size, 1327
 - start, 1328
 - stop, 1328
 - tryLock, 1328
 - unlock, 1328
 - wait, 1328, 1329
- activemq::core::MessageDispatchChannel, 1886
 - ~MessageDispatchChannel, 1887
 - clear, 1887
 - close, 1887
 - dequeue, 1887
 - dequeueNoWait, 1888
 - enqueue, 1888
 - enqueueFirst, 1888
 - isClosed, 1888
 - isEmpty, 1889
 - isRunning, 1889
 - peek, 1889
 - removeAll, 1889
 - size, 1890
 - start, 1890
 - stop, 1890
- activemq::core::PrefetchPolicy, 2110
 - ~PrefetchPolicy, 2111
 - PrefetchPolicy, 2111
 - clone, 2111
 - configure, 2111
 - getDurableTopicPrefetch, 2112
 - getMaxPrefetchLimit, 2112
 - getQueueBrowserPrefetch, 2112
 - getQueuePrefetch, 2112
 - getTopicPrefetch, 2113
 - setDurableTopicPrefetch, 2113
 - setQueueBrowserPrefetch, 2113
 - setQueuePrefetch, 2113
 - setTopicPrefetch, 2114
- activemq::core::RedeliveryPolicy, 2250
 - ~RedeliveryPolicy, 2252

- NO_MAXIMUM_REDELIVERIES, 2256
- RedeliveryPolicy, 2252
 - clone, 2252
 - configure, 2252
 - getBackOffMultiplier, 2252
 - getCollisionAvoidancePercent, 2253
 - getInitialRedeliveryDelay, 2253
 - getMaximumRedeliveries, 2253
 - getNextRedeliveryDelay, 2253
 - getRedeliveryDelay, 2254
 - isUseCollisionAvoidance, 2254
 - isUseExponentialBackOff, 2254
 - setBackOffMultiplier, 2254
 - setCollisionAvoidancePercent, 2254
 - setInitialRedeliveryDelay, 2255
 - setMaximumRedeliveries, 2255
 - setRedeliveryDelay, 2255
 - setUseCollisionAvoidance, 2255
 - setUseExponentialBackOff, 2256
- activemq::core::SimplePriorityMessageDispatchChannel, 2444
 - ~SimplePriorityMessageDispatchChannel, 2445
- SimplePriorityMessageDispatchChannel, 2445
- clear, 2445
- close, 2446
- dequeue, 2446
- dequeueNoWait, 2446
- enqueue, 2446
- enqueueFirst, 2447
- isClosed, 2447
- isEmpty, 2447
- isRunning, 2447
- lock, 2447
- notify, 2448
- notifyAll, 2448
- peek, 2448
- removeAll, 2449
- size, 2449
- start, 2449
- stop, 2449
- tryLock, 2449
- unlock, 2450
- wait, 2450, 2451
- activemq::core::Synchronization, 2654
 - ~Synchronization, 2654
 - afterCommit, 2654
 - afterRollback, 2654
 - beforeEnd, 2654
- activemq::core::policies, 69
- activemq::core::policies::DefaultPrefetchPolicy, 1146
 - ~DefaultPrefetchPolicy, 1147
- DEFAULT_DURABLE_TOPIC_PREFETCH, 1149
- DEFAULT_QUEUE_BROWSER_PREFETCH, 1149
- DEFAULT_QUEUE_PREFETCH, 1149
- DEFAULT_TOPIC_PREFETCH, 1149
- DefaultPrefetchPolicy, 1147
- MAX_PREFETCH_SIZE, 1150
- clone, 1147
- getDurableTopicPrefetch, 1147
- getMaxPrefetchLimit, 1147
- getQueueBrowserPrefetch, 1147
- getQueuePrefetch, 1148
- getTopicPrefetch, 1148
- setDurableTopicPrefetch, 1148
- setQueueBrowserPrefetch, 1148
- setQueuePrefetch, 1149
- setTopicPrefetch, 1149
- activemq::core::policies::DefaultRedeliveryPolicy, 1150
 - ~DefaultRedeliveryPolicy, 1151
- DefaultRedeliveryPolicy, 1151
- clone, 1151
- getBackOffMultiplier, 1151
- getCollisionAvoidancePercent, 1151
- getInitialRedeliveryDelay, 1151
- getMaximumRedeliveries, 1152
- getNextRedeliveryDelay, 1152
- getRedeliveryDelay, 1152
- isUseCollisionAvoidance, 1153
- isUseExponentialBackOff, 1153
- setBackOffMultiplier, 1153
- setCollisionAvoidancePercent, 1153
- setInitialRedeliveryDelay, 1153
- setMaximumRedeliveries, 1154
- setRedeliveryDelay, 1154
- setUseCollisionAvoidance, 1154
- setUseExponentialBackOff, 1154
- activemq::exceptions, 70
- activemq::exceptions::ActiveMQException, 262
 - ~ActiveMQException, 263
- ActiveMQException, 262, 263

- clone, 263
- convertToCMSException, 263
- activemq::exceptions::BrokerException, 563
 - ~BrokerException, 564
 - BrokerException, 563
 - clone, 564
- activemq::exceptions::ConnectionFailedException, 958
 - ~ConnectionFailedException, 959
 - ConnectionFailedException, 959
 - clone, 959
- activemq::io, 70
- activemq::io::LoggingInputStream, 1706
 - ~LoggingInputStream, 1707
 - LoggingInputStream, 1707
 - doReadArrayBounded, 1707
 - doReadByte, 1707
- activemq::io::LoggingOutputStream, 1707
 - ~LoggingOutputStream, 1708
 - LoggingOutputStream, 1708
 - doWriteArrayBounded, 1708
 - doWriteByte, 1708
- activemq::library, 70
- activemq::library::ActiveMQCPP, 244
 - ~ActiveMQCPP, 245
 - ActiveMQCPP, 245
 - activemq::util::IdGenerator, 1413
 - initializeLibrary, 245
 - operator=, 246
 - shutdownLibrary, 246
- activemq::state, 70
- activemq::state::CommandVisitor, 872
 - ~CommandVisitor, 874
 - processBeginTransaction, 874
 - processBrokerError, 874
 - processBrokerInfo, 874
 - processCommitTransactionOnePhase, 874
 - processCommitTransactionTwoPhase, 874
 - processConnectionControl, 874
 - processConnectionError, 875
 - processConnectionInfo, 875
 - processConsumerControl, 875
 - processConsumerInfo, 875
 - processControlCommand, 875
 - processDestinationInfo, 875
 - processEndTransaction, 875
 - processFlushCommand, 875
 - processForgetTransaction, 875
 - processKeepAliveInfo, 876
 - processMessage, 876
 - processMessageAck, 876
 - processMessageDispatch, 876
 - processMessageDispatchNotification, 876
 - processMessagePull, 876
 - processPrepareTransaction, 876
 - processProducerAck, 876
 - processProducerInfo, 876
 - processRecoverTransactions, 877
 - processRemoveConnection, 877
 - processRemoveConsumer, 877
 - processRemoveDestination, 877
 - processRemoveInfo, 877
 - processRemoveProducer, 877
 - processRemoveSession, 877
 - processRemoveSubscriptionInfo, 877
 - processReplayCommand, 878
 - processResponse, 878
 - processRollbackTransaction, 878
 - processSessionInfo, 878
 - processShutdownInfo, 878
 - processTransactionInfo, 878
 - processWireFormat, 878
- activemq::state::CommandVisitorAdapter, 878
 - ~CommandVisitorAdapter, 881
 - processBeginTransaction, 881
 - processBrokerError, 881
 - processBrokerInfo, 881
 - processCommitTransactionOnePhase, 881
 - processCommitTransactionTwoPhase, 881
 - processConnectionControl, 881
 - processConnectionError, 881
 - processConnectionInfo, 882
 - processConsumerControl, 882
 - processConsumerInfo, 882
 - processControlCommand, 882
 - processDestinationInfo, 882
 - processEndTransaction, 882
 - processFlushCommand, 882
 - processForgetTransaction, 882
 - processKeepAliveInfo, 882
 - processMessage, 883
 - processMessageAck, 883

- processMessageDispatch, 883
- processMessageDispatchNotification, 883
- processMessagePull, 883
- processPrepareTransaction, 883
- processProducerAck, 883
- processProducerInfo, 883
- processRecoverTransactions, 883
- processRemoveConnection, 883
- processRemoveConsumer, 884
- processRemoveDestination, 884
- processRemoveInfo, 884
- processRemoveProducer, 884
- processRemoveSession, 884
- processRemoveSubscriptionInfo, 884
- processReplayCommand, 884
- processResponse, 884
- processRollbackTransaction, 884
- processSessionInfo, 885
- processShutdownInfo, 885
- processTransactionInfo, 885
- processWireFormat, 885
- activemq::state::ConnectionState, 982
 - ~ConnectionState, 983
 - ConnectionState, 983
 - addSession, 983
 - addTempDestination, 983
 - addTransactionState, 983
 - checkShutdown, 983
 - getInfo, 983
 - getRecoveringPullConsumers, 983
 - getSessionState, 983
 - getSessionStates, 983
 - getTempDestinations, 983
 - getTransactionState, 983
 - getTransactionStates, 983
 - isConnectionInterruptProcessing-Complete, 984
 - removeSession, 984
 - removeTempDestination, 984
 - removeTransactionState, 984
 - reset, 984
 - setConnectionInterruptProcessing-Complete, 984
 - shutdown, 984
 - toString, 984
- activemq::state::ConnectionStateTracker, 984
 - ~ConnectionStateTracker, 986
- ConnectionStateTracker, 986
- RemoveTransactionAction, 990
- connectionInterruptProcessing-Complete, 986
- getMaxCacheSize, 986
- isRestoreConsumers, 986
- isRestoreProducers, 986
- isRestoreSessions, 986
- isRestoreTransaction, 986
- isTrackMessages, 986
- isTrackTransactionProducers, 986
- isTrackTransactions, 986
- processBeginTransaction, 986
- processCommitTransactionOne-Phase, 987
- processCommitTransactionTwo-Phase, 987
- processConnectionInfo, 987
- processConsumerInfo, 987
- processDestinationInfo, 987
- processEndTransaction, 987
- processMessage, 987
- processMessageAck, 988
- processPrepareTransaction, 988
- processProducerInfo, 988
- processRemoveConnection, 988
- processRemoveConsumer, 988
- processRemoveDestination, 988
- processRemoveProducer, 988
- processRemoveSession, 989
- processRollbackTransaction, 989
- processSessionInfo, 989
- restore, 989
- setMaxCacheSize, 989
- setRestoreConsumers, 989
- setRestoreProducers, 989
- setRestoreSessions, 989
- setRestoreTransaction, 989
- setTrackMessages, 989
- setTrackTransactionProducers, 989
- setTrackTransactions, 990
- track, 990
- trackBack, 990
- transportInterrupted, 990
- activemq::state::ConsumerState, 1022
 - ~ConsumerState, 1022
 - ConsumerState, 1022
 - getInfo, 1022
 - toString, 1022
- activemq::state::ProducerState, 2199

- ~ProducerState, 2199
- ProducerState, 2199
- getInfo, 2199
- getTransactionState, 2199
- setTransactionState, 2199
- toString, 2199
- activemq::state::SessionState, 2395
 - ~SessionState, 2396
 - SessionState, 2396
 - addConsumer, 2396
 - addProducer, 2396
 - checkShutdown, 2396
 - getConsumerState, 2396
 - getConsumerStates, 2396
 - getInfo, 2396
 - getProducerState, 2397
 - getProducerStates, 2397
 - removeConsumer, 2397
 - removeProducer, 2397
 - shutdown, 2397
 - toString, 2397
- activemq::state::Tracked, 2765
 - ~Tracked, 2766
 - Tracked, 2766
 - isWaitingForResponse, 2766
 - onResponse, 2766
- activemq::state::TransactionState, 2785
 - ~TransactionState, 2785
 - TransactionState, 2785
 - addCommand, 2785
 - addProducerState, 2785
 - checkShutdown, 2785
 - getCommands, 2785
 - getId, 2785
 - getPreparedResult, 2786
 - getProducerStates, 2786
 - isPrepared, 2786
 - setPrepared, 2786
 - setPreparedResult, 2786
 - shutdown, 2786
 - toString, 2786
- activemq::threads, 71
- activemq::threads::CompositeTask, 892
 - ~CompositeTask, 893
 - isPending, 893
- activemq::threads::CompositeTask-
Runner, 893
 - ~CompositeTaskRunner, 894
 - CompositeTaskRunner, 894
 - addTask, 894
 - iterate, 895
 - removeTask, 895
 - run, 895
 - shutdown, 895
 - wakeup, 896
- activemq::threads::DedicatedTaskRunner, 1144
 - ~DedicatedTaskRunner, 1145
 - DedicatedTaskRunner, 1145
 - run, 1145
 - shutdown, 1145
 - wakeup, 1145
- activemq::threads::Scheduler, 2317
 - ~Scheduler, 2318
 - Scheduler, 2318
 - cancel, 2318
 - doStart, 2318
 - doStop, 2318
 - executeAfterDelay, 2319
 - executePeriodically, 2319
 - scheduledPeriodically, 2319
 - shutdown, 2319
- activemq::threads::SchedulerTimerTask, 2319
 - ~SchedulerTimerTask, 2320
 - SchedulerTimerTask, 2320
 - run, 2320
- activemq::threads::Task, 2676
 - ~Task, 2676
 - iterate, 2676
- activemq::threads::TaskRunner, 2677
 - ~TaskRunner, 2677
 - shutdown, 2677, 2678
 - wakeup, 2678
- activemq::transport, 71
- activemq::transport::AbstractTransport-
Factory, 154
 - ~AbstractTransportFactory, 155
 - createWireFormat, 155
- activemq::transport::CompositeTransport, 896
 - ~CompositeTransport, 897
 - addURI, 897
 - removeURI, 897
- activemq::transport::DefaultTransport-
Listener, 1178
 - ~DefaultTransportListener, 1178
 - onCommand, 1179
 - onException, 1179
 - transportInterrupted, 1179

- transportResumed, 1179
- activemq::transport::IOTransport, 1548
 - ~IOTransport, 1550
 - IOTransport, 1550
 - close, 1551
 - getRemoteAddress, 1551
 - getTransportListener, 1551
 - getWireFormat, 1551
 - isClosed, 1551
 - isConnected, 1552
 - isFaultTolerant, 1552
 - isReconnectSupported, 1552
 - isUpdateURIsSupported, 1552
 - narrow, 1552
 - oneway, 1553
 - reconnect, 1553
 - request, 1553, 1554
 - run, 1554
 - setInputStream, 1554
 - setOutputStream, 1555
 - setTransportListener, 1555
 - setWireFormat, 1555
 - start, 1555
 - stop, 1556
 - updateURIs, 1556
- activemq::transport::Transport, 2790
 - ~Transport, 2792
 - getRemoteAddress, 2792
 - getTransportListener, 2792
 - getWireFormat, 2792
 - isClosed, 2793
 - isConnected, 2793
 - isFaultTolerant, 2793
 - isReconnectSupported, 2794
 - isUpdateURIsSupported, 2794
 - narrow, 2794
 - oneway, 2794
 - reconnect, 2795
 - request, 2795, 2796
 - setTransportListener, 2796
 - setWireFormat, 2797
 - start, 2797
 - stop, 2797
 - updateURIs, 2798
- activemq::transport::TransportFactory, 2798
 - ~TransportFactory, 2799
 - create, 2799
 - createComposite, 2799
- activemq::transport::TransportFilter, 2800
 - ~TransportFilter, 2802
 - TransportFilter, 2802
 - close, 2802
 - fire, 2803
 - getRemoteAddress, 2803
 - getTransportListener, 2803
 - getWireFormat, 2803
 - isClosed, 2804
 - isConnected, 2804
 - isFaultTolerant, 2804
 - isReconnectSupported, 2805
 - isUpdateURIsSupported, 2805
 - listener, 2810
 - narrow, 2805
 - next, 2810
 - onCommand, 2805
 - onException, 2806
 - oneway, 2806
 - reconnect, 2806
 - request, 2807
 - setTransportListener, 2808
 - setWireFormat, 2808
 - start, 2808
 - stop, 2809
 - transportInterrupted, 2809
 - transportResumed, 2809
 - updateURIs, 2809
- activemq::transport::TransportListener, 2810
 - ~TransportListener, 2811
 - onCommand, 2811
 - onException, 2811
 - transportInterrupted, 2811
 - transportResumed, 2812
- activemq::transport::TransportRegistry, 2812
 - ~TransportRegistry, 2813
 - findFactory, 2813
 - getInstance, 2813
 - getTransportNames, 2814
 - registerFactory, 2814
 - unregisterAllFactories, 2814
 - unregisterFactory, 2814
- activemq::transport::correlator, 72
- activemq::transport::correlator::FutureResponse, 1393
 - ~FutureResponse, 1393
 - FutureResponse, 1393
 - getResponse, 1393, 1394
 - setResponse, 1394

- activemq::transport::correlator::Response-
Correlator, 2303
 - ~ResponseCorrelator, 2304
 - ResponseCorrelator, 2304
 - close, 2304
 - onCommand, 2305
 - onTransportException, 2305
 - oneway, 2305
 - request, 2306
 - start, 2307
- activemq::transport::failover, 72
- activemq::transport::failover::Backup-
Transport, 486
 - ~BackupTransport, 487
 - BackupTransport, 487
 - getTransport, 487
 - getUri, 487
 - isClosed, 487
 - onException, 488
 - setClosed, 488
 - setTransport, 488
 - setUri, 488
- activemq::transport::failover::Backup-
TransportPool, 489
 - ~BackupTransportPool, 490
 - BackupTransport, 491
 - BackupTransportPool, 490
 - getBackup, 490
 - getBackupPoolSize, 490
 - isEnabled, 490
 - isPending, 490
 - iterate, 491
 - setBackupPoolSize, 491
 - setEnabled, 491
- activemq::transport::failover::Close-
TransportsTask, 818
 - ~CloseTransportsTask, 818
 - CloseTransportsTask, 818
 - add, 818
 - isPending, 818
 - iterate, 818
- activemq::transport::failover::Failover-
Transport, 1305
 - ~FailoverTransport, 1308
 - FailoverTransport, 1308
 - FailoverTransportListener, 1318
 - add, 1308
 - addURI, 1308
 - close, 1308
 - getBackOffMultiplier, 1308
 - getBackupPoolSize, 1309
 - getInitialReconnectDelay, 1309
 - getMaxCacheSize, 1309
 - getMaxReconnectAttempts, 1309
 - getMaxReconnectDelay, 1309
 - getReconnectDelay, 1309
 - getRemoteAddress, 1309
 - getStartupMaxReconnectAttempts, 1309
 - getTimeout, 1309
 - getTransportListener, 1309
 - getWireFormat, 1309
 - handleConnectionControl, 1310
 - handleTransportFailure, 1310
 - isBackup, 1310
 - isClosed, 1310
 - isConnected, 1311
 - isFaultTolerant, 1311
 - isInitialized, 1311
 - isPending, 1311
 - isRandomize, 1311
 - isReconnectSupported, 1312
 - isTrackMessages, 1312
 - isTrackTransactionProducers, 1312
 - isUpdateURIsSupported, 1312
 - isUseExponentialBackOff, 1312
 - iterate, 1312
 - narrow, 1312
 - oneway, 1313
 - reconnect, 1313
 - removeURI, 1314
 - request, 1314
 - restoreTransport, 1315
 - setBackOffMultiplier, 1315
 - setBackup, 1315
 - setBackupPoolSize, 1315
 - setConnectionInterruptProcessing-
Complete, 1315
 - setInitialReconnectDelay, 1316
 - setInitialized, 1316
 - setMaxCacheSize, 1316
 - setMaxReconnectAttempts, 1316
 - setMaxReconnectDelay, 1316
 - setRandomize, 1316
 - setReconnectDelay, 1316
 - setReconnectSupported, 1316
 - setStartupMaxReconnectAttempts, 1316
 - setTimeout, 1316
 - setTrackMessages, 1316

- setTrackTransactionProducers, 1316
- setTransportListener, 1316
- setUpdateURIsSupported, 1317
- setUseExponentialBackOff, 1317
- setWireFormat, 1317
- start, 1317
- stop, 1317
- updateURIs, 1317
- activemq::transport::failover::FailoverTransportFactory, 1318
 - ~FailoverTransportFactory, 1319
 - create, 1319
 - createComposite, 1319
 - doCreateComposite, 1320
- activemq::transport::failover::FailoverTransportListener, 1320
 - ~FailoverTransportListener, 1321
 - FailoverTransportListener, 1321
 - onCommand, 1321
 - onException, 1322
 - transportInterrupted, 1322
 - transportResumed, 1322
- activemq::transport::failover::URIPool, 2851
 - ~URIPool, 2852
 - URIPool, 2852
 - addURI, 2852
 - addURIs, 2852
 - getURI, 2853
 - isRandomize, 2853
 - removeURI, 2853
 - setRandomize, 2853
- activemq::transport::inactivity, 73
- activemq::transport::inactivity::InactivityMonitor, 1425
 - ~InactivityMonitor, 1426
 - AsyncSignalReadErrorTask, 1428
 - AsyncWriteTask, 1428
 - InactivityMonitor, 1426
 - ReadChecker, 1428
 - WriteChecker, 1428
 - close, 1427
 - getInitialDelayTime, 1427
 - getReadCheckTime, 1427
 - getWriteCheckTime, 1427
 - isKeepAliveResponseRequired, 1427
 - onCommand, 1427
 - onException, 1428
 - oneway, 1427
 - setInitialDelayTime, 1428
 - setKeepAliveResponseRequired, 1428
 - setReadCheckTime, 1428
 - setWriteCheckTime, 1428
- activemq::transport::inactivity::ReadChecker, 2236
 - ~ReadChecker, 2237
 - ReadChecker, 2237
 - run, 2237
- activemq::transport::inactivity::WriteChecker, 2907
 - ~WriteChecker, 2908
 - WriteChecker, 2908
 - run, 2908
- activemq::transport::logging, 73
- activemq::transport::logging::LoggingTransport, 1709
 - ~LoggingTransport, 1710
 - LoggingTransport, 1710
 - onCommand, 1710
 - oneway, 1710
 - request, 1711
- activemq::transport::mock, 73
- activemq::transport::mock::InternalCommandListener, 1527
 - ~InternalCommandListener, 1528
 - InternalCommandListener, 1528
 - onCommand, 1528
 - run, 1528
 - setResponseBuilder, 1528
 - setTransport, 1528
- activemq::transport::mock::MockTransport, 1948
 - ~MockTransport, 1950
 - MockTransport, 1950
 - close, 1950
 - fireCommand, 1951
 - fireException, 1951
 - getInstance, 1951
 - getName, 1951
 - getNumReceivedMessageBeforeFail, 1951
 - getNumReceivedMessages, 1951
 - getNumSentKeepAlives, 1951
 - getNumSentKeepAlivesBeforeFail, 1952
 - getNumSentMessageBeforeFail, 1952
 - getNumSentMessages, 1952

- getRemoteAddress, 1952
- getTransportListener, 1952
- getWireFormat, 1952
- isClosed, 1952
- isConnected, 1953
- isFailOnClose, 1953
- isFailOnKeepAliveSends, 1953
- isFailOnReceiveMessage, 1953
- isFailOnSendMessage, 1953
- isFailOnStart, 1953
- isFailOnStop, 1953
- isFaultTolerant, 1953
- isReconnectSupported, 1953
- isUpdateURIsSupported, 1954
- narrow, 1954
- oneway, 1954
- reconnect, 1955
- request, 1955
- setFailOnClose, 1956
- setFailOnKeepAliveSends, 1956
- setFailOnReceiveMessage, 1956
- setFailOnSendMessage, 1956
- setFailOnStart, 1956
- setFailOnStop, 1956
- setName, 1956
- setNumReceivedMessageBeforeFail, 1956
- setNumReceivedMessages, 1956
- setNumSentKeepAlives, 1956
- setNumSentKeepAlivesBeforeFail, 1956
- setNumSentMessageBeforeFail, 1956
- setNumSentMessages, 1956
- setOutgoingListener, 1957
- setResponseBuilder, 1957
- setTransportListener, 1957
- setWireFormat, 1957
- start, 1957
- stop, 1958
- updateURIs, 1958
- activemq::transport::mock::MockTransportFactory, 1958
 - ~MockTransportFactory, 1959
 - create, 1959
 - createComposite, 1959
 - doCreateComposite, 1960
- activemq::transport::mock::ResponseBuilder, 2301
 - ~ResponseBuilder, 2302
- buildIncomingCommands, 2302
- buildResponse, 2302
- activemq::transport::tcp, 73
- activemq::transport::tcp::SslTransport, 2525
 - ~SslTransport, 2526
 - SslTransport, 2526
 - configureSocket, 2526
 - createSocket, 2526
- activemq::transport::tcp::SslTransportFactory, 2527
 - ~SslTransportFactory, 2527
 - doCreateComposite, 2527
- activemq::transport::tcp::TcpTransport, 2693
 - ~TcpTransport, 2694
 - TcpTransport, 2694
 - close, 2694
 - configureSocket, 2694
 - connect, 2695
 - createSocket, 2695
 - isClosed, 2696
 - isConnected, 2696
 - isFaultTolerant, 2696
- activemq::transport::tcp::TcpTransportFactory, 2696
 - ~TcpTransportFactory, 2697
 - create, 2697
 - createComposite, 2698
 - doCreateComposite, 2698
- activemq::util, 74
- activemq::util::ActiveMQProperties, 316
 - ~ActiveMQProperties, 317
 - ActiveMQProperties, 317
 - clear, 317
 - clone, 317
 - copy, 317
 - getProperties, 318
 - getProperty, 318
 - hasProperty, 318
 - isEmpty, 319
 - propertyNames, 319
 - remove, 319
 - setProperties, 320
 - setProperty, 320
 - size, 320
 - toArray, 320
 - toString, 320
- activemq::util::CMSExceptionSupport, 829

- ~CMSExceptionSupport, 829
- create, 829
- createMessageEOFException, 829
- createMessageFormatException, 830
- activemq::util::CompositeData, 890
 - ~CompositeData, 891
 - CompositeData, 891
 - getComponents, 891
 - getFragment, 891
 - getHost, 891
 - getParameters, 891
 - getPath, 891
 - getScheme, 891
 - setComponents, 891
 - setFragment, 891
 - setHost, 891
 - setParameters, 892
 - setPath, 892
 - setScheme, 892
 - toURI, 892
- activemq::util::IdGenerator, 1411
 - ~IdGenerator, 1412
 - IdGenerator, 1412
 - activemq::library::ActiveMQCPP, 1413
 - compare, 1412
 - generateId, 1412
 - getHostname, 1412
 - getSeedFromId, 1412
 - getSequenceFromId, 1413
- activemq::util::LongSequenceGenerator, 1765
 - ~LongSequenceGenerator, 1765
 - LongSequenceGenerator, 1765
 - getLastSequenceId, 1765
 - getNextSequenceId, 1765
- activemq::util::MarshallingSupport, 1796
 - ~MarshallingSupport, 1797
 - MarshallingSupport, 1797
 - asciiToModifiedUtf8, 1797
 - modifiedUtf8ToAscii, 1798
 - readString16, 1798
 - readString32, 1799
 - writeString, 1799
 - writeString16, 1800
 - writeString32, 1800
- activemq::util::MemoryUsage, 1818
 - ~MemoryUsage, 1819
 - MemoryUsage, 1819
 - decreaseUsage, 1819
 - enqueueUsage, 1819
 - getLimit, 1820
 - getUsage, 1820
 - increaseUsage, 1820
 - isFull, 1820
 - setLimit, 1820
 - setUsage, 1821
 - waitForSpace, 1821
- activemq::util::PrimitiveList, 2114
 - ~PrimitiveList, 2116
 - PrimitiveList, 2116
 - getBool, 2117
 - getByte, 2117
 - getByteArray, 2117
 - getChar, 2118
 - getDouble, 2118
 - getFloat, 2119
 - getInt, 2119
 - getLong, 2120
 - getShort, 2120
 - getString, 2121
 - setBool, 2121
 - setByte, 2121
 - setByteArray, 2122
 - setChar, 2122
 - setDouble, 2123
 - setFloat, 2123
 - setInt, 2123
 - setLong, 2124
 - setShort, 2124
 - setString, 2124
 - toString, 2125
- activemq::util::PrimitiveMap, 2125
 - ~PrimitiveMap, 2127
 - PrimitiveMap, 2127
 - getBool, 2128
 - getByte, 2128
 - getByteArray, 2129
 - getChar, 2129
 - getDouble, 2129
 - getFloat, 2130
 - getInt, 2130
 - getLong, 2131
 - getShort, 2131
 - getString, 2132
 - setBool, 2132
 - setByte, 2133
 - setByteArray, 2133
 - setChar, 2133

- setDouble, 2133
- setFloat, 2134
- setInt, 2134
- setLong, 2134
- setShort, 2134
- setString, 2135
- toString, 2135
- activemq::util::PrimitiveValueConverter, 2144
 - ~PrimitiveValueConverter, 2144
 - PrimitiveValueConverter, 2144
 - convert, 2145
- activemq::util::PrimitiveValueNode, 2145
 - ~PrimitiveValueNode, 2151
 - BIG_STRING_TYPE, 2148
 - BOOLEAN_TYPE, 2148
 - BYTE_ARRAY_TYPE, 2148
 - BYTE_TYPE, 2148
 - CHAR_TYPE, 2148
 - DOUBLE_TYPE, 2148
 - FLOAT_TYPE, 2148
 - INTEGER_TYPE, 2148
 - LIST_TYPE, 2148
 - LONG_TYPE, 2148
 - MAP_TYPE, 2148
 - NULL_TYPE, 2148
 - PrimitiveType, 2148
 - PrimitiveValueNode, 2149–2151
 - SHORT_TYPE, 2148
 - STRING_TYPE, 2148
 - clear, 2151
 - getBool, 2152
 - getByte, 2152
 - getByteArray, 2152
 - getChar, 2152
 - getDouble, 2153
 - getFloat, 2153
 - getInt, 2153
 - getList, 2154
 - getLong, 2154
 - getMap, 2154
 - getShort, 2155
 - getString, 2155
 - getType, 2155
 - getValue, 2155
 - operator=, 2156
 - operator==, 2156
 - setBool, 2156
 - setByte, 2156
 - setByteArray, 2156
 - setChar, 2157
 - setDouble, 2157
 - setFloat, 2157
 - setInt, 2157
 - setList, 2157
 - setLong, 2158
 - setMap, 2158
 - setShort, 2158
 - setString, 2158
 - setValue, 2159
 - toString, 2159
- activemq::util::PrimitiveValueNode::-
 - PrimitiveValue, 2142
 - boolValue, 2143
 - byteArrayValue, 2143
 - byteValue, 2143
 - charValue, 2143
 - doubleValue, 2143
 - floatValue, 2143
 - intValue, 2143
 - listValue, 2143
 - longValue, 2143
 - mapValue, 2143
 - shortValue, 2143
 - stringValue, 2143
- activemq::util::Service, 2355
 - ~Service, 2356
 - start, 2356
 - stop, 2356
- activemq::util::ServiceListener, 2356
 - ~ServiceListener, 2357
 - started, 2357
 - stopped, 2357
- activemq::util::ServiceStopper, 2358
 - ~ServiceStopper, 2358
 - ServiceStopper, 2358
 - onException, 2358
 - stop, 2358
 - throwFirstException, 2358
- activemq::util::ServiceSupport, 2358
 - ~ServiceSupport, 2360
 - ServiceSupport, 2360
 - addServiceListener, 2360
 - dispose, 2360
 - doStart, 2360
 - doStop, 2360
 - isStarted, 2360
 - isStopped, 2361
 - isStopping, 2361
 - operator=, 2361

- removeServiceListener, 2361
- start, 2361
- stop, 2361
- activemq::util::URISupport, 2854
 - createQueryString, 2854
 - parseComposite, 2855
 - parseQuery, 2855, 2856
 - parseURL, 2856
- activemq::util::Usage, 2872
 - ~Usage, 2873
 - decreaseUsage, 2873
 - enqueueUsage, 2873
 - increaseUsage, 2873
 - isFull, 2873
 - waitForSpace, 2874
- activemq::wireformat, 75
- activemq::wireformat::MarshalAware, 1792
 - ~MarshalAware, 1793
 - afterMarshal, 1793
 - afterUnmarshal, 1793
 - beforeMarshal, 1793
 - beforeUnmarshal, 1794
 - getMarshaledForm, 1794
 - isMarshalAware, 1795
 - setMarshaledForm, 1795
- activemq::wireformat::WireFormat, 2884
 - ~WireFormat, 2885
 - createNegotiator, 2885
 - getVersion, 2886
 - hasNegotiator, 2886
 - inReceive, 2886
 - marshal, 2887
 - setVersion, 2887
 - unmarshal, 2887
- activemq::wireformat::WireFormatFactory, 2888
 - ~WireFormatFactory, 2889
 - createWireFormat, 2889
- activemq::wireformat::WireFormatNegotiator, 2904
 - ~WireFormatNegotiator, 2904
 - WireFormatNegotiator, 2904
- activemq::wireformat::WireFormatRegistry, 2905
 - ~WireFormatRegistry, 2906
 - findFactory, 2906
 - getInstance, 2906
 - getWireFormatNames, 2906
 - registerFactory, 2906
 - unregisterAllFactories, 2907
 - unregisterFactory, 2907
- activemq::wireformat::openwire, 75
- activemq::wireformat::openwire::OpenWireFormat, 2045
 - ~OpenWireFormat, 2048
 - DEFAULT_VERSION, 2058
 - MAX_SUPPORTED_VERSION, 2058
 - NULL_TYPE, 2059
 - OpenWireFormat, 2048
 - addMarshaller, 2048
 - createNegotiator, 2049
 - destroyMarshallers, 2049
 - doUnmarshal, 2049
 - getCacheSize, 2050
 - getMaxInactivityDuration, 2050
 - getMaxInactivityDurationInitialDelay, 2050
 - getPreferredWireFormatInfo, 2050
 - getVersion, 2051
 - hasNegotiator, 2051
 - inReceive, 2051
 - isCacheEnabled, 2051
 - isSizePrefixDisabled, 2052
 - isStackTraceEnabled, 2052
 - isTcpNoDelayEnabled, 2052
 - isTightEncodingEnabled, 2052
 - looseMarshalNestedObject, 2052
 - looseUnmarshalNestedObject, 2053
 - marshal, 2053
 - renegotiateWireFormat, 2054
 - setCacheEnabled, 2054
 - setCacheSize, 2054
 - setMaxInactivityDuration, 2054
 - setMaxInactivityDurationInitialDelay, 2055
 - setPreferredWireFormatInfo, 2055
 - setSizePrefixDisabled, 2055
 - setStackTraceEnabled, 2055
 - setTcpNoDelayEnabled, 2056
 - setTightEncodingEnabled, 2056
 - setVersion, 2056
 - tightMarshalNestedObject1, 2057
 - tightMarshalNestedObject2, 2057
 - tightUnmarshalNestedObject, 2057
 - unmarshal, 2058
- activemq::wireformat::openwire::OpenWireFormatFactory, 2059
 - ~OpenWireFormatFactory, 2059

- OpenWireFormatFactory, 2059
- createWireFormat, 2060
- activemq::wireformat::openwire::Open-
WireFormatNegotiator, 2060
- ~OpenWireFormatNegotiator, 2061
- OpenWireFormatNegotiator, 2061
- close, 2061
- onCommand, 2062
- onTransportException, 2063
- oneway, 2062
- request, 2063
- start, 2064
- activemq::wireformat::openwire::Open-
WireResponseBuilder, 2064
- ~OpenWireResponseBuilder, 2065
- OpenWireResponseBuilder, 2065
- buildIncomingCommands, 2065
- buildResponse, 2066
- activemq::wireformat::openwire::marshal,
75
- activemq::wireformat::openwire::marshal-
::BaseDataStreamMarshaller,
507
- ~BaseDataStreamMarshaller, 511
- looseMarshal, 511
- looseMarshalBrokerError, 511
- looseMarshalCachedObject, 511
- looseMarshalLong, 512
- looseMarshalNestedObject, 512
- looseMarshalObjectArray, 513
- looseMarshalString, 513
- looseUnmarshal, 513
- looseUnmarshalBrokerError, 514
- looseUnmarshalByteArray, 514
- looseUnmarshalCachedObject, 515
- looseUnmarshalConstByteArray, 515
- looseUnmarshalLong, 516
- looseUnmarshalNestedObject, 516
- looseUnmarshalString, 517
- readAsciiString, 517
- tightMarshal1, 517
- tightMarshal2, 518
- tightMarshalBrokerError1, 518
- tightMarshalBrokerError2, 518
- tightMarshalCachedObject1, 519
- tightMarshalCachedObject2, 519
- tightMarshalLong1, 520
- tightMarshalLong2, 520
- tightMarshalNestedObject1, 521
- tightMarshalNestedObject2, 521
- tightMarshalObjectArray1, 522
- tightMarshalObjectArray2, 522
- tightMarshalString1, 523
- tightMarshalString2, 523
- tightUnmarshal, 524
- tightUnmarshalBrokerError, 524
- tightUnmarshalByteArray, 524
- tightUnmarshalCachedObject, 525
- tightUnmarshalConstByteArray, 525
- tightUnmarshalLong, 526
- tightUnmarshalNestedObject, 526
- tightUnmarshalString, 527
- toHexFromBytes, 527
- toString, 527, 528
- activemq::wireformat::openwire::marshal-
::DataStreamMarshaller, 1119
- ~DataStreamMarshaller, 1120
- createObject, 1120
- getDataStructureType, 1122
- looseMarshal, 1123
- looseUnmarshal, 1125
- tightMarshal1, 1127
- tightMarshal2, 1129
- tightUnmarshal, 1131
- activemq::wireformat::openwire::marshal-
::PrimitiveTypesMarshaller,
2135
- ~PrimitiveTypesMarshaller, 2137
- PrimitiveTypesMarshaller, 2137
- marshal, 2137
- marshalList, 2138
- marshalMap, 2138
- marshalPrimitive, 2138
- marshalPrimitiveList, 2139
- marshalPrimitiveMap, 2139
- unmarshal, 2139, 2140
- unmarshalList, 2140
- unmarshalMap, 2141
- unmarshalPrimitive, 2141
- unmarshalPrimitiveList, 2141
- unmarshalPrimitiveMap, 2142
- activemq::wireformat::openwire::marshal-
::generated, 76
- activemq::wireformat::openwire::marshal-
::generated::ActiveMQBlob-
MessageMarshaller, 161
- ~ActiveMQBlobMessageMarshaller,
162
- ActiveMQBlobMessageMarshaller,
162

- createObject, 163
- getDataStructureType, 163
- looseMarshal, 163
- looseUnmarshal, 164
- tightMarshal1, 164
- tightMarshal2, 164
- tightUnmarshal, 165
- activemq::wireformat::openwire::marshal-
::generated::ActiveMQBytes-
MessageMarshaller, 183
- ~ActiveMQBytesMessageMarshaller,
184
- ActiveMQBytesMessageMarshaller,
184
- createObject, 184
- getDataStructureType, 185
- looseMarshal, 185
- looseUnmarshal, 185
- tightMarshal1, 186
- tightMarshal2, 186
- tightUnmarshal, 187
- activemq::wireformat::openwire::marshal-
::generated::ActiveMQDestination-activemq::wireformat::openwire::marshal-
Marshaller, 258
- ~ActiveMQDestinationMarshaller,
259
- ActiveMQDestinationMarshaller, 258
- looseMarshal, 259
- looseUnmarshal, 259
- tightMarshal1, 260
- tightMarshal2, 260
- tightUnmarshal, 261
- activemq::wireformat::openwire::marshal-
::generated::ActiveMQMap-
MessageMarshaller, 284
- ~ActiveMQMapMessageMarshaller,
285
- ActiveMQMapMessageMarshaller,
285
- createObject, 285
- getDataStructureType, 285
- looseMarshal, 285
- looseUnmarshal, 286
- tightMarshal1, 286
- tightMarshal2, 287
- tightUnmarshal, 287
- activemq::wireformat::openwire::marshal-
::generated::ActiveMQMessage-
Marshaller, 291
- ~ActiveMQMessageMarshaller, 292
- ActiveMQMessageMarshaller, 292
- createObject, 292
- getDataStructureType, 292
- looseMarshal, 292
- looseUnmarshal, 293
- tightMarshal1, 293
- tightMarshal2, 293
- tightUnmarshal, 294
- activemq::wireformat::openwire::marshal-
::generated::ActiveMQObject-
MessageMarshaller, 304
- ~ActiveMQObjectMessageMarshaller,
305
- ActiveMQObjectMessageMarshaller,
305
- createObject, 305
- getDataStructureType, 305
- looseMarshal, 305
- looseUnmarshal, 306
- tightMarshal1, 306
- tightMarshal2, 306
- tightUnmarshal, 307
- activemq::wireformat::openwire::marshal-
::generated::ActiveMQQueue-
Marshaller, 329
- ~ActiveMQQueueMarshaller, 330
- ActiveMQQueueMarshaller, 330
- createObject, 330
- getDataStructureType, 330
- looseMarshal, 330
- looseUnmarshal, 331
- tightMarshal1, 331
- tightMarshal2, 332
- tightUnmarshal, 332
- activemq::wireformat::openwire::marshal-
::generated::ActiveMQStream-
MessageMarshaller, 375
- ~ActiveMQStreamMessageMarshaller,
376
- ActiveMQStreamMessageMarshaller,
376
- createObject, 376
- getDataStructureType, 376
- looseMarshal, 376
- looseUnmarshal, 377
- tightMarshal1, 377
- tightMarshal2, 377
- tightUnmarshal, 378
- activemq::wireformat::openwire::marshal-
::generated::ActiveMQTemp-

- DestinationMarshaller, 382
- ~ActiveMQTempDestinationMarshaller,activemq::wireformat::openwire::marshal-383
- ActiveMQTempDestinationMarshaller, 383
- looseMarshal, 383
- looseUnmarshal, 384
- tightMarshal1, 384
- tightMarshal2, 385
- tightUnmarshal, 385
- activemq::wireformat::openwire::marshal-
::generated::ActiveMQTemp-
QueueMarshaller, 391
- ~ActiveMQTempQueueMarshaller, 392
- ActiveMQTempQueueMarshaller, 392
- createObject, 393
- getDataStructureType, 393
- looseMarshal, 393
- looseUnmarshal, 394
- tightMarshal1, 394
- tightMarshal2, 394
- tightUnmarshal, 395
- activemq::wireformat::openwire::marshal-
::generated::ActiveMQTemp-
TopicMarshaller, 401
- ~ActiveMQTempTopicMarshaller, 402
- ActiveMQTempTopicMarshaller, 402
- createObject, 402
- getDataStructureType, 402
- looseMarshal, 402
- looseUnmarshal, 403
- tightMarshal1, 403
- tightMarshal2, 404
- tightUnmarshal, 404
- activemq::wireformat::openwire::marshal-
::generated::ActiveMQText-
MessageMarshaller, 410
- ~ActiveMQTextMessageMarshaller, 411
- ActiveMQTextMessageMarshaller, 411
- createObject, 411
- getDataStructureType, 411
- looseMarshal, 412
- looseUnmarshal, 412
- tightMarshal1, 413
- tightMarshal2, 413
- tightUnmarshal, 413
- activemq::wireformat::openwire::marshal-
::generated::ActiveMQTopic-
Marshaller, 419
- ~ActiveMQTopicMarshaller, 420
- ActiveMQTopicMarshaller, 420
- createObject, 420
- getDataStructureType, 420
- looseMarshal, 420
- looseUnmarshal, 421
- tightMarshal1, 421
- tightMarshal2, 422
- tightUnmarshal, 422
- activemq::wireformat::openwire::marshal-
::generated::BaseCommand-
Marshaller, 499
- ~BaseCommandMarshaller, 500
- BaseCommandMarshaller, 500
- looseMarshal, 500
- looseUnmarshal, 501
- tightMarshal1, 502
- tightMarshal2, 504
- tightUnmarshal, 505
- activemq::wireformat::openwire::marshal-
::generated::BrokerIdMarshaller, 567
- ~BrokerIdMarshaller, 568
- BrokerIdMarshaller, 568
- createObject, 569
- getDataStructureType, 569
- looseMarshal, 569
- looseUnmarshal, 570
- tightMarshal1, 570
- tightMarshal2, 570
- tightUnmarshal, 571
- activemq::wireformat::openwire::marshal-
::generated::BrokerInfoMarshaller, 578
- ~BrokerInfoMarshaller, 579
- BrokerInfoMarshaller, 579
- createObject, 579
- getDataStructureType, 580
- looseMarshal, 580
- looseUnmarshal, 580
- tightMarshal1, 581
- tightMarshal2, 581
- tightUnmarshal, 582
- activemq::wireformat::openwire::marshal-
::generated::Connection-
ControlMarshaller, 943

- ~ConnectionControlMarshaller, 944
- ConnectionControlMarshaller, 944
- createObject, 945
- getDataStructureType, 945
- looseMarshal, 945
- looseUnmarshal, 946
- tightMarshal1, 946
- tightMarshal2, 946
- tightUnmarshal, 947
- activemq::wireformat::openwire::marshal-
::generated::ConnectionError-
Marshaller, 951
- ~ConnectionErrorMarshaller, 952
- ConnectionErrorMarshaller, 952
- createObject, 952
- getDataStructureType, 952
- looseMarshal, 953
- looseUnmarshal, 953
- tightMarshal1, 954
- tightMarshal2, 954
- tightUnmarshal, 954
- activemq::wireformat::openwire::marshal-
::generated::ConnectionId-
Marshaller, 963
- ~ConnectionIdMarshaller, 964
- ConnectionIdMarshaller, 964
- createObject, 965
- getDataStructureType, 965
- looseMarshal, 965
- looseUnmarshal, 966
- tightMarshal1, 966
- tightMarshal2, 966
- tightUnmarshal, 967
- activemq::wireformat::openwire::marshal-
::generated::ConnectionInfo-
Marshaller, 974
- ~ConnectionInfoMarshaller, 975
- ConnectionInfoMarshaller, 975
- createObject, 975
- getDataStructureType, 975
- looseMarshal, 975
- looseUnmarshal, 976
- tightMarshal1, 976
- tightMarshal2, 977
- tightUnmarshal, 977
- activemq::wireformat::openwire::marshal-
::generated::ConsumerControl-
Marshaller, 996
- ~ConsumerControlMarshaller, 997
- ConsumerControlMarshaller, 997
- createObject, 997
- getDataStructureType, 998
- looseMarshal, 998
- looseUnmarshal, 998
- tightMarshal1, 999
- tightMarshal2, 999
- tightUnmarshal, 1000
- activemq::wireformat::openwire::marshal-
::generated::ConsumerId-
Marshaller, 1005
- ~ConsumerIdMarshaller, 1006
- ConsumerIdMarshaller, 1006
- createObject, 1006
- getDataStructureType, 1006
- looseMarshal, 1006
- looseUnmarshal, 1007
- tightMarshal1, 1007
- tightMarshal2, 1007
- tightUnmarshal, 1008
- activemq::wireformat::openwire::marshal-
::generated::ConsumerInfo-
Marshaller, 1017
- ~ConsumerInfoMarshaller, 1018
- ConsumerInfoMarshaller, 1018
- createObject, 1019
- getDataStructureType, 1019
- looseMarshal, 1019
- looseUnmarshal, 1020
- tightMarshal1, 1020
- tightMarshal2, 1020
- tightUnmarshal, 1021
- activemq::wireformat::openwire::marshal-
::generated::ControlCommand-
Marshaller, 1025
- ~ControlCommandMarshaller, 1026
- ControlCommandMarshaller, 1026
- createObject, 1027
- getDataStructureType, 1027
- looseMarshal, 1027
- looseUnmarshal, 1028
- tightMarshal1, 1028
- tightMarshal2, 1028
- tightUnmarshal, 1029
- activemq::wireformat::openwire::marshal-
::generated::DataArrayResponse-
Marshaller, 1072
- ~DataArrayResponseMarshaller,
1073
- DataArrayResponseMarshaller, 1073
- createObject, 1073

- getDataStructureType, 1073
- looseMarshal, 1073
- looseUnmarshal, 1074
- tightMarshal1, 1074
- tightMarshal2, 1074
- tightUnmarshal, 1075
- activemq::wireformat::openwire::marshal-
::generated::DataResponse-
Marshaller, 1115
- ~DataResponseMarshaller, 1116
- DataResponseMarshaller, 1116
- createObject, 1116
- getDataStructureType, 1116
- looseMarshal, 1117
- looseUnmarshal, 1117
- tightMarshal1, 1118
- tightMarshal2, 1118
- tightUnmarshal, 1118
- activemq::wireformat::openwire::marshal-
::generated::DestinationInfo-
Marshaller, 1218
- ~DestinationInfoMarshaller, 1219
- DestinationInfoMarshaller, 1219
- createObject, 1219
- getDataStructureType, 1219
- looseMarshal, 1219
- looseUnmarshal, 1220
- tightMarshal1, 1220
- tightMarshal2, 1221
- tightUnmarshal, 1221
- activemq::wireformat::openwire::marshal-
::generated::DiscoveryEvent-
Marshaller, 1230
- ~DiscoveryEventMarshaller, 1231
- DiscoveryEventMarshaller, 1231
- createObject, 1231
- getDataStructureType, 1231
- looseMarshal, 1231
- looseUnmarshal, 1232
- tightMarshal1, 1232
- tightMarshal2, 1232
- tightUnmarshal, 1233
- activemq::wireformat::openwire::marshal-
::generated::ExceptionResponse-
Marshaller, 1290
- ~ExceptionResponseMarshaller,
1291
- ExceptionResponseMarshaller, 1291
- createObject, 1291
- getDataStructureType, 1292
- looseMarshal, 1292
- looseUnmarshal, 1292
- tightMarshal1, 1293
- tightMarshal2, 1293
- tightUnmarshal, 1294
- activemq::wireformat::openwire::marshal-
::generated::FlushCommand-
Marshaller, 1383
- ~FlushCommandMarshaller, 1384
- FlushCommandMarshaller, 1384
- createObject, 1384
- getDataStructureType, 1385
- looseMarshal, 1385
- looseUnmarshal, 1385
- tightMarshal1, 1386
- tightMarshal2, 1386
- tightUnmarshal, 1387
- activemq::wireformat::openwire::marshal-
::generated::IntegerResponse-
Marshaller, 1519
- ~IntegerResponseMarshaller, 1520
- IntegerResponseMarshaller, 1520
- createObject, 1520
- getDataStructureType, 1520
- looseMarshal, 1520
- looseUnmarshal, 1521
- tightMarshal1, 1521
- tightMarshal2, 1522
- tightUnmarshal, 1522
- activemq::wireformat::openwire::marshal-
::generated::JournalQueueAck-
Marshaller, 1564
- ~JournalQueueAckMarshaller, 1565
- JournalQueueAckMarshaller, 1565
- createObject, 1565
- getDataStructureType, 1565
- looseMarshal, 1565
- looseUnmarshal, 1566
- tightMarshal1, 1566
- tightMarshal2, 1567
- tightUnmarshal, 1567
- activemq::wireformat::openwire::marshal-
::generated::JournalTopicAck-
Marshaller, 1572
- ~JournalTopicAckMarshaller, 1573
- JournalTopicAckMarshaller, 1573
- createObject, 1574
- getDataStructureType, 1574
- looseMarshal, 1574
- looseUnmarshal, 1575

- tightMarshal1, 1575
- tightMarshal2, 1575
- tightUnmarshal, 1576
- activemq::wireformat::openwire::marshal-
::generated::JournalTrace-
Marshaller, 1579
 - ~JournalTraceMarshaller, 1580
- JournalTraceMarshaller, 1580
- createObject, 1580
- getDataStructureType, 1581
- looseMarshal, 1581
- looseUnmarshal, 1581
- tightMarshal1, 1582
- tightMarshal2, 1582
- tightUnmarshal, 1583
- activemq::wireformat::openwire::marshal-
::generated::JournalTransaction-
Marshaller, 1587
 - ~JournalTransactionMarshaller,
1588
- JournalTransactionMarshaller, 1588
- createObject, 1588
- getDataStructureType, 1588
- looseMarshal, 1589
- looseUnmarshal, 1589
- tightMarshal1, 1589
- tightMarshal2, 1590
- tightUnmarshal, 1590
- activemq::wireformat::openwire::marshal-
::generated::KeepAliveInfo-
Marshaller, 1594
 - ~KeepAliveInfoMarshaller, 1595
- KeepAliveInfoMarshaller, 1595
- createObject, 1595
- getDataStructureType, 1595
- looseMarshal, 1596
- looseUnmarshal, 1596
- tightMarshal1, 1597
- tightMarshal2, 1597
- tightUnmarshal, 1597
- activemq::wireformat::openwire::marshal-
::generated::LastPartialCommand-
Marshaller, 1608
 - ~LastPartialCommandMarshaller,
1609
- LastPartialCommandMarshaller,
1609
- createObject, 1609
- getDataStructureType, 1609
- looseMarshal, 1610
- looseUnmarshal, 1610
- tightMarshal1, 1611
- tightMarshal2, 1611
- tightUnmarshal, 1611
- activemq::wireformat::openwire::marshal-
::generated::LocalTransaction-
IdMarshaller, 1678
 - ~LocalTransactionIdMarshaller,
1679
- LocalTransactionIdMarshaller, 1679
- createObject, 1679
- getDataStructureType, 1680
- looseMarshal, 1680
- looseUnmarshal, 1680
- tightMarshal1, 1681
- tightMarshal2, 1681
- tightUnmarshal, 1682
- activemq::wireformat::openwire::marshal-
::generated::MarshallerFactory,
1795
 - ~MarshallerFactory, 1796
- configure, 1796
- activemq::wireformat::openwire::marshal-
::generated::MessageAck-
Marshaller, 1873
 - ~MessageAckMarshaller, 1874
- MessageAckMarshaller, 1874
- createObject, 1874
- getDataStructureType, 1874
- looseMarshal, 1874
- looseUnmarshal, 1875
- tightMarshal1, 1875
- tightMarshal2, 1875
- tightUnmarshal, 1876
- activemq::wireformat::openwire::marshal-
::generated::MessageDispatch-
Marshaller, 1890
 - ~MessageDispatchMarshaller, 1891
- MessageDispatchMarshaller, 1891
- createObject, 1891
- getDataStructureType, 1892
- looseMarshal, 1892
- looseUnmarshal, 1892
- tightMarshal1, 1893
- tightMarshal2, 1893
- tightUnmarshal, 1894
- activemq::wireformat::openwire::marshal-
::generated::MessageDispatch-
NotificationMarshaller, 1899

- ~MessageDispatchNotification-
Marshaller, 1900
- MessageDispatchNotification-
Marshaller, 1900
- createObject, 1900
- getDataStructureType, 1901
- looseMarshal, 1901
- looseUnmarshal, 1901
- tightMarshal1, 1902
- tightMarshal2, 1902
- tightUnmarshal, 1903
- activemq::wireformat::openwire::marshal-
::generated::MessageIdMarshaller,
1912
 - ~MessageIdMarshaller, 1913
 - MessageIdMarshaller, 1913
 - createObject, 1913
 - getDataStructureType, 1914
 - looseMarshal, 1914
 - looseUnmarshal, 1914
 - tightMarshal1, 1915
 - tightMarshal2, 1915
 - tightUnmarshal, 1916
- activemq::wireformat::openwire::marshal-
::generated::MessageMarshaller,
1917
 - ~MessageMarshaller, 1918
 - MessageMarshaller, 1918
 - looseMarshal, 1918
 - looseUnmarshal, 1919
 - tightMarshal1, 1920
 - tightMarshal2, 1920
 - tightUnmarshal, 1921
- activemq::wireformat::openwire::marshal-
::generated::MessagePull-
Marshaller, 1944
 - ~MessagePullMarshaller, 1945
 - MessagePullMarshaller, 1945
 - createObject, 1945
 - getDataStructureType, 1945
 - looseMarshal, 1945
 - looseUnmarshal, 1946
 - tightMarshal1, 1946
 - tightMarshal2, 1947
 - tightUnmarshal, 1947
- activemq::wireformat::openwire::marshal-
::generated::NetworkBridge-
FilterMarshaller, 1974
 - ~NetworkBridgeFilterMarshaller,
1975
 - NetworkBridgeFilterMarshaller, 1975
 - createObject, 1975
 - getDataStructureType, 1976
 - looseMarshal, 1976
 - looseUnmarshal, 1976
 - tightMarshal1, 1977
 - tightMarshal2, 1977
 - tightUnmarshal, 1978
- activemq::wireformat::openwire::marshal-
::generated::PartialCommand-
Marshaller, 2079
 - ~PartialCommandMarshaller, 2080
 - PartialCommandMarshaller, 2080
 - createObject, 2080
 - getDataStructureType, 2080
 - looseMarshal, 2081
 - looseUnmarshal, 2081
 - tightMarshal1, 2082
 - tightMarshal2, 2082
 - tightUnmarshal, 2083
- activemq::wireformat::openwire::marshal-
::generated::ProducerAck-
Marshaller, 2175
 - ~ProducerAckMarshaller, 2176
 - ProducerAckMarshaller, 2176
 - createObject, 2176
 - getDataStructureType, 2176
 - looseMarshal, 2176
 - looseUnmarshal, 2177
 - tightMarshal1, 2177
 - tightMarshal2, 2178
 - tightUnmarshal, 2178
- activemq::wireformat::openwire::marshal-
::generated::ProducerIdMarshaller,
2186
 - ~ProducerIdMarshaller, 2187
 - ProducerIdMarshaller, 2187
 - createObject, 2187
 - getDataStructureType, 2187
 - looseMarshal, 2188
 - looseUnmarshal, 2188
 - tightMarshal1, 2188
 - tightMarshal2, 2189
 - tightUnmarshal, 2189
- activemq::wireformat::openwire::marshal-
::generated::ProducerInfo-
Marshaller, 2195
 - ~ProducerInfoMarshaller, 2196
 - ProducerInfoMarshaller, 2196
 - createObject, 2196

- getDataStructureType, 2196
- looseMarshal, 2196
- looseUnmarshal, 2197
- tightMarshal1, 2197
- tightMarshal2, 2198
- tightUnmarshal, 2198
- activemq::wireformat::openwire::marshal-
::generated::RemoveInfo-
Marshaller, 2271
 - ~RemoveInfoMarshaller, 2272
- RemoveInfoMarshaller, 2272
- createObject, 2272
- getDataStructureType, 2272
- looseMarshal, 2273
- looseUnmarshal, 2273
- tightMarshal1, 2274
- tightMarshal2, 2274
- tightUnmarshal, 2274
- activemq::wireformat::openwire::marshal-
::generated::RemoveSubscription-
InfoMarshaller, 2280
 - ~RemoveSubscriptionInfoMarshaller,
2281
- RemoveSubscriptionInfoMarshaller,
2281
- createObject, 2281
- getDataStructureType, 2281
- looseMarshal, 2281
- looseUnmarshal, 2282
- tightMarshal1, 2282
- tightMarshal2, 2282
- tightUnmarshal, 2283
- activemq::wireformat::openwire::marshal-
::generated::ReplayCommand-
Marshaller, 2287
 - ~ReplayCommandMarshaller, 2288
- ReplayCommandMarshaller, 2288
- createObject, 2288
- getDataStructureType, 2288
- looseMarshal, 2289
- looseUnmarshal, 2289
- tightMarshal1, 2289
- tightMarshal2, 2290
- tightUnmarshal, 2290
- activemq::wireformat::openwire::marshal-
::generated::ResponseMarshaller,
2307
 - ~ResponseMarshaller, 2308
- ResponseMarshaller, 2308
- createObject, 2308
- getDataStructureType, 2308
- looseMarshal, 2309
- looseUnmarshal, 2309
- tightMarshal1, 2310
- tightMarshal2, 2311
- tightUnmarshal, 2311
- activemq::wireformat::openwire::marshal-
::generated::SessionIdMarshaller,
2382
 - ~SessionIdMarshaller, 2383
- SessionIdMarshaller, 2383
- createObject, 2383
- getDataStructureType, 2383
- looseMarshal, 2384
- looseUnmarshal, 2384
- tightMarshal1, 2385
- tightMarshal2, 2385
- tightUnmarshal, 2385
- activemq::wireformat::openwire::marshal-
::generated::SessionInfo-
Marshaller, 2390
 - ~SessionInfoMarshaller, 2391
- SessionInfoMarshaller, 2391
- createObject, 2391
- getDataStructureType, 2391
- looseMarshal, 2391
- looseUnmarshal, 2392
- tightMarshal1, 2392
- tightMarshal2, 2392
- tightUnmarshal, 2393
- activemq::wireformat::openwire::marshal-
::generated::ShutdownInfo-
Marshaller, 2434
 - ~ShutdownInfoMarshaller, 2435
- ShutdownInfoMarshaller, 2435
- createObject, 2435
- getDataStructureType, 2435
- looseMarshal, 2436
- looseUnmarshal, 2436
- tightMarshal1, 2437
- tightMarshal2, 2437
- tightUnmarshal, 2437
- activemq::wireformat::openwire::marshal-
::generated::SubscriptionInfo-
Marshaller, 2635
 - ~SubscriptionInfoMarshaller, 2636
- SubscriptionInfoMarshaller, 2636
- createObject, 2636
- getDataStructureType, 2636
- looseMarshal, 2637

- looseUnmarshal, 2637
- tightMarshal1, 2637
- tightMarshal2, 2638
- tightUnmarshal, 2638
- activemq::wireformat::openwire::marshal-
::generated::TransactionId-
Marshaller, 2770
 - ~TransactionIdMarshaller, 2771
 - TransactionIdMarshaller, 2771
 - looseMarshal, 2771
 - looseUnmarshal, 2772
 - tightMarshal1, 2772
 - tightMarshal2, 2773
 - tightUnmarshal, 2773
- activemq::wireformat::openwire::marshal-
::generated::TransactionInfo-
Marshaller, 2778
 - ~TransactionInfoMarshaller, 2779
 - TransactionInfoMarshaller, 2779
 - createObject, 2779
 - getDataStructureType, 2779
 - looseMarshal, 2780
 - looseUnmarshal, 2780
 - tightMarshal1, 2780
 - tightMarshal2, 2781
 - tightUnmarshal, 2781
- activemq::wireformat::openwire::marshal-
::generated::WireFormatInfo-
Marshaller, 2900
 - ~WireFormatInfoMarshaller, 2901
 - WireFormatInfoMarshaller, 2901
 - createObject, 2901
 - getDataStructureType, 2901
 - looseMarshal, 2901
 - looseUnmarshal, 2902
 - tightMarshal1, 2902
 - tightMarshal2, 2903
 - tightUnmarshal, 2903
- activemq::wireformat::openwire::marshal-
::generated::XATransactionId-
Marshaller, 2942
 - ~XATransactionIdMarshaller, 2943
 - XATransactionIdMarshaller, 2943
 - createObject, 2943
 - getDataStructureType, 2944
 - looseMarshal, 2944
 - looseUnmarshal, 2944
 - tightMarshal1, 2945
 - tightMarshal2, 2945
 - tightUnmarshal, 2946
- activemq::wireformat::openwire::utils, 79
- activemq::wireformat::openwire::utils:-
BooleanStream, 552
 - ~BooleanStream, 553
 - BooleanStream, 553
 - clear, 554
 - marshal, 554
 - marshalledSize, 554
 - readBoolean, 554
 - unmarshal, 555
 - writeBoolean, 555
- activemq::wireformat::openwire::utils:-
HexTable, 1407
 - ~HexTable, 1408
 - HexTable, 1408
 - operator[], 1408
 - size, 1408
- activemq::wireformat::openwire::utils:-
MessagePropertyInterceptor, 1933
 - ~MessagePropertyInterceptor, 1934
 - MessagePropertyInterceptor, 1934
 - getBooleanProperty, 1934
 - getByteProperty, 1935
 - getDoubleProperty, 1935
 - getFloatProperty, 1936
 - getIntProperty, 1936
 - getLongProperty, 1936
 - getShortProperty, 1936
 - getStringProperty, 1937
 - setBooleanProperty, 1937
 - setByteProperty, 1937
 - setDoubleProperty, 1938
 - setFloatProperty, 1938
 - setIntProperty, 1938
 - setLongProperty, 1938
 - setShortProperty, 1939
 - setStringProperty, 1939
- activemq::wireformat::stomp, 79
- activemq::wireformat::stomp::Stomp-
CommandConstants, 2581
 - ABORT, 2583
 - ACK, 2583
 - ACK_AUTO, 2583
 - ACK_CLIENT, 2583
 - ACK_INDIVIDUAL, 2583
 - BEGIN, 2583
 - BYTES, 2583
 - COMMIT, 2583
 - CONNECT, 2583

- CONNECTED, 2583
- DISCONNECT, 2583
- ERROR_CMD, 2584
- HEADER_ACK, 2584
- HEADER_CLIENT_ID, 2584
- HEADER_CONSUMERPRIORITY, 2584
- HEADER_CONTENTLENGTH, 2584
- HEADER_CORRELATIONID, 2584
- HEADER_DESTINATION, 2584
- HEADER_DISPATCH_ASYNC, 2584
- HEADER_EXCLUSIVE, 2584
- HEADER_EXPIRES, 2584
- HEADER_ID, 2584
- HEADER_JMSPRIORITY, 2584
- HEADER_LOGIN, 2584
- HEADER_MAXPENDINGMSGLIMIT, 2585
- HEADER_MESSAGE, 2585
- HEADER_MESSAGEID, 2585
- HEADER_NOLOCAL, 2585
- HEADER_OLDSUBSCRIPTIONNAME, 2585
- HEADER_PASSWORD, 2585
- HEADER_PERSISTENT, 2585
- HEADER_PREFETCHSIZE, 2585
- HEADER_RECEIPTID, 2585
- HEADER_RECEIPT_REQUIRED, 2585
- HEADER_REDELIVERED, 2585
- HEADER_REDELIVERYCOUNT, 2585
- HEADER_REPLYTO, 2586
- HEADER_REQUESTID, 2586
- HEADER_RESPONSEID, 2586
- HEADER_RETROACTIVE, 2586
- HEADER_SELECTOR, 2586
- HEADER_SESSIONID, 2586
- HEADER_SUBSCRIPTION, 2586
- HEADER_SUBSCRIPTIONNAME, 2586
- HEADER_TIMESTAMP, 2586
- HEADER_TRANSACTIONID, 2586
- HEADER_TRANSFORMATION, 2586
- HEADER_TRANSFORMATION_ERROR, 2586
- HEADER_TYPE, 2587
- MESSAGE, 2587
- QUEUE_PREFIX, 2587
- RECEIPT, 2587
- SEND, 2587
- SUBSCRIBE, 2587
- TEMPQUEUE_PREFIX, 2587
- TEMPTOPIC_PREFIX, 2587
- TEXT, 2587
- TOPIC_PREFIX, 2587
- UNSUBSCRIBE, 2587
- activemq::wireformat::stomp::StompFrame, 2587
- ~StompFrame, 2589
- StompFrame, 2589
- clone, 2589
- copy, 2589
- fromStream, 2589
- getBody, 2590
- getBodyLength, 2590
- getCommand, 2590
- getProperties, 2590
- getProperty, 2591
- hasProperty, 2591
- removeProperty, 2591
- setBody, 2591
- setCommand, 2592
- setProperty, 2592
- toStream, 2592
- activemq::wireformat::stomp::StompHelper, 2592
- ~StompHelper, 2594
- StompHelper, 2594
- convertConsumerId, 2594
- convertDestination, 2594, 2595
- convertMessageId, 2595
- convertProducerId, 2595, 2596
- convertProperties, 2596
- convertTransactionId, 2597
- activemq::wireformat::stomp::StompWireFormat, 2597
- ~StompWireFormat, 2598
- StompWireFormat, 2598
- createNegotiator, 2598
- getVersion, 2599
- hasNegotiator, 2599
- inReceive, 2599
- marshal, 2599
- setVersion, 2600
- unmarshal, 2600
- activemq::wireformat::stomp::StompWireFormatFactory, 2601

- ~StompWireFormatFactory, 2601
- StompWireFormatFactory, 2601
- createWireFormat, 2601
- add
 - activemq::transport::failover::Close-TransportsTask, 818
 - activemq::transport::failover::Failover-Transport, 1308
 - decaf::util::AbstractCollection, 110
 - decaf::util::AbstractList, 125, 126
 - decaf::util::AbstractQueue, 139
 - decaf::util::AbstractSequentialList, 146
 - decaf::util::ArrayList, 448, 449
 - decaf::util::Collection, 853
 - decaf::util::concurrent::CopyOn-WriteArrayList, 1032, 1033
 - decaf::util::concurrent::CopyOn-WriteArrayList::ArrayListIterator, 459
 - decaf::util::concurrent::CopyOn-WriteArraySet, 1054
 - decaf::util::LinkedList, 1639, 1640
 - decaf::util::List, 1660
 - decaf::util::ListIterator, 1672
 - decaf::util::PriorityQueue, 2165
 - decaf::util::StlList, 2542
 - decaf::util::StlSet, 2577
- addAll
 - decaf::util::AbstractCollection, 110
 - decaf::util::AbstractList, 126
 - decaf::util::AbstractQueue, 140
 - decaf::util::AbstractSequentialList, 146
 - decaf::util::ArrayList, 450, 451
 - decaf::util::Collection, 854
 - decaf::util::concurrent::CopyOn-WriteArrayList, 1034, 1035
 - decaf::util::concurrent::CopyOn-WriteArraySet, 1055
 - decaf::util::LinkedList, 1641, 1642
 - decaf::util::List, 1661
 - decaf::util::StlList, 2543, 2544
- addAllAbsent
 - decaf::util::concurrent::CopyOn-WriteArrayList, 1036
- addAndGet
 - decaf::util::concurrent::atomic::AtomicInteger, 477
- addAsResource
 - decaf::internal::net::Network, 1969
- addCommand
 - activemq::state::TransactionState, 2785
- addConnection
 - activemq::cmsutil::ResourceLifecycle-Manager, 2295
- addConsumer
 - activemq::core::ActiveMQSession, 337
 - activemq::state::SessionState, 2396
- addDestination
 - activemq::cmsutil::ResourceLifecycle-Manager, 2295
- addDispatcher
 - activemq::core::ActiveMQConnection, 193
- addFirst
 - decaf::util::Deque, 1198
 - decaf::util::LinkedList, 1643
- addHandler
 - decaf::util::logging::Logger, 1696
- addIfAbsent
 - decaf::util::concurrent::CopyOn-WriteArrayList, 1036
- addLast
 - decaf::util::Deque, 1199
 - decaf::util::LinkedList, 1643
- addLogger
 - decaf::util::logging::LogManager, 1715
- addMarshaller
 - activemq::wireformat::openwire::OpenWireFormat, 2048
- addMessageConsumer
 - activemq::cmsutil::ResourceLifecycle-Manager, 2295
- addMessageProducer
 - activemq::cmsutil::ResourceLifecycle-Manager, 2295
- addNetworkResource
 - decaf::internal::net::Network, 1969
- addProducer
 - activemq::core::ActiveMQConnection, 194
 - activemq::core::ActiveMQSession, 338
 - activemq::state::SessionState, 2396
- addProducerState

- activemq::state::TransactionState, 2785
- addPropertyChangeListener
 - decaf::util::logging::LogManager, 1715
- addResource
 - decaf::internal::util::ResourceLifecycleManager, 2297
- addServiceListener
 - activemq::util::ServiceSupport, 2360
- addSession
 - activemq::cmsutil::ResourceLifecycleManager, 2296
 - activemq::core::ActiveMQConnection, 194
 - activemq::state::ConnectionState, 983
- addShutdownTask
 - decaf::internal::net::Network, 1970
- addSynchronization
 - activemq::core::ActiveMQTransactionContext, 425
- addTask
 - activemq::threads::CompositeTaskRunner, 894
- addTempDestination
 - activemq::state::ConnectionState, 983
- addTransactionState
 - activemq::state::ConnectionState, 983
- addTransportListener
 - activemq::core::ActiveMQConnection, 194
- addURI
 - activemq::transport::CompositeTransport, 897
 - activemq::transport::failover::FailoverTransport, 1308
 - activemq::transport::failover::URIPool, 2852
- addURIs
 - activemq::transport::failover::URIPool, 2852
- additionalPredicate
 - activemq::commands::ConsumerInfo, 1016
- address
 - decaf::net::SocketImpl, 2487
- addressBytes
 - decaf::net::InetAddress, 1444
- adjustMinimum
 - decaf::internal::util::TimerTaskHeap, 2754
- adler
 - z_stream_s, 2952
- advisory
 - activemq::commands::ActiveMQDestination, 256
- after
 - decaf::util::Date, 1140
- afterCommit
 - activemq::core::Synchronization, 2654
- afterExecute
 - decaf::util::concurrent::ThreadPoolExecutor, 2722
- afterMarshal
 - activemq::commands::BaseDataStructure, 529
 - activemq::wireformat::MarshalAware, 1793
- afterMessagesConsumed
 - activemq::core::ActiveMQConsumer, 236
- afterRollback
 - activemq::core::Synchronization, 2654
- afterUnmarshal
 - activemq::commands::BaseDataStructure, 529
 - activemq::commands::Message, 1826
 - activemq::commands::WireFormatInfo, 2892
 - activemq::wireformat::MarshalAware, 1793
- allocate
 - decaf::nio::ByteBuffer, 695
 - decaf::nio::CharBuffer, 788
 - decaf::nio::DoubleBuffer, 1262
 - decaf::nio::FloatBuffer, 1370
 - decaf::nio::IntBuffer, 1490
 - decaf::nio::LongBuffer, 1755
 - decaf::nio::ShortBuffer, 2421
- allowCoreThreadTimeout
 - decaf::util::concurrent::ThreadPoolExecutor, 2722
- allowsCoreThreadTimeout

- decaf::util::concurrent::ThreadPool-
Executor, 2723
- anyBytes
 - decaf::net::InetAddress, 1444
- append
 - decaf::io::Writer, 2910
 - decaf::lang::Appendable, 442, 443
 - decaf::nio::CharBuffer, 789, 790
- array
 - decaf::internal::nio::ByteBuffer, 660
 - decaf::internal::nio::CharArrayBuffer, 779
 - decaf::internal::nio::DoubleArray-
Buffer, 1254
 - decaf::internal::nio::FloatArrayBuffer, 1362
 - decaf::internal::nio::IntArrayBuffer, 1482
 - decaf::internal::nio::LongArrayBuffer, 1747
 - decaf::internal::nio::ShortArray-
Buffer, 2414
 - decaf::nio::ByteBuffer, 695
 - decaf::nio::CharBuffer, 790
 - decaf::nio::DoubleBuffer, 1262
 - decaf::nio::FloatBuffer, 1370
 - decaf::nio::IntBuffer, 1490
 - decaf::nio::LongBuffer, 1755
 - decaf::nio::ShortBuffer, 2422
- arrayOffset
 - decaf::internal::nio::ByteBuffer, 661
 - decaf::internal::nio::CharArrayBuffer, 779
 - decaf::internal::nio::DoubleArray-
Buffer, 1254
 - decaf::internal::nio::FloatArrayBuffer, 1363
 - decaf::internal::nio::IntArrayBuffer, 1483
 - decaf::internal::nio::LongArrayBuffer, 1747
 - decaf::internal::nio::ShortArray-
Buffer, 2414
 - decaf::nio::ByteBuffer, 696
 - decaf::nio::CharBuffer, 791
 - decaf::nio::DoubleBuffer, 1262
 - decaf::nio::FloatBuffer, 1371
 - decaf::nio::IntBuffer, 1491
- decaf::nio::LongBuffer, 1756
- decaf::nio::ShortBuffer, 2422
- arraycopy
 - decaf::lang::System, 2668–2671
- arrival
 - activemq::commands::Message, 1837
- asCharBuffer
 - decaf::internal::nio::ByteBuffer, 661
 - decaf::nio::ByteBuffer, 696
- asDoubleBuffer
 - decaf::internal::nio::ByteBuffer, 662
 - decaf::nio::ByteBuffer, 696
- asFloatBuffer
 - decaf::internal::nio::ByteBuffer, 662
 - decaf::nio::ByteBuffer, 697
- asIntBuffer
 - decaf::internal::nio::ByteBuffer, 663
 - decaf::nio::ByteBuffer, 697
- asLongBuffer
 - decaf::internal::nio::ByteBuffer, 663
 - decaf::nio::ByteBuffer, 698
- asReadOnlyBuffer
 - decaf::internal::nio::ByteBuffer, 663
 - decaf::internal::nio::CharArrayBuffer, 780
 - decaf::internal::nio::DoubleArray-
Buffer, 1255
 - decaf::internal::nio::FloatArrayBuffer, 1363
 - decaf::internal::nio::IntArrayBuffer, 1483
 - decaf::internal::nio::LongArrayBuffer, 1748
 - decaf::internal::nio::ShortArray-
Buffer, 2415
 - decaf::nio::ByteBuffer, 698
 - decaf::nio::CharBuffer, 791
 - decaf::nio::DoubleBuffer, 1263
 - decaf::nio::FloatBuffer, 1371
 - decaf::nio::IntBuffer, 1491
 - decaf::nio::LongBuffer, 1756
 - decaf::nio::ShortBuffer, 2423
- asShortBuffer

- decaf::internal::nio::ByteBuffer, 664
- decaf::nio::ByteBuffer, 698
- asciiToModifiedUtf8
 - activemq::util::MarshallingSupport, 1797
- atEOF
 - decaf::util::zip::InflaterInputStream, 1464
- avail_in
 - z_stream_s, 2952
- avail_out
 - z_stream_s, 2952
- available
 - decaf::internal::io::StandardInputStream, 2531
 - decaf::internal::net::ssl::openssl::OpenSSLSocket, 2020
 - decaf::internal::net::ssl::openssl::OpenSSLSocketInputStream, 2041
 - decaf::internal::net::tcp::TcpSocket, 2682
 - decaf::internal::net::tcp::TcpSocketInputStream, 2689
 - decaf::io::BlockingByteArrayInputStream, 536
 - decaf::io::BufferedInputStream, 592
 - decaf::io::ByteArrayInputStream, 683
 - decaf::io::FilterInputStream, 1337
 - decaf::io::InputStream, 1466
 - decaf::io::PushbackInputStream, 2217
 - decaf::net::SocketImpl, 2481
 - decaf::util::zip::InflaterInputStream, 1460
- availablePermits
 - decaf::util::concurrent::Semaphore, 2336
- availableProcessors
 - decaf::lang::System, 2671
- await
 - decaf::util::concurrent::CountDownLatch, 1063, 1064
 - decaf::util::concurrent::locks::Condition, 923, 924
- awaitNanos
 - decaf::util::concurrent::locks::Condition, 925
- awaitTermination
 - decaf::util::concurrent::ExecutorService, 1303
 - decaf::util::concurrent::ThreadPoolExecutor, 2723
- awaitUninterruptibly
 - decaf::util::concurrent::locks::Condition, 926
- awaitUntil
 - decaf::util::concurrent::locks::Condition, 927
- back
 - decaf::util::StlQueue, 2567, 2568
 - inflate_state, 1446
- base_dist
 - trees.h, 3152
- base_length
 - trees.h, 3152
- before
 - decaf::util::Date, 1141
- beforeEnd
 - activemq::core::Synchronization, 2654
- beforeExecute
 - decaf::util::concurrent::ThreadPoolExecutor, 2723
- beforeMarshal
 - activemq::commands::ActiveMQMapMessage, 271
 - activemq::commands::ActiveMQTextMessage, 406
 - activemq::commands::BaseDataStructure, 529
 - activemq::commands::Message, 1826
 - activemq::commands::WireFormatInfo, 2892
 - activemq::wireformat::MarshalAware, 1793
- beforeMessagesConsumed
 - activemq::core::ActiveMQConsumer, 237
- beforeUnmarshal
 - activemq::commands::BaseDataStructure, 530
 - activemq::wireformat::MarshalAware, 1794
- begin
 - activemq::core::ActiveMQTransactionContext, 425

- bi_buf
 - internal_state, 1524
- bi_valid
 - internal_state, 1524
- bind
 - decaf::internal::net::tcp::TcpSocket, 2682
 - decaf::net::ServerSocket, 2347
 - decaf::net::Socket, 2458
 - decaf::net::SocketImpl, 2481
- bitCount
 - decaf::lang::Integer, 1503
 - decaf::lang::Long, 1729
- bits
 - code, 851
 - inflate_state, 1446
- bl_count
 - internal_state, 1524
- bl_desc
 - internal_state, 1524
- bl_tree
 - internal_state, 1524
- block_start
 - internal_state, 1524
- boolValue
 - activemq::util::PrimitiveValueNode::PrimitiveValue, 2143
- booleanValue
 - decaf::lang::Boolean, 546
- branchQualifier
 - activemq::commands::XATransaction-Id, 2942
- brokerId
 - activemq::commands::BrokerInfo, 577
- brokerInTime
 - activemq::commands::Message, 1837
- brokerMasterConnector
 - activemq::commands::Connection-Info, 973
- brokerName
 - activemq::commands::BrokerInfo, 577
 - activemq::commands::Discovery-Event, 1229
- brokerOutTime
 - activemq::commands::Message, 1837
- brokerPath
 - activemq::commands::Connection-Info, 973
 - activemq::commands::Consumer-Info, 1016
 - activemq::commands::Destination-Info, 1217
 - activemq::commands::Message, 1837
 - activemq::commands::ProducerInfo, 2194
- brokerSequencedId
 - activemq::commands::MessageId, 1912
- brokerURL
 - activemq::commands::BrokerInfo, 577
- brokerUploadUrl
 - activemq::commands::BrokerInfo, 577
- browser
 - activemq::commands::Consumer-Info, 1016
- buf
 - decaf::util::zip::DeflaterOutput-Stream, 1193
- buff
 - decaf::util::zip::InflaterInputStream, 1464
- buffer
 - decaf::io::DataOutputStream, 1112
- buildIncomingCommands
 - activemq::transport::mock::Response-Builder, 2302
 - activemq::wireformat::openwire::OpenWireResponseBuilder, 2065
- buildMessage
 - decaf::lang::Exception, 1282
- buildResponse
 - activemq::transport::mock::Response-Builder, 2302
 - activemq::wireformat::openwire::OpenWireResponseBuilder, 2066
- byteArrayValue
 - activemq::util::PrimitiveValueNode::PrimitiveValue, 2143
- byteValue
 - activemq::util::PrimitiveValueNode::PrimitiveValue, 2143

- decaf::lang::Byte, 616
- decaf::lang::Character, 767
- decaf::lang::Double, 1238
- decaf::lang::Float, 1347
- decaf::lang::Integer, 1503
- decaf::lang::Long, 1729
- decaf::lang::Number, 1992
- decaf::lang::Short, 2400
- bytesToInt
 - decaf::net::InetAddress, 1439
- call
 - decaf::util::concurrent::Callable, 746
- cancel
 - activemq::threads::Scheduler, 2318
 - decaf::util::concurrent::Future, 1390
 - decaf::util::Timer, 2741
 - decaf::util::TimerTask, 2752
- capacity
 - decaf::nio::Buffer, 585
- cause
 - decaf::lang::Exception, 1286
- ceil
 - decaf::lang::Math, 1804
- charAt
 - decaf::lang::CharSequence, 804
 - decaf::lang::String, 2624
 - decaf::nio::CharBuffer, 792
- charValue
 - activemq::util::PrimitiveValueNode::PrimitiveValue, 2143
- charf
 - zconf.h, 3153
- check
 - inflate_state, 1446
- checkClosed
 - activemq::core::ActiveMQConnection, 195
 - decaf::io::InputStreamReader, 1476
 - decaf::io::OutputStreamWriter, 2075
 - decaf::net::ServerSocket, 2348
 - decaf::net::Socket, 2458
- checkClosedOrFailed
 - activemq::core::ActiveMQConnection, 195
- checkConnectionFactory
 - activemq::cmsutil::CmsAccessor, 820
- checkDestinationResolver
 - activemq::cmsutil::CmsDestination-Accessor, 824
- checkMapsUnmarshalled
 - activemq::commands::ActiveMQ-MapMessage, 271
- checkResult
 - decaf::internal::net::tcp::TcpSocket, 2682
- checkShutdown
 - activemq::state::ConnectionState, 983
 - activemq::state::SessionState, 2396
 - activemq::state::TransactionState, 2785
- checkValidity
 - decaf::security::cert::X509Certificate, 2916
- cleanup
 - activemq::core::ActiveMQConnection, 195
 - decaf::internal::AprPool, 445
- clear
 - activemq::core::ActiveMQSession-Executor, 356
 - activemq::core::FifoMessageDispatch-Channel, 1324
 - activemq::core::MessageDispatch-Channel, 1887
 - activemq::core::SimplePriority-MessageDispatchChannel, 2445
 - activemq::util::ActiveMQProperties, 317
 - activemq::util::PrimitiveValueNode, 2151
 - activemq::wireformat::openwire-::utils::BooleanStream, 554
- cms::CMSProperties, 831
- decaf::internal::util::ByteArray-Adapter, 630
- decaf::nio::Buffer, 585
- decaf::util::AbstractCollection, 111
- decaf::util::AbstractList, 127
- decaf::util::AbstractQueue, 141
- decaf::util::ArrayList, 451
- decaf::util::Collection, 856
- decaf::util::concurrent::Concurrent-StlMap, 909
- decaf::util::concurrent::CopyOn-WriteArrayList, 1036

- decaf::util::concurrent::CopyOn-WriteArraySet, 1055
- decaf::util::concurrent::LinkedBlockingQueue, 1624
- decaf::util::concurrent::SynchronousQueue, 2657
- decaf::util::LinkedList, 1643
- decaf::util::Map, 1770
- decaf::util::PriorityQueue, 2166
- decaf::util::Properties, 2202
- decaf::util::StlList, 2545
- decaf::util::StlMap, 2556
- decaf::util::StlQueue, 2568
- decaf::util::StlSet, 2578
- clearBody
 - activemq::commands::ActiveMQ-BytesMessage, 168
 - activemq::commands::ActiveMQ-MapMessage, 271
 - activemq::commands::ActiveMQ-MessageTemplate, 296
 - activemq::commands::ActiveMQ-StreamMessage, 361
 - activemq::commands::ActiveMQ-TextMessage, 406
 - cms::Message, 1843
- clearMessagesInProgress
 - activemq::core::ActiveMQConsumer, 237
 - activemq::core::ActiveMQSession, 338
 - activemq::core::ActiveMQSession-Executor, 356
- clearProperties
 - activemq::commands::ActiveMQ-MessageTemplate, 296
 - cms::Message, 1844
- clearProperty
 - decaf::lang::System, 2671
- clientId
 - activemq::commands::Connection-Info, 973
 - activemq::commands::JournalTopic-Ack, 1572
 - activemq::commands::Remove-SubscriptionInfo, 2279
 - activemq::commands::Subscription-Info, 2634
- clientMaster
 - activemq::commands::Connection-Info, 973
- clockSequence
 - decaf::util::UUID, 2879
- clone
 - activemq::commands::ActiveMQ-BlobMessage, 158
 - activemq::commands::ActiveMQ-BytesMessage, 168
 - activemq::commands::ActiveMQ-MapMessage, 271
 - activemq::commands::ActiveMQ-Message, 289
 - activemq::commands::ActiveMQ-ObjectMessage, 302
 - activemq::commands::ActiveMQ-Queue, 322
 - activemq::commands::ActiveMQ-StreamMessage, 361
 - activemq::commands::ActiveMQ-TempQueue, 387
 - activemq::commands::ActiveMQ-TempTopic, 397
 - activemq::commands::ActiveMQ-TextMessage, 407
 - activemq::commands::ActiveMQ-Topic, 415
 - activemq::commands::XATransaction-Id, 2938
 - activemq::core::policies::Default-PrefetchPolicy, 1147
 - activemq::core::policies::Default-RedeliveryPolicy, 1151
 - activemq::core::PrefetchPolicy, 2111
 - activemq::core::RedeliveryPolicy, 2252
 - activemq::exceptions::ActiveMQ-Exception, 263
 - activemq::exceptions::BrokerException, 564
 - activemq::exceptions::Connection-FailedException, 959
 - activemq::util::ActiveMQProperties, 317
 - activemq::wireformat::stomp::Stomp-Frame, 2589
 - cms::BytesMessage, 720
 - cms::CMSProperties, 831
 - cms::Destination, 1211
 - cms::Message, 1844

- cms::Xid, 2948
- decaf::internal::net::ssl::openssl::OpenSSLParameters, 2002
- decaf::internal::net::ssl::openssl::OpenSSLSocketException, 2032
- decaf::io::EOFException, 1277
- decaf::io::InterruptedIOException, 1533
- decaf::io::IOException, 1547
- decaf::io::UnsupportedEncodingException, 2824
- decaf::io::UTFDataFormatException, 2876
- decaf::lang::ArrayPointer, 466
- decaf::lang::Exception, 1282
- decaf::lang::exceptions::ClassCastException, 815
- decaf::lang::exceptions::IllegalArgumentException, 1415
- decaf::lang::exceptions::IllegalMonitorStateException, 1418
- decaf::lang::exceptions::IllegalStateException, 1422
- decaf::lang::exceptions::IllegalThreadStateException, 1425
- decaf::lang::exceptions::IndexOutOfBoundsException, 1431
- decaf::lang::exceptions::InterruptedException, 1531
- decaf::lang::exceptions::InvalidStateException, 1545
- decaf::lang::exceptions::NullPointerException, 1991
- decaf::lang::exceptions::NumberFormatException, 1997
- decaf::lang::exceptions::RuntimeException, 2317
- decaf::lang::exceptions::UnsupportedOperationException, 2827
- decaf::lang::Throwable, 2733
- decaf::net::BindException, 534
- decaf::net::ConnectException, 933
- decaf::net::HttpRetryException, 1411
- decaf::net::Inet4Address, 1433
- decaf::net::Inet6Address, 1436
- decaf::net::InetAddress, 1439
- decaf::net::MalformedURLException, 1768
- decaf::net::NoRouteToHostException, 1981
- decaf::net::PortUnreachableException, 2109
- decaf::net::ProtocolException, 2213
- decaf::net::SocketException, 2473
- decaf::net::SocketTimeoutException, 2495
- decaf::net::UnknownHostException, 2818
- decaf::net::UnknownServiceException, 2821
- decaf::net::URISyntaxException, 2859
- decaf::nio::BufferOverflowException, 611
- decaf::nio::BufferUnderflowException, 613
- decaf::nio::InvalidMarkException, 1541
- decaf::nio::ReadOnlyBufferException, 2246
- decaf::security::cert::CertificateEncodingException, 756
- decaf::security::cert::CertificateException, 758
- decaf::security::cert::CertificateExpiredException, 760
- decaf::security::cert::CertificateNotYetValidException, 762
- decaf::security::cert::CertificateParsingException, 764
- decaf::security::GeneralSecurityException, 1397
- decaf::security::InvalidKeyException, 1538
- decaf::security::KeyException, 1602
- decaf::security::KeyManagementException, 1605
- decaf::security::NoSuchAlgorithmException, 1983
- decaf::security::NoSuchProviderException, 1989
- decaf::security::SignatureException, 2440
- decaf::util::concurrent::BrokenBarrierException, 558
- decaf::util::concurrent::CancellationException, 750
- decaf::util::concurrent::Execution-

- Exception, 1297
- decaf::util::concurrent::Rejected-
ExecutionException, 2265
- decaf::util::concurrent::Timeout-
Exception, 2739
- decaf::util::ConcurrentModification-
Exception, 904
- decaf::util::NoSuchElementException, 1986
- decaf::util::Properties, 2202
- decaf::util::zip::DataFormatException, 1078
- decaf::util::zip::ZipException, 2955
- cloneDataStructure
 - activemq::commands::ActiveMQ-
BlobMessage, 158
 - activemq::commands::ActiveMQ-
BytesMessage, 169
 - activemq::commands::ActiveMQ-
Destination, 249
 - activemq::commands::ActiveMQ-
MapMessage, 272
 - activemq::commands::ActiveMQ-
Message, 289
 - activemq::commands::ActiveMQ-
ObjectMessage, 302
 - activemq::commands::ActiveMQ-
Queue, 322
 - activemq::commands::ActiveMQ-
StreamMessage, 362
 - activemq::commands::ActiveMQ-
TempDestination, 380
 - activemq::commands::ActiveMQ-
TempQueue, 388
 - activemq::commands::ActiveMQ-
TempTopic, 397
 - activemq::commands::ActiveMQ-
TextMessage, 407
 - activemq::commands::ActiveMQ-
Topic, 415
 - activemq::commands::Boolean-
Expression, 551
 - activemq::commands::BrokerError, 560
 - activemq::commands::BrokerId, 565
 - activemq::commands::BrokerInfo, 574
 - activemq::commands::Connection-
Control, 940
 - activemq::commands::Connection-
Error, 949
 - activemq::commands::ConnectionId, 961
 - activemq::commands::Connection-
Info, 969
 - activemq::commands::Consumer-
Control, 993
 - activemq::commands::ConsumerId, 1002
 - activemq::commands::Consumer-
Info, 1011
 - activemq::commands::Control-
Command, 1023
 - activemq::commands::DataArray-
Response, 1070
 - activemq::commands::DataResponse, 1113
 - activemq::commands::DataStructure, 1133
 - activemq::commands::Destination-
Info, 1215
 - activemq::commands::Discovery-
Event, 1227
 - activemq::commands::Exception-
Response, 1288
 - activemq::commands::FlushCommand, 1381
 - activemq::commands::Integer-
Response, 1517
 - activemq::commands::Journal-
QueueAck, 1562
 - activemq::commands::JournalTopic-
Ack, 1569
 - activemq::commands::JournalTrace, 1578
 - activemq::commands::Journal-
Transaction, 1585
 - activemq::commands::KeepAliveInfo, 1592
 - activemq::commands::LastPartial-
Command, 1606
 - activemq::commands::LocalTransaction-
Id, 1676
 - activemq::commands::Message, 1826
 - activemq::commands::MessageAck, 1869
 - activemq::commands::Message-
Dispatch, 1883

- activemq::commands::Message-DispatchNotification, 1896
- activemq::commands::MessageId, 1909
- activemq::commands::MessagePull, 1941
- activemq::commands::Network-BridgeFilter, 1972
- activemq::commands::Partial-Command, 2077
- activemq::commands::ProducerAck, 2172
- activemq::commands::ProducerId, 2183
- activemq::commands::ProducerInfo, 2191
- activemq::commands::RemoveInfo, 2269
- activemq::commands::Remove-SubscriptionInfo, 2276
- activemq::commands::Replay-Command, 2285
- activemq::commands::Response, 2299
- activemq::commands::SessionId, 2379
- activemq::commands::SessionInfo, 2387
- activemq::commands::ShutdownInfo, 2432
- activemq::commands::Subscription-Info, 2632
- activemq::commands::TransactionId, 2768
- activemq::commands::Transaction-Info, 2775
- activemq::commands::WireFormat-Info, 2892
- activemq::commands::XATransaction-Id, 2938
- close
 - activemq::cmsutil::CachedConsumer, 735
 - activemq::cmsutil::CachedProducer, 740
 - activemq::cmsutil::PooledSession, 2095
 - activemq::commands::ActiveMQ-TempDestination, 380
 - activemq::commands::Connection-Control, 943
 - activemq::commands::Consumer-Control, 996
 - activemq::core::ActiveMQConnection, 195
 - activemq::core::ActiveMQConsumer, 237
 - activemq::core::ActiveMQProducer, 309
 - activemq::core::ActiveMQQueue-Browser, 327
 - activemq::core::ActiveMQSession, 338
 - activemq::core::ActiveMQSession-Executor, 357
 - activemq::core::FifoMessageDispatch-Channel, 1324
 - activemq::core::MessageDispatch-Channel, 1887
 - activemq::core::SimplePriority-MessageDispatchChannel, 2446
 - activemq::transport::correlator::-ResponseCorrelator, 2304
 - activemq::transport::failover::Failover-Transport, 1308
 - activemq::transport::inactivity::-InactivityMonitor, 1427
 - activemq::transport::IOTransport, 1551
 - activemq::transport::mock::Mock-Transport, 1950
 - activemq::transport::tcp::TcpTransport, 2694
 - activemq::transport::TransportFilter, 2802
 - activemq::wireformat::openwire::-OpenWireFormatNegotiator, 2061
- cms::Closeable, 816
- cms::Connection, 935
- cms::Session, 2365
- decaf::internal::io::StandardError-OutputStream, 2529
- decaf::internal::io::StandardOutput-Stream, 2532
- decaf::internal::net::ssl::openssl::-OpenSSLSocket, 2020
- decaf::internal::net::ssl::openssl::-OpenSSLSocketInputStream, 2061

- 2042
- decaf::internal::net::ssl::openssl::-
 OpenSSLSocketOutputStream,
 2044
- decaf::internal::net::tcp::TcpSocket,
 2682
- decaf::internal::net::tcp::TcpSocket-
 InputStream, 2690
- decaf::internal::net::tcp::TcpSocket-
 OutputStream, 2692
- decaf::io::BlockingByteArrayInput-
 Stream, 537
- decaf::io::BufferedInputStream, 592
- decaf::io::Closeable, 817
- decaf::io::FilterInputStream, 1337
- decaf::io::FilterOutputStream, 1342
- decaf::io::InputStream, 1466
- decaf::io::InputStreamReader, 1476
- decaf::io::OutputStream, 2068
- decaf::io::OutputStreamWriter, 2075
- decaf::net::ServerSocket, 2348
- decaf::net::Socket, 2458
- decaf::net::SocketImpl, 2482
- decaf::util::logging::ConsoleHandler,
 991
- decaf::util::logging::StreamHandler,
 2604, 2605
- decaf::util::zip::DeflaterOutput-
 Stream, 1192
- decaf::util::zip::InflaterInputStream,
 1461
- closed
 - activemq::core::ActiveMQSession,
 354
 - decaf::io::FilterInputStream, 1340
 - decaf::io::FilterOutputStream, 1344
- cluster
 - activemq::commands::Message,
 1837
- cms, 80
- cms/Config.h
 - CMS_API, 3040
- cms::BytesMessage, 718
 - ~BytesMessage, 720
 - clone, 720
 - getBodyBytes, 721
 - getBodyLength, 721
 - readBoolean, 721
 - readByte, 722
 - readBytes, 722, 723
 - readChar, 724
 - readDouble, 724
 - readFloat, 725
 - readInt, 725
 - readLong, 726
 - readShort, 726
 - readString, 727
 - readUTF, 728
 - readUnsignedShort, 727
 - reset, 728
 - setBodyBytes, 728
 - writeBoolean, 729
 - writeByte, 729
 - writeBytes, 730
 - writeChar, 731
 - writeDouble, 731
 - writeFloat, 731
 - writeInt, 732
 - writeLong, 732
 - writeShort, 733
 - writeString, 733
 - writeUTF, 734
 - writeUnsignedShort, 733
- cms::CMSException, 826
 - ~CMSException, 827
 - CMSException, 827
 - getCause, 827
 - getMessage, 827
 - getStackTrace, 828
 - getStackTraceString, 828
 - printStackTrace, 828
 - setMark, 828
 - what, 829
- cms::CMSProperties, 830
 - ~CMSProperties, 831
 - clear, 831
 - clone, 831
 - copy, 832
 - getProperty, 832
 - hasProperty, 832
 - isEmpty, 833
 - propertyNames, 833
 - remove, 833
 - setProperty, 834
 - size, 834
 - toArray, 834
 - toString, 834
- cms::CMSSecurityException, 835
 - ~CMSSecurityException, 836
 - CMSSecurityException, 835, 836

- cms::Closeable, 815
 - ~Closeable, 816
 - close, 816
- cms::Connection, 933
 - ~Connection, 935
 - close, 935
 - createSession, 935, 936
 - getClientID, 936
 - getExceptionListener, 936
 - getMetaData, 937
 - setClientID, 937
 - setExceptionListener, 938
- cms::ConnectionFactory, 955
 - ~ConnectionFactory, 956
 - createCMSConnectionFactory, 956
 - createConnection, 957, 958
- cms::ConnectionMetaData, 978
 - ~ConnectionMetaData, 979
 - getCMSMajorVersion, 979
 - getCMSMinorVersion, 979
 - getCMSProviderName, 979
 - getCMSVersion, 980
 - getCMSXPropertyNames, 980
 - getProviderMajorVersion, 981
 - getProviderMinorVersion, 981
 - getProviderVersion, 981
- cms::DeliveryMode, 1195
 - ~DeliveryMode, 1196
 - DELIVERY_MODE, 1196
 - NON_PERSISTENT, 1196
 - PERSISTENT, 1196
- cms::Destination, 1210
 - ~Destination, 1211
 - DestinationType, 1211
 - QUEUE, 1211
 - TEMPORARY_QUEUE, 1211
 - TEMPORARY_TOPIC, 1211
 - TOPIC, 1211
 - clone, 1211
 - copy, 1211
 - equals, 1212
 - getCMSProperties, 1212
 - getDestinationType, 1212
- cms::ExceptionListener, 1286
 - ~ExceptionListener, 1287
 - onException, 1287
- cms::IllegalStateException, 1419
 - ~IllegalStateException, 1420
 - IllegalStateException, 1419, 1420
- cms::InvalidClientIDException, 1534
 - ~InvalidClientIDException, 1535
 - InvalidClientIDException, 1534, 1535
- cms::InvalidDestinationException, 1535
 - ~InvalidDestinationException, 1536
 - InvalidDestinationException, 1536
- cms::InvalidSelectorException, 1541
 - ~InvalidSelectorException, 1542
 - InvalidSelectorException, 1542
- cms::MapMessage, 1779
 - ~MapMessage, 1782
 - getBoolean, 1782
 - getByte, 1782
 - getBytes, 1782
 - getChar, 1783
 - getDouble, 1783
 - getFloat, 1784
 - getInt, 1784
 - getLong, 1784
 - getMapNames, 1785
 - getShort, 1785
 - getString, 1786
 - isEmpty, 1786
 - itemExists, 1786
 - setBoolean, 1787
 - setByte, 1787
 - setBytes, 1788
 - setChar, 1788
 - setDouble, 1789
 - setFloat, 1789
 - setInt, 1789
 - setLong, 1790
 - setShort, 1790
 - setString, 1791
- cms::Message, 1839
 - ~Message, 1843
 - DEFAULT_DELIVERY_MODE, 1867
 - DEFAULT_MSG_PRIORITY, 1867
 - DEFAULT_TIME_TO_LIVE, 1867
 - acknowledge, 1843
 - clearBody, 1843
 - clearProperties, 1844
 - clone, 1844
 - getBooleanProperty, 1844
 - getByteProperty, 1845
 - getCMSCorrelationID, 1846
 - getCMSDeliveryMode, 1846
 - getCMSDestination, 1847
 - getCMSExpiration, 1847
 - getCMSMessageID, 1848
 - getCMSPriority, 1849

- getCMSRedelivered, 1849
- getCMSReplyTo, 1850
- getCMSTimestamp, 1850
- getCMSType, 1851
- getDoubleProperty, 1852
- getFloatProperty, 1852
- getIntProperty, 1853
- getLongProperty, 1854
- getPropertyNames, 1854
- getShortProperty, 1855
- getStringProperty, 1855
- propertyExists, 1856
- setBooleanProperty, 1857
- setByteProperty, 1857
- setCMSCorrelationID, 1858
- setCMSDeliveryMode, 1859
- setCMSDestination, 1859
- setCMSExpiration, 1860
- setCMSMessageID, 1860
- setCMSPriority, 1860
- setCMSRedelivered, 1861
- setCMSReplyTo, 1861
- setCMSTimestamp, 1862
- setCMSType, 1863
- setDoubleProperty, 1864
- setFloatProperty, 1864
- setIntProperty, 1865
- setLongProperty, 1865
- setShortProperty, 1866
- setStringProperty, 1866
- cms::MessageConsumer, 1877
 - ~MessageConsumer, 1878
 - getMessageListener, 1878
 - getMessageSelector, 1878
 - receive, 1879
 - receiveNoWait, 1879
 - setMessageListener, 1880
- cms::MessageEOFException, 1905
 - ~MessageEOFException, 1906
 - MessageEOFException, 1906
- cms::MessageEnumeration, 1903
 - ~MessageEnumeration, 1904
 - hasMoreMessages, 1904
 - nextMessage, 1904
- cms::MessageFormatException, 1906
 - ~MessageFormatException, 1907
 - MessageFormatException, 1907
- cms::MessageListener, 1916
 - ~MessageListener, 1917
 - onMessage, 1917
- cms::MessageNotReadableException, 1922
 - ~MessageNotReadableException, 1923
 - MessageNotReadableException, 1923
- cms::MessageNotWriteableException, 1923
 - ~MessageNotWriteableException, 1924
 - MessageNotWriteableException, 1924
- cms::MessageProducer, 1924
 - ~MessageProducer, 1926
 - getDeliveryMode, 1926
 - getDisableMessageID, 1926
 - getDisableMessageTimeStamp, 1927
 - getPriority, 1927
 - getTimeToLive, 1928
 - send, 1928–1930
 - setDeliveryMode, 1930
 - setDisableMessageID, 1931
 - setDisableMessageTimeStamp, 1931
 - setPriority, 1932
 - setTimeToLive, 1932
- cms::ObjectMessage, 1997
 - ~ObjectMessage, 1998
- cms::Queue, 2221
 - ~Queue, 2222
 - getQueueName, 2222
- cms::QueueBrowser, 2227
 - ~QueueBrowser, 2228
 - getEnumeration, 2228
 - getMessageSelector, 2228
 - getQueue, 2228
- cms::Session, 2361
 - ~Session, 2365
 - AUTO_ACKNOWLEDGE, 2365
 - AcknowledgeMode, 2365
 - CLIENT_ACKNOWLEDGE, 2365
 - DUPS_OK_ACKNOWLEDGE, 2365
 - INDIVIDUAL_ACKNOWLEDGE, 2365
 - SESSION_TRANSACTED, 2365
 - close, 2365
 - commit, 2366
 - createBrowser, 2366, 2367
 - createBytesMessage, 2367

- createConsumer, 2368, 2369
- createDurableConsumer, 2370
- createMapMessage, 2371
- createMessage, 2371
- createProducer, 2371
- createQueue, 2372
- createStreamMessage, 2372
- createTemporaryQueue, 2373
- createTemporaryTopic, 2373
- createTextMessage, 2373, 2374
- createTopic, 2374
- getAcknowledgeMode, 2374
- isTransacted, 2375
- recover, 2375
- rollback, 2376
- unsubscribe, 2376
- cms::Startable, 2534
 - ~Startable, 2534
- start, 2534
- cms::Stoppable, 2602
 - ~Stoppable, 2602
- stop, 2602
- cms::StreamMessage, 2606
 - ~StreamMessage, 2608
- readBoolean, 2608
- readByte, 2609
- readBytes, 2609, 2610
- readChar, 2611
- readDouble, 2612
- readFloat, 2612
- readInt, 2613
- readLong, 2613
- readShort, 2614
- readString, 2614
- readUnsignedShort, 2615
- writeBoolean, 2615
- writeByte, 2615
- writeBytes, 2616
- writeChar, 2617
- writeDouble, 2617
- writeFloat, 2618
- writeInt, 2618
- writeLong, 2618
- writeShort, 2619
- writeString, 2619
- writeUnsignedShort, 2620
- cms::TemporaryQueue, 2698
 - ~TemporaryQueue, 2699
- destroy, 2699
- getQueueName, 2699
- cms::TemporaryTopic, 2700
 - ~TemporaryTopic, 2700
- destroy, 2701
- getTopicName, 2701
- cms::TextMessage, 2701
 - ~TextMessage, 2702
- getText, 2702
- setText, 2702, 2703
- cms::Topic, 2764
 - ~Topic, 2765
- getTopicName, 2765
- cms::TransactionInProgressException, 2782
 - ~TransactionInProgressException, 2783
- TransactionInProgressException, 2783
- cms::TransactionRolledBackException, 2783
 - ~TransactionRolledBackException, 2784
- TransactionRolledBackException, 2784
- cms::UnsupportedOperationException, 2827
 - ~UnsupportedOperationException, 2828
- UnsupportedOperationException, 2828
- cms::XAConnection, 2917
 - ~XAConnection, 2917
- createXASession, 2918
- cms::XAConnectionFactory, 2918
 - ~XAConnectionFactory, 2919
- createCMSXAConnectionFactory, 2919
- createXAConnection, 2920
- cms::XAException, 2921
 - ~XAException, 2923
- XAER_ASYNC, 2925
- XAER_DUPID, 2926
- XAER_INVALID, 2926
- XAER_NOTA, 2926
- XAER_OUTSIDE, 2926
- XAER_PROTO, 2926
- XAER_RMERR, 2926
- XAER_RMFAIL, 2926
- XAException, 2923
- XA_HEURCOM, 2924
- XA_HEURHAZ, 2924

- XA_HEURMIX, 2924
- XA_HEURRB, 2924
- XA_NOMIGRATE, 2924
- XA_RBBASE, 2924
- XA_RBCOMMFAIL, 2924
- XA_RBDEADLOCK, 2924
- XA_RBEND, 2925
- XA_RBINTEGRITY, 2925
- XA_RBOTHER, 2925
- XA_RBPROTO, 2925
- XA_RBROLLBACK, 2925
- XA_RBTIMEOUT, 2925
- XA_RBTRANSIENT, 2925
- XA_RDONLY, 2925
- XA_RETRY, 2925
- getErrorCode, 2923
- setErrorCode, 2924
- cms::XAResource, 2926
 - ~XAResource, 2928
 - TMENDRSCAN, 2933
 - TMFAIL, 2933
 - TMJOIN, 2933
 - TMNOFLAGS, 2933
 - TMONEPHASE, 2934
 - TMRESUME, 2934
 - TMSTARTRSCAN, 2934
 - TMSUCCESS, 2934
 - TMSUSPEND, 2934
 - XA_OK, 2934
 - XA_RDONLY, 2934
 - commit, 2928
 - end, 2929
 - forget, 2929
 - getTransactionTimeout, 2930
 - isSameRM, 2930
 - prepare, 2931
 - recover, 2931
 - rollback, 2932
 - setTransactionTimeout, 2932
 - start, 2933
- cms::XASession, 2935
 - ~XASession, 2935
 - getXAResource, 2936
- cms::Xid, 2946
 - ~Xid, 2948
 - MAXBQUALSIZE, 2949
 - MAXGTRIDSIZ, 2949
 - Xid, 2948
 - clone, 2948
 - equals, 2948
 - getBranchQualifier, 2948
 - getFormatId, 2949
 - getGlobalTransactionId, 2949
- code, 851
 - bits, 851
 - ct_data_s, 1068
 - op, 851
 - val, 851
- codes
 - inflate_state, 1446
- codetype
 - inftrees.h, 3150
- comm_max
 - gz_header_s, 1399
- command
 - activemq::commands::Control-Command, 1025
- commandId
 - activemq::commands::Partial-Command, 2079
- comment
 - gz_header_s, 1399
- commit
 - activemq::cmsutil::PooledSession, 2095
 - activemq::core::ActiveMQConsumer, 237
 - activemq::core::ActiveMQSession, 338
 - activemq::core::ActiveMQTransaction-Context, 425
 - activemq::core::ActiveMQXASession, 437
 - cms::Session, 2366
 - cms::XAResource, 2928
- compact
 - decaf::internal::nio::ByteBuffer, 664
 - decaf::internal::nio::CharArrayBuffer, 780
 - decaf::internal::nio::DoubleArray-Buffer, 1255
 - decaf::internal::nio::FloatArrayBuffer, 1363
 - decaf::internal::nio::IntArrayBuffer, 1484
 - decaf::internal::nio::LongArrayBuffer, 1748
 - decaf::internal::nio::ShortArray-Buffer, 2415

- decaf::nio::ByteBuffer, 699
- decaf::nio::CharBuffer, 792
- decaf::nio::DoubleBuffer, 1263
- decaf::nio::FloatBuffer, 1372
- decaf::nio::IntBuffer, 1492
- decaf::nio::LongBuffer, 1757
- decaf::nio::ShortBuffer, 2423
- comparator
 - decaf::util::PriorityQueue, 2167
- compare
 - activemq::util::IdGenerator, 1412
 - decaf::lang::ArrayPointerComparator, 471
 - decaf::lang::Double, 1238
 - decaf::lang::Float, 1347
 - decaf::lang::PointerComparator, 2092
 - decaf::util::Comparator, 889
 - decaf::util::comparators::Less, 1613
- compareAndSet
 - decaf::util::concurrent::atomic::AtomicBoolean, 474
 - decaf::util::concurrent::atomic::AtomicInteger, 477
 - decaf::util::concurrent::atomic::AtomicReference, 485
- compareTo
 - activemq::commands::BrokerId, 566
 - activemq::commands::ConnectionId, 961
 - activemq::commands::ConsumerId, 1002
 - activemq::commands::LocalTransactionId, 1676
 - activemq::commands::MessageId, 1910
 - activemq::commands::ProducerId, 2183
 - activemq::commands::SessionId, 2380
 - activemq::commands::TransactionId, 2768
 - activemq::commands::XATransactionId, 2938
 - decaf::lang::Boolean, 547
 - decaf::lang::Byte, 616, 617
 - decaf::lang::Character, 767
 - decaf::lang::Comparable, 886
 - decaf::lang::Double, 1239
 - decaf::lang::Float, 1348
 - decaf::lang::Integer, 1503
 - decaf::lang::Long, 1729, 1730
 - decaf::lang::Short, 2400, 2401
 - decaf::net::URI, 2833
 - decaf::nio::ByteBuffer, 699
 - decaf::nio::CharBuffer, 793
 - decaf::nio::DoubleBuffer, 1264
 - decaf::nio::FloatBuffer, 1372
 - decaf::nio::IntBuffer, 1492
 - decaf::nio::LongBuffer, 1757
 - decaf::nio::ShortBuffer, 2424
 - decaf::util::concurrent::TimeUnit, 2758
 - decaf::util::Date, 1141
 - decaf::util::logging::Level, 1618
 - decaf::util::UUID, 2879
- compressed
 - activemq::commands::Message, 1837
- condition
 - decaf::util::concurrent::ConditionHandle, 928
- config
 - activemq::core::ActiveMQSession, 354
 - decaf::util::logging::Logger, 1697
- configure
 - activemq::core::PrefetchPolicy, 2111
 - activemq::core::RedeliveryPolicy, 2252
 - activemq::wireformat::openwire::marshal::generated::MarshallerFactory, 1796
- configureSocket
 - activemq::transport::tcp::SslTransport, 2526
 - activemq::transport::tcp::TcpTransport, 2694
- connect
 - activemq::transport::tcp::TcpTransport, 2695
 - decaf::internal::net::ssl::openssl::OpenSSLSocket, 2020
 - decaf::internal::net::tcp::TcpSocket, 2683
 - decaf::net::Socket, 2458, 2459
 - decaf::net::SocketImpl, 2482
- connectedBrokers
 - activemq::commands::ConnectionControl, 943

- connection
 - activemq::commands::ActiveMQ-TempDestination, 382
 - activemq::commands::Message, 1837
 - activemq::core::ActiveMQSession, 354
- connectionId
 - activemq::commands::BrokerInfo, 577
 - activemq::commands::Connection-Error, 951
 - activemq::commands::Connection-Info, 973
 - activemq::commands::ConsumerId, 1004
 - activemq::commands::Destination-Info, 1217
 - activemq::commands::LocalTransaction-Id, 1678
 - activemq::commands::ProducerId, 2185
 - activemq::commands::Remove-SubscriptionInfo, 2279
 - activemq::commands::SessionId, 2382
 - activemq::commands::Transaction-Info, 2777
- connectionInterruptProcessingComplete
 - activemq::state::ConnectionState-Tracker, 986
- consumerId
 - activemq::commands::Consumer-Control, 996
 - activemq::commands::Consumer-Info, 1016
 - activemq::commands::MessageAck, 1872
 - activemq::commands::Message-Dispatch, 1885
 - activemq::commands::Message-DispatchNotification, 1899
 - activemq::commands::MessagePull, 1943
- consumerIds
 - activemq::core::ActiveMQSession, 354
- consumers
 - activemq::core::ActiveMQSession, 354
- contains
 - decaf::util::AbstractCollection, 112
 - decaf::util::ArrayList, 452
 - decaf::util::Collection, 857
 - decaf::util::concurrent::CopyOn-WriteArrayList, 1037
 - decaf::util::concurrent::CopyOn-WriteArraySet, 1056
 - decaf::util::concurrent::Synchronous-Queue, 2658
 - decaf::util::LinkedList, 1644
 - decaf::util::StlList, 2545
 - decaf::util::StlSet, 2578
- containsAll
 - decaf::util::AbstractCollection, 113
 - decaf::util::Collection, 858
 - decaf::util::concurrent::CopyOn-WriteArrayList, 1037
 - decaf::util::concurrent::CopyOn-WriteArraySet, 1056
 - decaf::util::concurrent::Synchronous-Queue, 2658
- containsKey
 - decaf::util::concurrent::Concurrent-StlMap, 910
 - decaf::util::Map, 1770
 - decaf::util::StlMap, 2557
- containsValue
 - decaf::util::concurrent::Concurrent-StlMap, 910
 - decaf::util::Map, 1771
 - decaf::util::StlMap, 2557
- content
 - activemq::commands::Message, 1837
- convert
 - activemq::util::PrimitiveValueConverter, 2145
 - decaf::util::concurrent::TimeUnit, 2758
- convertConsumerId
 - activemq::wireformat::stomp::Stomp-Helper, 2594
- convertDestination
 - activemq::wireformat::stomp::Stomp-Helper, 2594, 2595
- convertMessageId
 - activemq::wireformat::stomp::Stomp-Helper, 2595
- convertProducerId
 -

- activemq::wireformat::stomp::Stomp-Helper, 2595, 2596
- convertProperties
 - activemq::wireformat::stomp::Stomp-Helper, 2596
- convertToCMSException
 - activemq::exceptions::ActiveMQ-Exception, 263
- convertTransactionId
 - activemq::wireformat::stomp::Stomp-Helper, 2597
- copy
 - activemq::commands::ActiveMQ-Queue, 323
 - activemq::commands::ActiveMQ-TempQueue, 388
 - activemq::commands::ActiveMQ-TempTopic, 397
 - activemq::commands::ActiveMQ-Topic, 416
 - activemq::util::ActiveMQProperties, 317
 - activemq::wireformat::stomp::Stomp-Frame, 2589
- cms::CMSProperties, 832
- cms::Destination, 1211
- decaf::util::AbstractCollection, 113
- decaf::util::Collection, 858
- decaf::util::concurrent::Concurrent-StlMap, 911
- decaf::util::concurrent::CopyOn-WriteArrayList, 1037
- decaf::util::concurrent::CopyOn-WriteArraySet, 1057
- decaf::util::LinkedList, 1645
- decaf::util::Map, 1772
- decaf::util::Properties, 2202
- decaf::util::StlList, 2546
- decaf::util::StlMap, 2558
- decaf::util::StlSet, 2579
- copyDataStructure
 - activemq::commands::ActiveMQ-BlobMessage, 158
 - activemq::commands::ActiveMQ-BytesMessage, 169
 - activemq::commands::ActiveMQ-Destination, 249
 - activemq::commands::ActiveMQ-MapMessage, 272
 - activemq::commands::ActiveMQ-Message, 289
 - activemq::commands::ActiveMQ-ObjectMessage, 302
 - activemq::commands::ActiveMQ-Queue, 323
 - activemq::commands::ActiveMQ-StreamMessage, 362
 - activemq::commands::ActiveMQ-TempDestination, 380
 - activemq::commands::ActiveMQ-TempQueue, 388
 - activemq::commands::ActiveMQ-TempTopic, 398
 - activemq::commands::ActiveMQ-TextMessage, 407
 - activemq::commands::ActiveMQ-Topic, 416
 - activemq::commands::BaseCommand, 493
 - activemq::commands::BaseData-Structure, 530
 - activemq::commands::Boolean-Expression, 551
 - activemq::commands::BrokerError, 560
 - activemq::commands::BrokerId, 566
 - activemq::commands::BrokerInfo, 574
 - activemq::commands::Connection-Control, 940
 - activemq::commands::Connection-Error, 949
 - activemq::commands::ConnectionId, 962
 - activemq::commands::Connection-Info, 970
 - activemq::commands::Consumer-Control, 993
 - activemq::commands::ConsumerId, 1002
 - activemq::commands::Consumer-Info, 1011
 - activemq::commands::Control-Command, 1024
 - activemq::commands::DataArray-Response, 1070
 - activemq::commands::DataResponse, 1113
 - activemq::commands::DataStructure,

- 1134
- activemq::commands::Destination-Info, 1215
- activemq::commands::Discovery-Event, 1228
- activemq::commands::Exception-Response, 1289
- activemq::commands::FlushCommand, 1382
- activemq::commands::Integer-Response, 1517
- activemq::commands::Journal-QueueAck, 1562
- activemq::commands::JournalTopic-Ack, 1569
- activemq::commands::JournalTrace, 1578
- activemq::commands::Journal-Transaction, 1585
- activemq::commands::KeepAliveInfo, 1592
- activemq::commands::LastPartial-Command, 1607
- activemq::commands::LocalTransaction-Id, 1676
- activemq::commands::Message, 1827
- activemq::commands::MessageAck, 1869
- activemq::commands::Message-Dispatch, 1883
- activemq::commands::Message-DispatchNotification, 1896
- activemq::commands::MessageId, 1910
- activemq::commands::MessagePull, 1941
- activemq::commands::Network-BridgeFilter, 1972
- activemq::commands::Partial-Command, 2077
- activemq::commands::ProducerAck, 2172
- activemq::commands::ProducerId, 2183
- activemq::commands::ProducerInfo, 2192
- activemq::commands::RemoveInfo, 2269
- activemq::commands::Remove-SubscriptionInfo, 2277
- activemq::commands::Replay-Command, 2285
- activemq::commands::Response, 2299
- activemq::commands::SessionId, 2380
- activemq::commands::SessionInfo, 2387
- activemq::commands::ShutdownInfo, 2432
- activemq::commands::Subscription-Info, 2632
- activemq::commands::TransactionId, 2768
- activemq::commands::Transaction-Info, 2775
- activemq::commands::WireFormat-Info, 2892
- activemq::commands::XATransaction-Id, 2939
- correlationId
 - activemq::commands::Message, 1837
 - activemq::commands::MessagePull, 1943
 - activemq::commands::Response, 2301
- countDown
 - decaf::util::concurrent::CountDown-Latch, 1065
- countTokens
 - decaf::util::StringTokenizer, 2628
- crc32.h
 - crc_table, 3142
- crc_table
 - crc32.h, 3142
- create
 - activemq::transport::failover::Failover-TransportFactory, 1319
 - activemq::transport::mock::Mock-TransportFactory, 1959
 - activemq::transport::tcp::TcpTransport-Factory, 2697
 - activemq::transport::Transport-Factory, 2799
 - activemq::util::CMSException-Support, 829
 - decaf::internal::net::tcp::TcpSocket, 2683

- decaf::internal::util::concurrent::ConditionImpl, 929
- decaf::internal::util::concurrent::MutexImpl, 1967
- decaf::net::SocketImpl, 2482
- decaf::net::URI, 2833
- createActiveMQConnection
 - activemq::core::ActiveMQConnectionFactory, 216
 - activemq::core::ActiveMQXAConnectionFactory, 435
- createBrowser
 - activemq::cmsutil::PooledSession, 2095, 2096
 - activemq::core::ActiveMQSession, 338, 339
 - cms::Session, 2366, 2367
- createByteBuffer
 - decaf::internal::nio::BufferFactory, 599, 600
- createBytesMessage
 - activemq::cmsutil::PooledSession, 2096, 2097
 - activemq::core::ActiveMQSession, 339, 340
 - cms::Session, 2367
- createCMSConnectionFactory
 - cms::ConnectionFactory, 956
- createCMSXAConnectionFactory
 - cms::XAConnectionFactory, 2919
- createCachedConsumer
 - activemq::cmsutil::PooledSession, 2097
- createCachedProducer
 - activemq::cmsutil::PooledSession, 2098
- createCharBuffer
 - decaf::internal::nio::BufferFactory, 600, 601
- createComposite
 - activemq::transport::failover::FailoverTransportFactory, 1319
 - activemq::transport::mock::MockTransportFactory, 1959
 - activemq::transport::tcp::TcpTransportFactory, 2698
 - activemq::transport::TransportFactory, 2799
- createConnection
 - activemq::cmsutil::CmsAccessor, 821
 - activemq::core::ActiveMQConnectionFactory, 217, 218
 - cms::ConnectionFactory, 957, 958
- createConsumer
 - activemq::cmsutil::PooledSession, 2098, 2099
 - activemq::core::ActiveMQSession, 340, 341
 - cms::Session, 2368, 2369
- createDestination
 - activemq::commands::ActiveMQDestination, 249
- createDoubleBuffer
 - decaf::internal::nio::BufferFactory, 602, 603
- createDurableConsumer
 - activemq::cmsutil::PooledSession, 2100
 - activemq::core::ActiveMQSession, 342
 - cms::Session, 2370
- createFloatBuffer
 - decaf::internal::nio::BufferFactory, 603, 604
- createIntBuffer
 - decaf::internal::nio::BufferFactory, 604, 605
- createLongBuffer
 - decaf::internal::nio::BufferFactory, 606, 607
- createMapMessage
 - activemq::cmsutil::PooledSession, 2101
 - activemq::core::ActiveMQSession, 342
 - cms::Session, 2371
- createMessage
 - activemq::cmsutil::MessageCreator, 1881
 - activemq::cmsutil::PooledSession, 2101
 - activemq::core::ActiveMQSession, 343
 - cms::Session, 2371
- createMessageEOFException
 - activemq::util::CMSExceptionSupport, 829
- createMessageFormatException

- activemq::util::CMSException-
Support, 830
- createNegotiator
 - activemq::wireformat::openwire::-
OpenWireFormat, 2049
 - activemq::wireformat::stomp::Stomp-
WireFormat, 2598
 - activemq::wireformat::WireFormat,
2885
- createObject
 - activemq::wireformat::openwire-
::marshal::DataStreamMarshaller,
1120
 - activemq::wireformat::openwire-
::marshal::generated::Active-
MQBlobMessageMarshaller,
163
 - activemq::wireformat::openwire-
::marshal::generated::Active-
MQBytesMessageMarshaller,
184
 - activemq::wireformat::openwire-
::marshal::generated::Active-
MQMapMessageMarshaller,
285
 - activemq::wireformat::openwire-
::marshal::generated::Active-
MQMessageMarshaller, 292
 - activemq::wireformat::openwire-
::marshal::generated::Active-
MQObjectMessageMarshaller,
305
 - activemq::wireformat::openwire-
::marshal::generated::Active-
MQQueueMarshaller, 330
 - activemq::wireformat::openwire-
::marshal::generated::Active-
MQStreamMessageMarshaller,
376
 - activemq::wireformat::openwire-
::marshal::generated::Active-
MQTempQueueMarshaller, 393
 - activemq::wireformat::openwire-
::marshal::generated::Active-
MQTempTopicMarshaller, 402
 - activemq::wireformat::openwire-
::marshal::generated::ActiveM-
QTextMessageMarshaller, 411
 - activemq::wireformat::openwire-
::marshal::generated::Active-
MQTopicMarshaller, 420
- activemq::wireformat::openwire-
::marshal::generated::Broker-
IdMarshaller, 569
- activemq::wireformat::openwire-
::marshal::generated::Broker-
InfoMarshaller, 579
- activemq::wireformat::openwire-
::marshal::generated::Connection-
ControlMarshaller, 945
- activemq::wireformat::openwire-
::marshal::generated::Connection-
ErrorMarshaller, 952
- activemq::wireformat::openwire-
::marshal::generated::Connection-
IdMarshaller, 965
- activemq::wireformat::openwire-
::marshal::generated::Connection-
InfoMarshaller, 975
- activemq::wireformat::openwire-
::marshal::generated::Consumer-
ControlMarshaller, 997
- activemq::wireformat::openwire-
::marshal::generated::Consumer-
IdMarshaller, 1006
- activemq::wireformat::openwire-
::marshal::generated::Consumer-
InfoMarshaller, 1019
- activemq::wireformat::openwire-
::marshal::generated::Control-
CommandMarshaller, 1027
- activemq::wireformat::openwire-
::marshal::generated::Data-
ArrayResponseMarshaller,
1073
- activemq::wireformat::openwire-
::marshal::generated::Data-
ResponseMarshaller, 1116
- activemq::wireformat::openwire-
::marshal::generated::Destination-
InfoMarshaller, 1219
- activemq::wireformat::openwire-
::marshal::generated::Discovery-
EventMarshaller, 1231
- activemq::wireformat::openwire-
::marshal::generated::Exception-
ResponseMarshaller, 1291
- activemq::wireformat::openwire-
::marshal::generated::Flush-
CommandMarshaller, 1384

- activemq::wireformat::openwire-
::marshal::generated::Integer-
ResponseMarshaller, 1520
- activemq::wireformat::openwire-
::marshal::generated::Journal-
QueueAckMarshaller, 1565
- activemq::wireformat::openwire-
::marshal::generated::Journal-
TopicAckMarshaller, 1574
- activemq::wireformat::openwire-
::marshal::generated::Journal-
TraceMarshaller, 1580
- activemq::wireformat::openwire-
::marshal::generated::Journal-
TransactionMarshaller, 1588
- activemq::wireformat::openwire-
::marshal::generated::Keep-
AliveInfoMarshaller, 1595
- activemq::wireformat::openwire-
::marshal::generated::Last-
PartialCommandMarshaller,
1609
- activemq::wireformat::openwire-
::marshal::generated::Local-
TransactionIdMarshaller, 1679
- activemq::wireformat::openwire-
::marshal::generated::Message-
AckMarshaller, 1874
- activemq::wireformat::openwire-
::marshal::generated::Message-
DispatchMarshaller, 1891
- activemq::wireformat::openwire-
::marshal::generated::Message-
DispatchNotificationMarshaller,
1900
- activemq::wireformat::openwire-
::marshal::generated::Message-
IdMarshaller, 1913
- activemq::wireformat::openwire-
::marshal::generated::Message-
PullMarshaller, 1945
- activemq::wireformat::openwire-
::marshal::generated::Network-
BridgeFilterMarshaller, 1975
- activemq::wireformat::openwire-
::marshal::generated::Partial-
CommandMarshaller, 2080
- activemq::wireformat::openwire-
::marshal::generated::Producer-
AckMarshaller, 2176
- activemq::wireformat::openwire-
::marshal::generated::Producer-
IdMarshaller, 2187
- activemq::wireformat::openwire-
::marshal::generated::Producer-
InfoMarshaller, 2196
- activemq::wireformat::openwire-
::marshal::generated::Remove-
InfoMarshaller, 2272
- activemq::wireformat::openwire-
::marshal::generated::Remove-
SubscriptionInfoMarshaller,
2281
- activemq::wireformat::openwire-
::marshal::generated::Replay-
CommandMarshaller, 2288
- activemq::wireformat::openwire-
::marshal::generated::Response-
Marshaller, 2308
- activemq::wireformat::openwire-
::marshal::generated::Session-
IdMarshaller, 2383
- activemq::wireformat::openwire-
::marshal::generated::Session-
InfoMarshaller, 2391
- activemq::wireformat::openwire-
::marshal::generated::Shutdown-
InfoMarshaller, 2435
- activemq::wireformat::openwire-
::marshal::generated::Subscription-
InfoMarshaller, 2636
- activemq::wireformat::openwire-
::marshal::generated::Transaction-
InfoMarshaller, 2779
- activemq::wireformat::openwire-
::marshal::generated::Wire-
FormatInfoMarshaller, 2901
- activemq::wireformat::openwire-
::marshal::generated::XA-
TransactionIdMarshaller, 2943
- createProducer
 - activemq::cmsutil::PooledSession,
2101
 - activemq::core::ActiveMQSession,
343
 - cms::Session, 2371
- createQueryString
 - activemq::util::URISupport, 2854
- createQueue
 - activemq::cmsutil::PooledSession,

- 2102
- activemq::core::ActiveMQSession, 343
- cms::Session, 2372
- createRemoveCommand
 - activemq::commands::ConnectionInfo, 970
 - activemq::commands::ConsumerInfo, 1012
 - activemq::commands::ProducerInfo, 2192
 - activemq::commands::SessionInfo, 2388
- createServerSocket
 - decaf::internal::net::DefaultServerSocketFactory, 1157, 1158
 - decaf::internal::net::ssl::DefaultSSLServerSocketFactory, 1167, 1168
 - decaf::internal::net::ssl::openssl::OpenSSLServerSocketFactory, 2011–2013
 - decaf::net::ServerSocketFactory, 2353, 2354
- createSession
 - activemq::cmsutil::CmsAccessor, 821
 - activemq::core::ActiveMQConnection, 196
 - activemq::core::ActiveMQXAConnection, 433
 - cms::Connection, 935, 936
- createShortBuffer
 - decaf::internal::nio::BufferFactory, 607, 608
- createSocket
 - activemq::transport::tcp::SslTransport, 2526
 - activemq::transport::tcp::TcpTransport, 2695
 - decaf::internal::net::DefaultSocketFactory, 1161–1163
 - decaf::internal::net::ssl::DefaultSSLSocketFactory, 1173–1176
 - decaf::internal::net::ssl::openssl::OpenSSLSocketFactory, 2036–2038
 - decaf::net::SocketFactory, 2475–2477
 - decaf::net::ssl::SSLSocketFactory, 2523
- createSocketImpl
 - decaf::net::SocketImplFactory, 2488
- createStreamMessage
 - activemq::cmsutil::PooledSession, 2102
 - activemq::core::ActiveMQSession, 344
 - cms::Session, 2372
- createTemporaryName
 - activemq::commands::ActiveMQDestination, 250
- createTemporaryQueue
 - activemq::cmsutil::PooledSession, 2102
 - activemq::core::ActiveMQSession, 344
 - cms::Session, 2373
- createTemporaryTopic
 - activemq::cmsutil::PooledSession, 2103
 - activemq::core::ActiveMQSession, 344
 - cms::Session, 2373
- createTextMessage
 - activemq::cmsutil::PooledSession, 2103
 - activemq::core::ActiveMQSession, 344, 345
 - cms::Session, 2373, 2374
- createTopic
 - activemq::cmsutil::PooledSession, 2104
 - activemq::core::ActiveMQSession, 345
 - cms::Session, 2374
- createWireFormat
 - activemq::transport::AbstractTransportFactory, 155
 - activemq::wireformat::openwire::OpenWireFormatFactory, 2060
 - activemq::wireformat::stomp::StompWireFormatFactory, 2601
 - activemq::wireformat::WireFormatFactory, 2889
- createXAConnection
 - activemq::core::ActiveMQXAConnectionFactory, 435, 436
 - cms::XAConnectionFactory, 2920
- createXASession

- activemq::core::ActiveMQXAConnection, 433
- cms::XAConnection, 2918
- criticalSection
 - decaf::util::concurrent::Condition-Handle, 928
- ct_data
 - deflate.h, 3145
- ct_data_s, 1068
 - code, 1068
 - dad, 1068
 - dl, 1068
 - fc, 1068
 - freq, 1068
 - len, 1068
- currentThread
 - decaf::lang::Thread, 2708
- currentTimeMillis
 - decaf::lang::System, 2672
- d_buf
 - internal_state, 1524
- d_code
 - deflate.h, 3144
- d_desc
 - internal_state, 1525
- dad
 - ct_data_s, 1068
- data
 - activemq::commands::DataArray-Response, 1071
 - activemq::commands::DataResponse, 1115
 - activemq::commands::Partial-Command, 2079
- data_type
 - z_stream_s, 2952
- dataStructure
 - activemq::commands::Message, 1837
- debug
 - decaf::util::logging::Logger, 1697
 - decaf::util::logging::SimpleLogger, 2443
- decaf, 83
- decaf/lang/exceptions/ExceptionDefines.h
 - DECAF_CATCHALL_NOTHROW, 3011
 - DECAF_CATCHALL_THROW, 3011
- DECAF_CATCH_EXCEPTION_CO-NVERT, 3010
- DECAF_CATCH_NOTHROW, 3010
- DECAF_CATCH_RETHROW, 3010
- decaf/util/Config.h
 - DECAF_API, 3040
 - DECAF_UNUSED, 3040
 - HAVE_PTHREAD_H, 3040
 - HAVE_UUID_T, 3040
 - HAVE_UUID_UUID_H, 3040
- decaf::internal, 83
- decaf::internal::AprPool, 444
 - ~AprPool, 445
 - AprPool, 445
 - cleanup, 445
 - getAprPool, 445
 - getGlobalPool, 445
- decaf::internal::DecafRuntime, 1143
 - ~DecafRuntime, 1143
 - DecafRuntime, 1143
 - getGlobalLock, 1143
 - getGlobalPool, 1144
- decaf::internal::io, 84
- decaf::internal::io::StandardErrorOutput-Stream, 2528
 - ~StandardErrorOutputStream, 2529
 - StandardErrorOutputStream, 2529
 - close, 2529
 - doWriteArrayBounded, 2530
 - doWriteByte, 2530
 - flush, 2530
- decaf::internal::io::StandardInputStream, 2530
 - ~StandardInputStream, 2531
 - StandardInputStream, 2531
 - available, 2531
 - doReadByte, 2531
- decaf::internal::io::StandardOutput-Stream, 2532
 - ~StandardOutputStream, 2532
 - StandardOutputStream, 2532
 - close, 2532
 - doWriteArrayBounded, 2533
 - doWriteByte, 2533
 - flush, 2533
- decaf::internal::net, 84
- decaf::internal::net::DefaultServerSocket-Factory, 1155
 - ~DefaultServerSocketFactory, 1157
 - DefaultServerSocketFactory, 1156

- createServerSocket, 1157, 1158
- decaf::internal::net::DefaultSocketFactory, 1159
 - ~DefaultSocketFactory, 1161
 - DefaultSocketFactory, 1161
 - createSocket, 1161–1163
- decaf::internal::net::Network, 1968
 - ~Network, 1969
 - Network, 1969
 - addAsResource, 1969
 - addNetworkResource, 1969
 - addShutdownTask, 1970
 - getNetworkRuntime, 1970
 - getRuntimeLock, 1970
 - initializeNetworking, 1971
 - shutdownNetworking, 1971
- decaf::internal::net::SocketFileDescriptor, 2478
 - ~SocketFileDescriptor, 2479
 - SocketFileDescriptor, 2479
 - getValue, 2479
- decaf::internal::net::URIEncoderDecoder, 2841
 - ~URIEncoderDecoder, 2842
 - URIEncoderDecoder, 2842
 - decode, 2842
 - encodeOthers, 2842
 - quoteIllegal, 2843
 - validate, 2843
 - validateSimple, 2843
- decaf::internal::net::URIHelper, 2844
 - ~URIHelper, 2846
 - URIHelper, 2845, 2846
 - isValidDomainName, 2846
 - isValidHexChar, 2846
 - isValidHost, 2846
 - isValidIP4Word, 2847
 - isValidIP6Address, 2847
 - isValidIPv4Address, 2847
 - parseAuthority, 2848
 - parseURI, 2848
 - validateAuthority, 2848
 - validateFragment, 2849
 - validatePath, 2849
 - validateQuery, 2849
 - validateScheme, 2850
 - validateSsp, 2850
 - validateUserInfo, 2850
- decaf::internal::net::URIType, 2860
 - ~URIType, 2862
- URIType, 2862
- getAuthority, 2862
- getFragment, 2862
- getHost, 2863
- getPath, 2863
- getPort, 2863
- getQuery, 2863
- getScheme, 2863
- getSchemeSpecificPart, 2863
- getSource, 2864
- getUserInfo, 2864
- isAbsolute, 2864
- isOpaque, 2864
- isServerAuthority, 2864
- isValid, 2865
- setAbsolute, 2865
- setAuthority, 2865
- setFragment, 2865
- setHost, 2865
- setOpaque, 2866
- setPath, 2866
- setPort, 2866
- setQuery, 2866
- setScheme, 2866
- setSchemeSpecificPart, 2867
- setServerAuthority, 2867
- setSource, 2867
- setUserInfo, 2867
- setValid, 2868
- decaf::internal::net::ssl, 85
- decaf::internal::net::ssl::DefaultSSLContext, 1164
 - ~DefaultSSLContext, 1164
 - DefaultSSLContext, 1164
 - getContext, 1165
- decaf::internal::net::ssl::DefaultSSLServerSocketFactory, 1165
 - ~DefaultSSLServerSocketFactory, 1167
 - DefaultSSLServerSocketFactory, 1167
 - createServerSocket, 1167, 1168
 - getDefaultCipherSuites, 1169
 - getSupportedCipherSuites, 1169
- decaf::internal::net::ssl::DefaultSSLSocketFactory, 1170
 - ~DefaultSSLSocketFactory, 1173
 - DefaultSSLSocketFactory, 1173
 - createSocket, 1173–1176
 - getDefaultCipherSuites, 1177

- getSupportedCipherSuites, 1177
- decaf::internal::net::ssl::openssl, 85
- decaf::internal::net::ssl::openssl::OpenS-
 - SLContextSpi, 1998
 - ~OpenSSLContextSpi, 1999
 - OpenSSLContextSpi, 1999
 - OpenSSLSocket, 2001
 - OpenSSLSocketFactory, 2001
 - providerGetServerSocketFactory, 1999
 - providerGetSocketFactory, 2000
 - providerInit, 2000
- decaf::internal::net::ssl::openssl::OpenS-
 - SLParameters, 2001
 - ~OpenSSLParameters, 2002
 - clone, 2002
 - getEnabledCipherSuites, 2002
 - getEnabledProtocols, 2002
 - getNeedClientAuth, 2002
 - getSupportedCipherSuites, 2002
 - getSupportedProtocols, 2002
 - getUseClientMode, 2002
 - getWantClientAuth, 2002
 - setEnabledCipherSuites, 2003
 - setEnabledProtocols, 2003
 - setNeedClientAuth, 2003
 - setUseClientMode, 2003
 - setWantClientAuth, 2003
- decaf::internal::net::ssl::openssl::OpenS-
 - SLServerSocket, 2003
 - ~OpenSSLServerSocket, 2005
 - OpenSSLServerSocket, 2005
 - accept, 2006
 - getEnabledCipherSuites, 2006
 - getEnabledProtocols, 2006
 - getNeedClientAuth, 2007
 - getSupportedCipherSuites, 2007
 - getSupportedProtocols, 2007
 - getWantClientAuth, 2007
 - setEnabledCipherSuites, 2008
 - setEnabledProtocols, 2008
 - setNeedClientAuth, 2008
 - setWantClientAuth, 2009
- decaf::internal::net::ssl::openssl::OpenS-
 - SLServerSocketFactory, 2009
 - ~OpenSSLServerSocketFactory, 2011
 - OpenSSLServerSocketFactory, 2011
 - createServerSocket, 2011–2013
 - getDefaultCipherSuites, 2013
 - getSupportedCipherSuites, 2014
- decaf::internal::net::ssl::openssl::OpenS-
 - SLSocket, 2014
 - ~OpenSSLSocket, 2020
 - OpenSSLSocket, 2019, 2020
 - available, 2020
 - close, 2020
 - connect, 2020
 - getEnabledCipherSuites, 2021
 - getEnabledProtocols, 2021
 - getInputStream, 2021
 - getNeedClientAuth, 2022
 - getOutputStream, 2022
 - getSupportedCipherSuites, 2023
 - getSupportedProtocols, 2023
 - getUseClientMode, 2023
 - getWantClientAuth, 2023
 - read, 2024
 - sendUrgentData, 2024
 - setEnabledCipherSuites, 2025
 - setEnabledProtocols, 2025
 - setNeedClientAuth, 2025
 - setOOBInline, 2026
 - setUseClientMode, 2026
 - setWantClientAuth, 2027
 - shutdownInput, 2027
 - shutdownOutput, 2027
 - startHandshake, 2028
 - write, 2028
- decaf::internal::net::ssl::openssl::OpenS-
 - SLSocketException, 2029
 - ~OpenSSLSocketException, 2032
 - OpenSSLSocketException, 2030, 2031
 - clone, 2032
 - getErrorString, 2032
- decaf::internal::net::ssl::openssl::OpenS-
 - SLSocketFactory, 2032
 - ~OpenSSLSocketFactory, 2036
 - OpenSSLSocketFactory, 2036
 - createSocket, 2036–2038
 - getDefaultCipherSuites, 2039
 - getSupportedCipherSuites, 2039
- decaf::internal::net::ssl::openssl::OpenS-
 - SLSocketInputStream, 2040
 - ~OpenSSLSocketInputStream, 2041
 - OpenSSLSocketInputStream, 2041
 - available, 2041
 - close, 2042

- doReadArrayBounded, 2042
- doReadByte, 2042
- skip, 2042
- decaf::internal::net::ssl::openssl::OpenSSLSocketOutputStream, 2043
- ~OpenSSLSocketOutputStream, 2044
- OpenSSLSocketOutputStream, 2044
- close, 2044
- doWriteArrayBounded, 2045
- doWriteByte, 2045
- decaf::internal::net::tcp, 86
- decaf::internal::net::tcp::TcpSocket, 2678
 - ~TcpSocket, 2681
 - TcpSocket, 2681
 - accept, 2682
 - available, 2682
 - bind, 2682
 - checkResult, 2682
 - close, 2682
 - connect, 2683
 - create, 2683
 - getInputStream, 2683
 - getLocalAddress, 2684
 - getOption, 2684
 - getOutputStream, 2684
 - getSocketHandle, 2685
 - isClosed, 2685
 - isConnected, 2685
 - listen, 2685
 - read, 2686
 - setOption, 2686
 - shutdownInput, 2687
 - shutdownOutput, 2687
 - write, 2687
- decaf::internal::net::tcp::TcpSocketInputStream, 2688
 - ~TcpSocketInputStream, 2689
 - TcpSocketInputStream, 2689
 - available, 2689
 - close, 2690
 - doReadArrayBounded, 2690
 - doReadByte, 2690
 - skip, 2690
- decaf::internal::net::tcp::TcpSocketOutputStream, 2691
 - ~TcpSocketOutputStream, 2692
 - TcpSocketOutputStream, 2692
 - close, 2692
 - doWriteArrayBounded, 2692
 - doWriteByte, 2692
- decaf::internal::nio, 86
- decaf::internal::nio::BufferFactory, 597
 - ~BufferFactory, 599
 - createByteBuffer, 599, 600
 - createCharBuffer, 600, 601
 - createDoubleBuffer, 602, 603
 - createFloatBuffer, 603, 604
 - createIntBuffer, 604, 605
 - createLongBuffer, 606, 607
 - createShortBuffer, 607, 608
- decaf::internal::nio::ByteBuffer, 646
 - ~ByteBuffer, 660
 - ByteBuffer, 659, 660
 - array, 660
 - arrayOffset, 661
 - asCharBuffer, 661
 - asDoubleBuffer, 662
 - asFloatBuffer, 662
 - asIntBuffer, 663
 - asLongBuffer, 663
 - asReadOnlyBuffer, 663
 - asShortBuffer, 664
 - compact, 664
 - duplicate, 665
 - get, 665
 - getChar, 666
 - getDouble, 667
 - getFloat, 668
 - getInt, 668, 669
 - getLong, 669, 670
 - getShort, 670
 - hasArray, 671
 - isReadOnly, 671
 - put, 671, 672
 - putChar, 672, 673
 - putDouble, 673, 674
 - putFloat, 675
 - putInt, 676
 - putLong, 677
 - putShort, 678
 - setReadOnly, 679
 - slice, 679
- decaf::internal::nio::CharArrayBuffer, 773
 - ~CharArrayBuffer, 779
 - CharArrayBuffer, 777, 778
 - _array, 785
 - array, 779
 - arrayOffset, 779
 - asReadOnlyBuffer, 780

- compact, 780
- duplicate, 781
- get, 781
- hasArray, 782
- isReadOnly, 782
- length, 785
- offset, 785
- put, 782, 783
- readOnly, 785
- setReadOnly, 783
- slice, 784
- subSequence, 784
- decaf::internal::nio::DoubleArrayBuffer,
 - 1248
 - ~DoubleArrayBuffer, 1253
 - DoubleArrayBuffer, 1252, 1253
 - array, 1254
 - arrayOffset, 1254
 - asReadOnlyBuffer, 1255
 - compact, 1255
 - duplicate, 1255
 - get, 1256
 - hasArray, 1257
 - isReadOnly, 1257
 - put, 1257, 1258
 - setReadOnly, 1258
 - slice, 1258
- decaf::internal::nio::FloatArrayBuffer,
 - 1357
 - ~FloatArrayBuffer, 1362
 - FloatArrayBuffer, 1360–1362
 - array, 1362
 - arrayOffset, 1363
 - asReadOnlyBuffer, 1363
 - compact, 1363
 - duplicate, 1364
 - get, 1364, 1365
 - hasArray, 1365
 - isReadOnly, 1365
 - put, 1366
 - setReadOnly, 1367
 - slice, 1367
- decaf::internal::nio::IntArrayBuffer, 1477
 - ~IntArrayBuffer, 1482
 - IntArrayBuffer, 1481, 1482
 - array, 1482
 - arrayOffset, 1483
 - asReadOnlyBuffer, 1483
 - compact, 1484
 - duplicate, 1484
 - get, 1485
 - hasArray, 1485
 - isReadOnly, 1486
 - put, 1486, 1487
 - setReadOnly, 1487
 - slice, 1487
- decaf::internal::nio::LongArrayBuffer,
 - 1742
 - ~LongArrayBuffer, 1747
 - LongArrayBuffer, 1745–1747
 - array, 1747
 - arrayOffset, 1747
 - asReadOnlyBuffer, 1748
 - compact, 1748
 - duplicate, 1749
 - get, 1749, 1750
 - hasArray, 1750
 - isReadOnly, 1750
 - put, 1751
 - setReadOnly, 1752
 - slice, 1752
- decaf::internal::nio::ShortArrayBuffer,
 - 2408
 - ~ShortArrayBuffer, 2413
 - ShortArrayBuffer, 2412, 2413
 - array, 2414
 - arrayOffset, 2414
 - asReadOnlyBuffer, 2415
 - compact, 2415
 - duplicate, 2415
 - get, 2416
 - hasArray, 2417
 - isReadOnly, 2417
 - put, 2417, 2418
 - setReadOnly, 2418
 - slice, 2418
- decaf::internal::security, 87
- decaf::internal::security::SecureRandom-
 - Impl, 2325
 - ~SecureRandomImpl, 2327
 - SecureRandomImpl, 2327
 - providerGenerateSeed, 2327
 - providerNextBytes, 2327, 2328
 - providerSetSeed, 2328
- decaf::internal::util, 87
- decaf::internal::util::ByteArrayAdapter,
 - 624
 - ~ByteArrayAdapter, 630
 - ByteArrayAdapter, 627–629
 - clear, 630

- get, 630
- getByteArray, 630
- getCapacity, 631
- getChar, 631
- getCharArray, 631
- getCharCapacity, 631
- getDouble, 632
- getDoubleArray, 632
- getDoubleAt, 632
- getDoubleCapacity, 633
- getFloat, 633
- getFloatArray, 633
- getFloatAt, 634
- getFloatCapacity, 634
- getInt, 634
- getIntArray, 635
- getIntAt, 635
- getIntCapacity, 635
- getLong, 636
- getLongArray, 636
- getLongAt, 636
- getLongCapacity, 637
- getShort, 637
- getShortArray, 637
- getShortAt, 638
- getShortCapacity, 638
- operator[], 638, 639
- put, 639
- putChar, 639
- putDouble, 640
- putDoubleAt, 640
- putFloat, 641
- putFloatAt, 641
- putInt, 642
- putIntAt, 642
- putLong, 642
- putLongAt, 643
- putShort, 643
- putShortAt, 644
- read, 644
- resize, 645
- write, 645
- decaf::internal::util::GenericResource
 - ~GenericResource, 1398
 - GenericResource, 1398
 - getManaged, 1398
 - setManaged, 1398
- decaf::internal::util::GenericResource< T
 - >, 1397
- decaf::internal::util::HexStringParser,
 - 1406
 - ~HexStringParser, 1406
 - HexStringParser, 1406
 - parse, 1406
 - parseDouble, 1407
 - parseFloat, 1407
- decaf::internal::util::Resource, 2293
 - ~Resource, 2293
- decaf::internal::util::ResourceLifecycle-
 - Manager, 2297
 - ~ResourceLifecycleManager, 2297
 - ResourceLifecycleManager, 2297
 - addResource, 2297
 - destroyResources, 2297
- decaf::internal::util::TimerTaskHeap, 2753
 - ~TimerTaskHeap, 2754
 - TimerTaskHeap, 2754
 - adjustMinimum, 2754
 - decaf::util::TimerTask, 2753
 - deleteIfCancelled, 2754
 - find, 2754
 - insert, 2755
 - isEmpty, 2755
 - peek, 2755
 - remove, 2755
 - reset, 2755
 - size, 2755
- decaf::internal::util::concurrent, 87
- decaf::internal::util::concurrent::Condition-
 - Impl, 929
 - create, 929
 - destroy, 930
 - notify, 930
 - notifyAll, 930
 - wait, 930
- decaf::internal::util::concurrent::Mutex-
 - Impl, 1966
 - create, 1967
 - destroy, 1967
 - lock, 1967
 - trylock, 1967
 - unlock, 1968
- decaf::internal::util::concurrent::Synchronizable-
 - Impl, 2650
 - ~SynchronizableImpl, 2651
 - SynchronizableImpl, 2651
 - lock, 2651
 - notify, 2651
 - notifyAll, 2651

- tryLock, 2652
- unlock, 2652
- wait, 2652, 2653
- decaf::internal::util::concurrent::Transfer-
Queue
 - ~TransferQueue, 2787
 - TransferQueue, 2787
 - transfer, 2788
- decaf::internal::util::concurrent::Transfer-
Queue< E >, 2787
- decaf::internal::util::concurrent::Transfer-
Stack
 - ~TransferStack, 2789
 - TransferStack, 2789
 - transfer, 2789, 2790
- decaf::internal::util::concurrent::Transfer-
Stack< E >, 2789
- decaf::internal::util::concurrent::Transferer<
E >, 2786
- decaf::io, 88
- decaf::io::BlockingByteArrayInputStream, 534
 - ~BlockingByteArrayInputStream, 536
 - BlockingByteArrayInputStream, 536
 - available, 536
 - close, 537
 - doReadArrayBounded, 537
 - doReadByte, 537
 - setByteArray, 537
 - skip, 537
- decaf::io::BufferedInputStream, 589
 - ~BufferedInputStream, 592
 - BufferedInputStream, 591
 - available, 592
 - close, 592
 - doReadArrayBounded, 593
 - doReadByte, 593
 - mark, 593
 - markSupported, 593
 - reset, 593
 - skip, 594
- decaf::io::BufferedOutputStream, 595
 - ~BufferedOutputStream, 596
 - BufferedOutputStream, 596
 - doWriteArray, 596
 - doWriteArrayBounded, 596
 - doWriteByte, 596
 - flush, 597
- decaf::io::ByteArrayInputStream, 679
 - ~ByteArrayInputStream, 683
 - ByteArrayInputStream, 682, 683
 - available, 683
 - doReadArrayBounded, 684
 - doReadByte, 684
 - mark, 684
 - markSupported, 684
 - reset, 685
 - setByteArray, 685, 686
 - skip, 686
- decaf::io::ByteArrayOutputStream, 687
 - ~ByteArrayOutputStream, 688
 - ByteArrayOutputStream, 688
 - doWriteArrayBounded, 689
 - doWriteByte, 689
 - reset, 689
 - size, 689
 - toByteArray, 689
 - toString, 689
 - writeTo, 690
- decaf::io::Closeable, 816
 - ~Closeable, 817
 - close, 817
- decaf::io::DataInput, 1086
 - ~DataInput, 1087
 - readBoolean, 1087
 - readByte, 1088
 - readChar, 1088
 - readDouble, 1088
 - readFloat, 1089
 - readFully, 1089
 - readInt, 1090
 - readLine, 1091
 - readLong, 1091
 - readShort, 1091
 - readString, 1092
 - readUTF, 1093
 - readUnsignedByte, 1092
 - readUnsignedShort, 1093
 - skipBytes, 1093
- decaf::io::DataInputStream, 1094
 - ~DataInputStream, 1096
 - DataInputStream, 1096
 - readBoolean, 1096
 - readByte, 1096
 - readChar, 1097
 - readDouble, 1097
 - readFloat, 1097
 - readFully, 1098
 - readInt, 1099

- readLine, 1099
- readLong, 1100
- readShort, 1100
- readString, 1101
- readUTF, 1102
- readUnsignedByte, 1101
- readUnsignedShort, 1101
- skipBytes, 1102
- decaf::io::DataOutput, 1103
 - ~DataOutput, 1104
 - writeBoolean, 1104
 - writeByte, 1105
 - writeBytes, 1105
 - writeChar, 1105
 - writeChars, 1106
 - writeDouble, 1106
 - writeFloat, 1106
 - writeInt, 1107
 - writeLong, 1107
 - writeShort, 1107
 - writeUTF, 1108
 - writeUnsignedShort, 1108
- decaf::io::DataOutputStream, 1109
 - ~DataOutputStream, 1110
 - DataOutputStream, 1110
 - buffer, 1112
 - doWriteArrayBounded, 1110
 - doWriteByte, 1110
 - size, 1111
 - writeBoolean, 1111
 - writeByte, 1111
 - writeBytes, 1111
 - writeChar, 1111
 - writeChars, 1111
 - writeDouble, 1111
 - writeFloat, 1111
 - writeInt, 1111
 - writeLong, 1111
 - writeShort, 1111
 - writeUTF, 1112
 - writeUnsignedShort, 1112
 - written, 1112
- decaf::io::EOFException, 1275
 - ~EOFException, 1277
 - EOFException, 1275, 1276
 - clone, 1277
- decaf::io::FileDescriptor, 1330
 - ~FileDescriptor, 1331
 - FileDescriptor, 1331
 - descriptor, 1332
 - err, 1332
 - in, 1332
 - out, 1332
 - readonly, 1332
 - sync, 1332
 - valid, 1332
- decaf::io::FilterInputStream, 1334
 - ~FilterInputStream, 1336
 - FilterInputStream, 1336
 - available, 1337
 - close, 1337
 - closed, 1340
 - doReadArray, 1337
 - doReadArrayBounded, 1337
 - doReadByte, 1338
 - inputStream, 1340
 - isClosed, 1338
 - mark, 1338
 - markSupported, 1338
 - own, 1340
 - reset, 1339
 - skip, 1339
- decaf::io::FilterOutputStream, 1341
 - ~FilterOutputStream, 1342
 - FilterOutputStream, 1342
 - close, 1342
 - closed, 1344
 - doWriteArray, 1343
 - doWriteArrayBounded, 1343
 - doWriteByte, 1343
 - flush, 1343
 - isClosed, 1344
 - outputStream, 1344
 - own, 1344
 - toString, 1344
- decaf::io::Flushable, 1380
 - ~Flushable, 1380
 - flush, 1380
- decaf::io::IOException, 1545
 - ~IOException, 1547
 - IOException, 1546, 1547
 - clone, 1547
- decaf::io::InputStream, 1464
 - ~InputStream, 1466
 - InputStream, 1466
 - available, 1466
 - close, 1466
 - doReadArray, 1467
 - doReadArrayBounded, 1467
 - doReadByte, 1467

- lock, 1467
- mark, 1468
- markSupported, 1468
- notify, 1468
- notifyAll, 1469
- read, 1469, 1470
- reset, 1471
- skip, 1472
- toString, 1472
- tryLock, 1473
- unlock, 1473
- wait, 1473, 1474
- decaf::io::InputStreamReader, 1475
 - ~InputStreamReader, 1476
 - InputStreamReader, 1476
 - checkClosed, 1476
 - close, 1476
 - doReadArrayBounded, 1476
 - ready, 1476
- decaf::io::InterruptedIOException, 1531
 - ~InterruptedIOException, 1533
 - InterruptedIOException, 1532, 1533
 - clone, 1533
- decaf::io::OutputStream, 2066
 - ~OutputStream, 2068
 - OutputStream, 2068
 - close, 2068
 - doWriteArray, 2068
 - doWriteArrayBounded, 2068
 - doWriteByte, 2069
 - flush, 2069
 - lock, 2069
 - notify, 2069
 - notifyAll, 2070
 - toString, 2070
 - tryLock, 2070
 - unlock, 2071
 - wait, 2071
 - write, 2072, 2073
- decaf::io::OutputStreamWriter, 2074
 - ~OutputStreamWriter, 2075
 - OutputStreamWriter, 2075
 - checkClosed, 2075
 - close, 2075
 - doWriteArrayBounded, 2075
 - flush, 2075
- decaf::io::PushbackInputStream, 2214
 - ~PushbackInputStream, 2217
 - PushbackInputStream, 2216
 - available, 2217
 - doReadArrayBounded, 2217
 - doReadByte, 2217
 - mark, 2218
 - markSupported, 2218
 - reset, 2218
 - skip, 2219
 - unread, 2220
- decaf::io::Reader, 2237
 - ~Reader, 2239
 - Reader, 2239
 - doReadArray, 2239
 - doReadArrayBounded, 2239
 - doReadChar, 2239
 - doReadCharBuffer, 2239
 - doReadVector, 2239
 - mark, 2239
 - markSupported, 2240
 - read, 2240–2242
 - ready, 2242
 - reset, 2243
 - skip, 2243
- decaf::io::UTFDataFormatException, 2874
 - ~UTFDataFormatException, 2876
 - UTFDataFormatException, 2875, 2876
 - clone, 2876
- decaf::io::UnsupportedEncodingException, 2821
 - ~UnsupportedEncodingException, 2824
 - UnsupportedEncodingException, 2822, 2823
 - clone, 2824
- decaf::io::Writer, 2908
 - ~Writer, 2910
 - Writer, 2910
 - append, 2910
 - doAppendChar, 2911
 - doAppendCharSequence, 2911
 - doAppendCharSequenceStartEnd, 2911
 - doWriteArray, 2911
 - doWriteArrayBounded, 2911
 - doWriteChar, 2911
 - doWriteString, 2912
 - doWriteStringBounded, 2912
 - doWriteVector, 2912
 - write, 2912, 2913
- decaf::lang, 89

- operator==, 91, 92
- decaf::lang::Appendable, 442
 - ~Appendable, 442
 - append, 442, 443
- decaf::lang::ArrayPointer
 - ~ArrayPointer, 466
 - ArrayPointer, 464, 465
 - ConstReferenceType, 464
 - CounterType, 464
 - PointerType, 464
 - ReferenceType, 464
 - clone, 466
 - get, 466
 - length, 467
 - operator=, 467, 468
 - operator==, 468–470
 - operator[], 468
 - release, 468
 - reset, 469
 - swap, 469
- decaf::lang::ArrayPointer< T, REFCOUNT >, 462
- decaf::lang::ArrayPointerComparator
 - ~ArrayPointerComparator, 471
 - compare, 471
 - operator(), 471
- decaf::lang::ArrayPointerComparator< T, R >, 470
- decaf::lang::Boolean, 545
 - ~Boolean, 546
 - Boolean, 546
 - _FALSE, 550
 - _TRUE, 550
 - booleanValue, 546
 - compareTo, 547
 - equals, 547
 - operator<, 548
 - operator==, 548, 549
 - parseBoolean, 549
 - toString, 549
 - valueOf, 550
- decaf::lang::Byte, 614
 - ~Byte, 616
 - Byte, 616
 - MAX_VALUE, 623
 - MIN_VALUE, 623
 - SIZE, 623
 - byteValue, 616
 - compareTo, 616, 617
 - decode, 617
 - doubleValue, 618
 - equals, 618
 - floatValue, 618
 - intValue, 618
 - longValue, 618
 - operator<, 619
 - operator==, 619, 620
 - parseByte, 620, 621
 - shortValue, 621
 - toString, 621, 622
 - valueOf, 622, 623
- decaf::lang::CharSequence, 803
 - ~CharSequence, 804
 - charAt, 804
 - length, 804
 - subSequence, 804
 - toString, 805
- decaf::lang::Character, 765
 - Character, 766
 - MAX_RADIX, 772
 - MAX_VALUE, 772
 - MIN_RADIX, 772
 - MIN_VALUE, 773
 - SIZE, 773
 - byteValue, 767
 - compareTo, 767
 - digit, 767
 - doubleValue, 768
 - equals, 768
 - floatValue, 768
 - intValue, 769
 - isDigit, 769
 - isISOControl, 769
 - isLetter, 769
 - isLetterOrDigit, 769
 - isLowerCase, 770
 - isUpperCase, 770
 - isWhitespace, 770
 - longValue, 770
 - operator<, 770, 771
 - operator==, 771
 - shortValue, 772
 - toString, 772
 - valueOf, 772
- decaf::lang::Comparable
 - ~Comparable, 886
 - compareTo, 886
 - equals, 887
 - operator<, 887
 - operator==, 888

- decaf::lang::Comparable< T >, 885
- decaf::lang::DYNAMIC_CAST_TOKEN, 1271
- decaf::lang::Double, 1235
 - ~Double, 1238
 - Double, 1238
 - MAX_VALUE, 1248
 - MIN_VALUE, 1248
 - NEGATIVE_INFINITY, 1248
 - NaN, 1248
 - POSITIVE_INFINITY, 1248
 - SIZE, 1248
 - byteValue, 1238
 - compare, 1238
 - compareTo, 1239
 - doubleToLongBits, 1239
 - doubleToRawLongBits, 1240
 - doubleValue, 1241
 - equals, 1241
 - floatValue, 1241
 - intValue, 1242
 - isInfinite, 1242
 - isNaN, 1242
 - longBitsToDouble, 1243
 - longValue, 1243
 - operator<, 1243, 1244
 - operator==, 1244
 - parseDouble, 1245
 - shortValue, 1245
 - toHexString, 1245
 - toString, 1246
 - valueOf, 1247
- decaf::lang::Exception, 1279
 - ~Exception, 1282
 - Exception, 1281, 1282
 - buildMessage, 1282
 - cause, 1286
 - clone, 1282
 - getCause, 1283
 - getMessage, 1284
 - getStackTrace, 1284
 - getStackTraceString, 1284
 - initCause, 1284
 - message, 1286
 - operator=, 1285
 - printStackTrace, 1285
 - setMark, 1285
 - setMessage, 1285
 - setStackTrace, 1286
 - stackTrace, 1286
 - what, 1286
- decaf::lang::Float, 1344
 - ~Float, 1347
 - Float, 1346, 1347
 - MAX_VALUE, 1356
 - MIN_VALUE, 1356
 - NEGATIVE_INFINITY, 1356
 - NaN, 1356
 - POSITIVE_INFINITY, 1356
 - SIZE, 1357
 - byteValue, 1347
 - compare, 1347
 - compareTo, 1348
 - doubleValue, 1348
 - equals, 1348, 1349
 - floatToIntBits, 1349
 - floatToRawIntBits, 1349
 - floatValue, 1350
 - intBitsToFloat, 1350
 - intValue, 1351
 - isInfinite, 1351
 - isNaN, 1351
 - longValue, 1352
 - operator<, 1352
 - operator==, 1353
 - parseFloat, 1353
 - shortValue, 1354
 - toHexString, 1354
 - toString, 1355
 - valueOf, 1355, 1356
- decaf::lang::Integer, 1500
 - ~Integer, 1503
 - Integer, 1502
 - MAX_VALUE, 1516
 - MIN_VALUE, 1516
 - SIZE, 1516
 - bitCount, 1503
 - byteValue, 1503
 - compareTo, 1503
 - decode, 1504
 - doubleValue, 1504
 - equals, 1505
 - floatValue, 1505
 - highestOneBit, 1505
 - intValue, 1506
 - longValue, 1506
 - lowestOneBit, 1506
 - numberOfLeadingZeros, 1506
 - numberOfTrailingZeros, 1507
 - operator<, 1507, 1508

- operator==, 1508
- parseInt, 1509
- reverse, 1510
- reverseBytes, 1510
- rotateLeft, 1510
- rotateRight, 1511
- shortValue, 1511
- signum, 1512
- toBinaryString, 1512
- toHexString, 1512
- toOctalString, 1513
- toString, 1513, 1514
- valueOf, 1514, 1515
- decaf::lang::Iterable
 - ~Iterable, 1557
 - iterator, 1557, 1558
- decaf::lang::Iterable< E >, 1556
- decaf::lang::Long, 1726
 - ~Long, 1729
 - Long, 1728
 - MAX_VALUE, 1741
 - MIN_VALUE, 1741
 - SIZE, 1741
 - bitCount, 1729
 - byteValue, 1729
 - compareTo, 1729, 1730
 - decode, 1730
 - doubleValue, 1731
 - equals, 1731
 - floatValue, 1731
 - highestOneBit, 1731
 - intValue, 1732
 - longValue, 1732
 - lowestOneBit, 1732
 - numberOfLeadingZeros, 1733
 - numberOfTrailingZeros, 1733
 - operator<, 1734
 - operator==, 1734, 1735
 - parseLong, 1735, 1736
 - reverse, 1736
 - reverseBytes, 1736
 - rotateLeft, 1737
 - rotateRight, 1737
 - shortValue, 1737
 - signum, 1738
 - toBinaryString, 1738
 - toHexString, 1738
 - toOctalString, 1739
 - toString, 1739, 1740
 - valueOf, 1740, 1741
- decaf::lang::Math, 1800
 - ~Math, 1802
 - E, 1818
 - Math, 1802
 - PI, 1818
 - abs, 1802, 1803
 - ceil, 1804
 - floor, 1805
 - max, 1806, 1807
 - min, 1807–1810
 - pow, 1810
 - random, 1810
 - round, 1811, 1812
 - signum, 1812, 1813
 - sqrt, 1814
 - toDegrees, 1817
 - toRadians, 1817
- decaf::lang::Number, 1992
 - ~Number, 1992
 - byteValue, 1992
 - doubleValue, 1993
 - floatValue, 1993
 - intValue, 1993
 - longValue, 1994
 - shortValue, 1994
- decaf::lang::Pointer
 - ~Pointer, 2087
 - CounterType, 2085
 - Pointer, 2086, 2087
 - PointerType, 2085
 - ReferenceType, 2086
 - dynamicCast, 2087
 - get, 2088
 - operator*, 2088, 2089
 - operator->, 2089
 - operator=, 2089
 - operator==, 2089, 2091
 - release, 2089
 - reset, 2090
 - staticCast, 2090
 - swap, 2090
- decaf::lang::Pointer< T, REFCOUNTER >, 2083
- decaf::lang::PointerComparator
 - ~PointerComparator, 2092
 - compare, 2092
 - operator(), 2092
- decaf::lang::PointerComparator< T, R >, 2091
- decaf::lang::Readable, 2235

- ~Readable, 2236
- read, 2236
- decaf::lang::Runnable, 2312
 - ~Runnable, 2312
 - run, 2312
- decaf::lang::Runtime, 2313
 - ~Runtime, 2313
 - decaf::lang::System, 2676
 - decaf::lang::Thread, 2713
 - decaf::util::logging::LogManager, 1718
 - getRuntime, 2313
 - initializeRuntime, 2314
 - shutdownRuntime, 2314
- decaf::lang::STATIC_CAST_TOKEN, 2535
- decaf::lang::Short, 2398
 - ~Short, 2400
 - MAX_VALUE, 2408
 - MIN_VALUE, 2408
 - SIZE, 2408
 - Short, 2400
 - byteValue, 2400
 - compareTo, 2400, 2401
 - decode, 2401
 - doubleValue, 2402
 - equals, 2402
 - floatValue, 2402
 - intValue, 2403
 - longValue, 2403
 - operator<, 2403, 2404
 - operator==, 2404
 - parseShort, 2405
 - reverseBytes, 2406
 - shortValue, 2406
 - toString, 2406
 - valueOf, 2407
- decaf::lang::String, 2620
 - ~String, 2624
 - String, 2622, 2623
 - charAt, 2624
 - isEmpty, 2624
 - length, 2624
 - operator=, 2624
 - subSequence, 2624
 - toString, 2625
 - valueOf, 2625–2627
- decaf::lang::System, 2665
 - ~System, 2667
 - System, 2667
- arraycopy, 2668–2671
- availableProcessors, 2671
- clearProperty, 2671
- currentTimeMillis, 2672
- decaf::lang::Runtime, 2676
- getProperties, 2673
- getProperty, 2673, 2674
- getenv, 2672, 2673
- nanoTime, 2674
- setProperty, 2675
- setenv, 2674
- unsetenv, 2675
- decaf::lang::Thread, 2703
 - ~Thread, 2708
 - BLOCKED, 2707
 - MAX_PRIORITY, 2713
 - MIN_PRIORITY, 2713
 - NEW, 2707
 - NORM_PRIORITY, 2713
 - RUNNABLE, 2707
 - SLEEPING, 2707
 - State, 2707
 - TERMINATED, 2707
 - TIMED_WAITING, 2707
 - Thread, 2707, 2708
 - WAITING, 2707
 - currentThread, 2708
 - decaf::util::concurrent::Executors, 1302
 - decaf::lang::Runtime, 2713
 - decaf::util::concurrent::locks::Lock-Support, 2713
 - getId, 2708
 - getName, 2708
 - getPriority, 2709
 - getState, 2709
 - getUncaughtExceptionHandler, 2709
 - isAlive, 2709
 - isDaemon, 2709
 - join, 2709, 2710
 - run, 2710
 - setDaemon, 2711
 - setName, 2711
 - setPriority, 2711
 - setUncaughtExceptionHandler, 2711
 - sleep, 2712
 - start, 2712
 - toString, 2713
 - yield, 2713

- decaf::lang::Thread::UncaughtException-Handler, 2815
 - ~UncaughtExceptionHandler, 2816
 - uncaughtException, 2816
- decaf::lang::ThreadGroup, 2715
 - ~ThreadGroup, 2715
 - ThreadGroup, 2715
- decaf::lang::Throwable, 2732
 - ~Throwable, 2733
 - Throwable, 2733
 - clone, 2733
 - getCause, 2734
 - getMessage, 2735
 - getStackTrace, 2735
 - getStackTraceString, 2735
 - initCause, 2735
 - printStackTrace, 2736
 - setMark, 2736
- decaf::lang::exceptions, 92
- decaf::lang::exceptions::ClassCastException, 813
 - ~ClassCastException, 815
 - ClassCastException, 813, 814
 - clone, 815
- decaf::lang::exceptions::IllegalArgumentException-Exception, 1413
 - ~IllegalArgumentException, 1415
 - IllegalArgumentException, 1414, 1415
 - clone, 1415
- decaf::lang::exceptions::IllegalMonitorStateException, 1416
 - ~IllegalMonitorStateException, 1418
 - IllegalMonitorStateException, 1417, 1418
 - clone, 1418
- decaf::lang::exceptions::IllegalStateException, 1420
 - ~IllegalStateException, 1422
 - IllegalStateException, 1421, 1422
 - clone, 1422
- decaf::lang::exceptions::IllegalThreadStateException, 1423
 - ~IllegalThreadStateException, 1425
 - IllegalThreadStateException, 1423, 1424
 - clone, 1425
- decaf::lang::exceptions::IndexOutOfBoundsException, 1429
 - ~IndexOutOfBoundsException, 1431
 - IndexOutOfBoundsException, 1429, 1430
 - clone, 1431
- decaf::lang::exceptions::InterruptedException-Exception, 1529
 - ~InterruptedException, 1531
 - InterruptedException, 1529, 1530
 - clone, 1531
- decaf::lang::exceptions::InvalidStateException, 1543
 - ~InvalidStateException, 1545
 - InvalidStateException, 1543, 1544
 - clone, 1545
- decaf::lang::exceptions::NullPointerException-Exception, 1989
 - ~NullPointerException, 1991
 - NullPointerException, 1990, 1991
 - clone, 1991
- decaf::lang::exceptions::NumberFormatException-Exception, 1994
 - ~NumberFormatException, 1997
 - NumberFormatException, 1995, 1996
 - clone, 1997
- decaf::lang::exceptions::RuntimeException, 2315
 - ~RuntimeException, 2317
 - RuntimeException, 2315, 2316
 - clone, 2317
- decaf::lang::exceptions::UnsupportedOperationException, 2824
 - ~UnsupportedOperationException, 2826
 - UnsupportedOperationException, 2825, 2826
 - clone, 2827
- decaf::net, 92
- decaf::net::BindException, 532
 - ~BindException, 534
 - BindException, 533, 534
 - clone, 534
- decaf::net::ConnectException, 931
 - ~ConnectException, 933
 - ConnectException, 932, 933
 - clone, 933
- decaf::net::DatagramPacket, 1078
 - ~DatagramPacket, 1082
 - DatagramPacket, 1080–1082

- getAddress, 1082
- getData, 1082
- getLength, 1082
- getOffset, 1083
- getPort, 1083
- getSize, 1083
- getSocketAddress, 1083
- setAddress, 1083
- setData, 1083, 1084
- setLength, 1084
- setOffset, 1085
- setPort, 1085
- setSocketAddress, 1085
- decaf::net::HttpRetryException, 1408
 - ~HttpRetryException, 1410
 - HttpRetryException, 1409, 1410
 - clone, 1411
- decaf::net::Inet4Address, 1431
 - ~Inet4Address, 1432
 - Inet4Address, 1432
 - InetAddress, 1435
 - clone, 1433
 - isAnyLocalAddress, 1433
 - isLinkLocalAddress, 1433
 - isLoopbackAddress, 1433
 - isMCGlobal, 1433
 - isMCLinkLocal, 1434
 - isMCNodeLocal, 1434
 - isMCOrgLocal, 1434
 - isMCSiteLocal, 1434
 - isMulticastAddress, 1435
 - isSiteLocalAddress, 1435
- decaf::net::Inet6Address, 1435
 - ~Inet6Address, 1436
 - Inet6Address, 1436
 - InetAddress, 1437
 - clone, 1436
- decaf::net::InetAddress, 1437
 - ~InetAddress, 1439
 - InetAddress, 1439
 - addressBytes, 1444
 - anyBytes, 1444
 - bytesToInt, 1439
 - clone, 1439
 - getAddress, 1440
 - getAnyAddress, 1440
 - getByAddress, 1440
 - getHostAddress, 1441
 - getHostName, 1441
 - getLocalHost, 1441
 - getLoopbackAddress, 1442
 - hostname, 1444
 - isAnyLocalAddress, 1442
 - isLinkLocalAddress, 1442
 - isLoopbackAddress, 1442
 - isMCGlobal, 1442
 - isMCLinkLocal, 1443
 - isMCNodeLocal, 1443
 - isMCOrgLocal, 1443
 - isMCSiteLocal, 1443
 - isMulticastAddress, 1443
 - isSiteLocalAddress, 1444
 - loopbackBytes, 1444
 - reached, 1444
 - toString, 1444
- decaf::net::InetSocketAddress, 1445
 - ~InetSocketAddress, 1445
 - InetSocketAddress, 1445
- decaf::net::MalformedURLException, 1766
 - ~MalformedURLException, 1768
 - MalformedURLException, 1766, 1767
 - clone, 1768
- decaf::net::NoRouteToHostException, 1978
 - ~NoRouteToHostException, 1980
 - NoRouteToHostException, 1979, 1980
 - clone, 1981
- decaf::net::PortUnreachableException, 2107
 - ~PortUnreachableException, 2109
 - PortUnreachableException, 2108, 2109
 - clone, 2109
- decaf::net::ProtocolException, 2211
 - ~ProtocolException, 2213
 - ProtocolException, 2212, 2213
 - clone, 2213
- decaf::net::ServerSocket, 2342
 - ~ServerSocket, 2346
 - ServerSocket, 2344–2346
 - accept, 2346
 - bind, 2347
 - checkClosed, 2348
 - close, 2348
 - ensureCreated, 2348
 - getDefaultBacklog, 2348
 - getLocalPort, 2348

- getReceiveBufferSize, 2348
- getReuseAddress, 2349
- getSoTimeout, 2349
- implAccept, 2349
- isBound, 2350
- isClosed, 2350
- setReceiveBufferSize, 2350
- setReuseAddress, 2350
- setSoTimeout, 2351
- setSocketImplFactory, 2351
- setupSocketImpl, 2351
- toString, 2351
- decaf::net::ServerSocketFactory, 2352
 - ~ServerSocketFactory, 2353
 - ServerSocketFactory, 2353
 - createServerSocket, 2353, 2354
 - getDefault, 2355
- decaf::net::Socket, 2452
 - ~Socket, 2457
 - ServerSocket, 2469
 - Socket, 2455–2457
 - accepted, 2457
 - bind, 2458
 - checkClosed, 2458
 - close, 2458
 - connect, 2458, 2459
 - ensureCreated, 2459
 - getInetAddress, 2459
 - getInputStream, 2459
 - getKeepAlive, 2460
 - getLocalAddress, 2460
 - getLocalPort, 2460
 - getOOBInline, 2461
 - getOutputStream, 2461
 - getPort, 2461
 - getReceiveBufferSize, 2461
 - getReuseAddress, 2462
 - getSendBufferSize, 2462
 - getSoLinger, 2462
 - getSoTimeout, 2463
 - getTcpNoDelay, 2463
 - getTrafficClass, 2463
 - impl, 2469
 - initSocketImpl, 2464
 - isBound, 2464
 - isClosed, 2464
 - isConnected, 2464
 - isInputShutdown, 2464
 - isOutputShutdown, 2464
 - sendUrgentData, 2465
 - setKeepAlive, 2465
 - setOOBInline, 2465
 - setReceiveBufferSize, 2466
 - setReuseAddress, 2466
 - setSendBufferSize, 2466
 - setSoLinger, 2467
 - setSoTimeout, 2467
 - setSocketImplFactory, 2467
 - setTcpNoDelay, 2468
 - setTrafficClass, 2468
 - shutdownInput, 2469
 - shutdownOutput, 2469
 - toString, 2469
- decaf::net::SocketAddress, 2470
 - ~SocketAddress, 2470
- decaf::net::SocketError, 2470
 - getErrorCode, 2471
 - getErrorString, 2471
- decaf::net::SocketException, 2471
 - ~SocketException, 2473
 - SocketException, 2472
 - clone, 2473
- decaf::net::SocketFactory, 2473
 - ~SocketFactory, 2475
 - SocketFactory, 2475
 - createSocket, 2475–2477
 - getDefault, 2477
- decaf::net::SocketImpl, 2479
 - ~SocketImpl, 2481
 - SocketImpl, 2481
 - accept, 2481
 - address, 2487
 - available, 2481
 - bind, 2481
 - close, 2482
 - connect, 2482
 - create, 2482
 - fd, 2487
 - getFileDescriptor, 2483
 - getInetAddress, 2483
 - getInputStream, 2483
 - getLocalAddress, 2483
 - getLocalPort, 2484
 - getOption, 2484
 - getOutputStream, 2484
 - getPort, 2485
 - listen, 2485
 - localPort, 2487
 - port, 2487
 - sendUrgentData, 2485

- setOption, 2486
- shutdownInput, 2486
- shutdownOutput, 2486
- supportsUrgentData, 2487
- toString, 2487
- decaf::net::SocketImplFactory, 2487
 - ~SocketImplFactory, 2488
 - createSocketImpl, 2488
- decaf::net::SocketOptions, 2488
 - ~SocketOptions, 2490
 - SOCKET_OPTION_BINDADDR, 2490
 - SOCKET_OPTION_BROADCAST, 2490
 - SOCKET_OPTION_IP_MULTICAST_IF, 2490
 - SOCKET_OPTION_IP_MULTICAST_IF2, 2490
 - SOCKET_OPTION_IP_MULTICAST_LOOP, 2491
 - SOCKET_OPTION_IP_TOS, 2491
 - SOCKET_OPTION_KEEPALIVE, 2491
 - SOCKET_OPTION_LINGER, 2491
 - SOCKET_OPTION_OOBINLINE, 2491
 - SOCKET_OPTION_RCVBUF, 2492
 - SOCKET_OPTION_REUSEADDR, 2492
 - SOCKET_OPTION_SNDBUF, 2492
 - SOCKET_OPTION_TCP_NODELAY, 2492
 - SOCKET_OPTION_TIMEOUT, 2492
- decaf::net::SocketTimeoutException, 2493
 - ~SocketTimeoutException, 2495
 - SocketTimeoutException, 2493, 2494
 - clone, 2495
- decaf::net::URI, 2828
 - ~URI, 2833
 - URI, 2830–2832
 - compareTo, 2833
 - create, 2833
 - equals, 2834
 - getAuthority, 2834
 - getFragment, 2834
 - getHost, 2834
 - getPath, 2834
 - getPort, 2834
 - getQuery, 2834
 - getRawAuthority, 2834
 - getRawFragment, 2835
 - getRawPath, 2835
 - getRawQuery, 2835
 - getRawSchemeSpecificPart, 2835
 - getRawUserInfo, 2836
 - getScheme, 2836
 - getSchemeSpecificPart, 2836
 - getUserInfo, 2836
 - isAbsolute, 2836
 - isOpaque, 2837
 - normalize, 2837
 - operator<, 2837
 - operator==, 2838
 - parseServerAuthority, 2838
 - relativize, 2839
 - resolve, 2839
 - toString, 2840
 - toURL, 2840
- decaf::net::URISyntaxException, 2856
 - ~URISyntaxException, 2859
 - URISyntaxException, 2857–2859
 - clone, 2859
 - getIndex, 2860
 - getInput, 2860
 - getReason, 2860
- decaf::net::URL, 2868
 - ~URL, 2870
 - URL, 2870
- decaf::net::URLDecoder, 2870
 - ~URLDecoder, 2870
 - decode, 2870
- decaf::net::URLEncoder, 2871
 - ~URLEncoder, 2871
 - encode, 2871
- decaf::net::UnknownHostException, 2816
 - ~UnknownHostException, 2818
 - UnknownHostException, 2817, 2818
 - clone, 2818
- decaf::net::UnknownServiceException, 2819
 - ~UnknownServiceException, 2821
 - UnknownServiceException, 2819, 2820
 - clone, 2821
- decaf::net::ssl, 94
- decaf::net::ssl::SSLContext, 2495
 - ~SSLContext, 2496
 - SSLContext, 2496

- getDefault, 2496
- getDefaultSSLParameters, 2497
- getServerSocketFactory, 2497
- getSocketFactory, 2497
- getSupportedSSLParameters, 2498
- setDefault, 2498
- decaf::net::ssl::SSLContextSpi, 2498
 - ~SSLContextSpi, 2499
 - providerGetDefaultSSLParameters, 2499
 - providerGetServerSocketFactory, 2499
 - providerGetSocketFactory, 2500
 - providerGetSupportedSSLParameters, 2500
 - providerInit, 2501
- decaf::net::ssl::SSLParameters, 2501
 - ~SSLParameters, 2503
 - SSLParameters, 2502
 - getCipherSuites, 2503
 - getNeedClientAuth, 2503
 - getProtocols, 2503
 - getWantClientAuth, 2503
 - setCipherSuites, 2503
 - setNeedClientAuth, 2504
 - setProtocols, 2504
 - setWantClientAuth, 2504
- decaf::net::ssl::SSLServerSocket, 2504
 - ~SSLServerSocket, 2507
 - SSLServerSocket, 2506, 2507
 - getEnabledCipherSuites, 2508
 - getEnabledProtocols, 2508
 - getNeedClientAuth, 2508
 - getSupportedCipherSuites, 2508
 - getSupportedProtocols, 2508
 - getWantClientAuth, 2509
 - setEnabledCipherSuites, 2509
 - setEnabledProtocols, 2509
 - setNeedClientAuth, 2510
 - setWantClientAuth, 2510
- decaf::net::ssl::SSLServerSocketFactory, 2510
 - ~SSLServerSocketFactory, 2511
 - SSLServerSocketFactory, 2511
 - getDefault, 2511
 - getDefaultCipherSuites, 2512
 - getSupportedCipherSuites, 2512
- decaf::net::ssl::SSLSocket, 2513
 - ~SSLSocket, 2517
 - SSLSocket, 2515, 2516
 - getEnabledCipherSuites, 2517
 - getEnabledProtocols, 2517
 - getNeedClientAuth, 2517
 - getSSLParameters, 2517
 - getSupportedCipherSuites, 2518
 - getSupportedProtocols, 2518
 - getUseClientMode, 2518
 - getWantClientAuth, 2518
 - setEnabledCipherSuites, 2519
 - setEnabledProtocols, 2519
 - setNeedClientAuth, 2520
 - setSSLParameters, 2520
 - setUseClientMode, 2520
 - setWantClientAuth, 2521
 - startHandshake, 2521
- decaf::net::ssl::SSLSocketFactory, 2522
 - ~SSLSocketFactory, 2523
 - SSLSocketFactory, 2523
 - createSocket, 2523
 - getDefault, 2523
 - getDefaultCipherSuites, 2524
 - getSupportedCipherSuites, 2524
- decaf::nio, 94
- decaf::nio::Buffer, 582
 - ~Buffer, 585
 - Buffer, 585
 - _capacity, 589
 - _limit, 589
 - _mark, 589
 - _markSet, 589
 - _position, 589
 - capacity, 585
 - clear, 585
 - flip, 585
 - hasRemaining, 586
 - isReadOnly, 586
 - limit, 586
 - mark, 587
 - position, 587
 - remaining, 588
 - reset, 588
 - rewind, 588
- decaf::nio::BufferOverflowException, 609
 - ~BufferOverflowException, 611
 - BufferOverflowException, 610, 611
 - clone, 611
- decaf::nio::BufferUnderflowException, 611
 - ~BufferUnderflowException, 613
 - BufferUnderflowException, 612, 613
 - clone, 613

- decaf::nio::ByteBuffer, 690
 - ~ByteBuffer, 695
 - ByteBuffer, 694
 - allocate, 695
 - array, 695
 - arrayOffset, 696
 - asCharBuffer, 696
 - asDoubleBuffer, 696
 - asFloatBuffer, 697
 - asIntBuffer, 697
 - asLongBuffer, 698
 - asReadOnlyBuffer, 698
 - asShortBuffer, 698
 - compact, 699
 - compareTo, 699
 - duplicate, 699
 - equals, 700
 - get, 700, 701
 - getChar, 702
 - getDouble, 703
 - getFloat, 703, 704
 - getInt, 704, 705
 - getLong, 705
 - getShort, 706
 - hasArray, 707
 - isReadOnly, 707
 - operator<, 707
 - operator==, 707
 - put, 707–709
 - putChar, 710
 - putDouble, 711
 - putFloat, 712
 - putInt, 713
 - putLong, 714
 - putShort, 715
 - slice, 716
 - toString, 716
 - wrap, 716, 717
- decaf::nio::CharBuffer, 785
 - ~CharBuffer, 788
 - CharBuffer, 788
 - allocate, 788
 - append, 789, 790
 - array, 790
 - arrayOffset, 791
 - asReadOnlyBuffer, 791
 - charAt, 792
 - compact, 792
 - compareTo, 793
 - duplicate, 793
 - equals, 793
 - get, 793, 794
 - hasArray, 795
 - length, 795
 - operator<, 796
 - operator==, 796
 - put, 796–799
 - read, 800
 - slice, 801
 - subSequence, 801
 - toString, 802
 - wrap, 802
- decaf::nio::DoubleBuffer, 1259
 - ~DoubleBuffer, 1261
 - DoubleBuffer, 1261
 - allocate, 1262
 - array, 1262
 - arrayOffset, 1262
 - asReadOnlyBuffer, 1263
 - compact, 1263
 - compareTo, 1264
 - duplicate, 1264
 - equals, 1264
 - get, 1264–1266
 - hasArray, 1266
 - operator<, 1267
 - operator==, 1267
 - put, 1267–1269
 - slice, 1270
 - toString, 1270
 - wrap, 1270, 1271
- decaf::nio::FloatBuffer, 1368
 - ~FloatBuffer, 1370
 - FloatBuffer, 1370
 - allocate, 1370
 - array, 1370
 - arrayOffset, 1371
 - asReadOnlyBuffer, 1371
 - compact, 1372
 - compareTo, 1372
 - duplicate, 1372
 - equals, 1373
 - get, 1373, 1374
 - hasArray, 1375
 - operator<, 1375
 - operator==, 1375
 - put, 1375–1377
 - slice, 1378
 - toString, 1378
 - wrap, 1378, 1379

- decaf::nio::IntBuffer, 1488
 - ~IntBuffer, 1490
 - IntBuffer, 1490
 - allocate, 1490
 - array, 1490
 - arrayOffset, 1491
 - asReadOnlyBuffer, 1491
 - compact, 1492
 - compareTo, 1492
 - duplicate, 1492
 - equals, 1493
 - get, 1493, 1494
 - hasArray, 1495
 - operator<, 1495
 - operator==, 1495
 - put, 1495–1497
 - slice, 1498
 - toString, 1498
 - wrap, 1498, 1499
- decaf::nio::InvalidMarkException, 1539
 - ~InvalidMarkException, 1541
 - InvalidMarkException, 1539, 1540
 - clone, 1541
- decaf::nio::LongBuffer, 1752
 - ~LongBuffer, 1755
 - LongBuffer, 1755
 - allocate, 1755
 - array, 1755
 - arrayOffset, 1756
 - asReadOnlyBuffer, 1756
 - compact, 1757
 - compareTo, 1757
 - duplicate, 1757
 - equals, 1758
 - get, 1758, 1759
 - hasArray, 1760
 - operator<, 1760
 - operator==, 1760
 - put, 1760–1763
 - slice, 1763
 - toString, 1763
 - wrap, 1764
- decaf::nio::ReadOnlyBufferException, 2244
 - ~ReadOnlyBufferException, 2246
 - ReadOnlyBufferException, 2244, 2245
 - clone, 2246
- decaf::nio::ShortBuffer, 2419
 - ~ShortBuffer, 2421
 - ShortBuffer, 2421
 - allocate, 2421
 - array, 2422
 - arrayOffset, 2422
 - asReadOnlyBuffer, 2423
 - compact, 2423
 - compareTo, 2424
 - duplicate, 2424
 - equals, 2424
 - get, 2424, 2425
 - hasArray, 2426
 - operator<, 2426
 - operator==, 2426
 - put, 2427–2429
 - slice, 2429
 - toString, 2430
 - wrap, 2430, 2431
- decaf::security, 95
- decaf::security::GeneralSecurityException, 1395
 - ~GeneralSecurityException, 1397
 - GeneralSecurityException, 1395, 1396
 - clone, 1397
- decaf::security::InvalidKeyException, 1536
 - ~InvalidKeyException, 1538
 - InvalidKeyException, 1537, 1538
 - clone, 1538
- decaf::security::Key, 1598
 - ~Key, 1599
 - getAlgorithm, 1599
 - getEncoded, 1600
 - getFormat, 1600
- decaf::security::KeyException, 1600
 - ~KeyException, 1602
 - KeyException, 1601, 1602
 - clone, 1602
- decaf::security::KeyManagementException, 1603
 - ~KeyManagementException, 1605
 - KeyManagementException, 1604, 1605
 - clone, 1605
- decaf::security::NoSuchAlgorithmException, 1981
 - ~NoSuchAlgorithmException, 1983
 - NoSuchAlgorithmException, 1982, 1983
 - clone, 1983

- decaf::security::NoSuchProviderException, 1986
 - ~NoSuchProviderException, 1988
 - NoSuchProviderException, 1987, 1988
 - clone, 1989
- decaf::security::Principal, 2159
 - ~Principal, 2160
 - equals, 2160
 - getName, 2160
- decaf::security::PublicKey, 2213
 - ~PublicKey, 2214
- decaf::security::SecureRandom, 2320
 - ~SecureRandom, 2323
 - SecureRandom, 2322
 - next, 2323
 - nextBytes, 2323, 2324
 - setSeed, 2324, 2325
- decaf::security::SecureRandomSpi, 2329
 - ~SecureRandomSpi, 2330
 - SecureRandomSpi, 2330
 - providerGenerateSeed, 2330
 - providerNextBytes, 2330
 - providerSetSeed, 2330
- decaf::security::SignatureException, 2438
 - ~SignatureException, 2440
 - SignatureException, 2439, 2440
 - clone, 2440
- decaf::security::auth, 95
- decaf::security::auth::x500, 95
- decaf::security::auth::x500::X500Principal, 2914
 - ~X500Principal, 2914
 - getEncoded, 2914
 - getName, 2914
 - hashCode, 2914
- decaf::security::cert, 96
- decaf::security::cert::Certificate, 751
 - ~Certificate, 752
 - equals, 752
 - getEncoded, 752
 - getPublicKey, 752
 - getType, 753
 - toString, 753
 - verify, 753, 754
- decaf::security::cert::CertificateEncodingException, 755
 - ~CertificateEncodingException, 756
 - CertificateEncodingException, 755, 756
- clone, 756
- decaf::security::cert::CertificateException, 757
 - ~CertificateException, 758
 - CertificateException, 757, 758
 - clone, 758
- decaf::security::cert::CertificateExpiredException, 759
 - ~CertificateExpiredException, 760
 - CertificateExpiredException, 759, 760
 - clone, 760
- decaf::security::cert::CertificateNotYetValidException, 761
 - ~CertificateNotYetValidException, 762
 - CertificateNotYetValidException, 761, 762
 - clone, 762
- decaf::security::cert::CertificateParsingException, 763
 - ~CertificateParsingException, 764
 - CertificateParsingException, 763, 764
 - clone, 764
- decaf::security::cert::X509Certificate, 2915
 - ~X509Certificate, 2916
 - checkValidity, 2916
 - getBasicConstraints, 2916
 - getIssuerUniqueID, 2916
 - getIssuerX500Principal, 2916
 - getKeyUsage, 2916
 - getNotAfter, 2916
 - getNotBefore, 2916
 - getSigAlgName, 2916
 - getSigAlgOID, 2916
 - getSigAlgParams, 2916
 - getSignature, 2916
 - getSubjectUniqueID, 2916
 - getSubjectX500Principal, 2916
 - getTBSCertificate, 2917
 - getVersion, 2917
- decaf::util, 96
- decaf::util::AbstractCollection
 - ~AbstractCollection, 110
 - AbstractCollection, 110
 - add, 110
 - addAll, 110
 - clear, 111

- contains, 112
- containsAll, 113
- copy, 113
- equals, 114
- isEmpty, 114
- lock, 115
- mutex, 121
- notify, 115
- notifyAll, 116
- operator=, 116
- remove, 116
- removeAll, 117
- retainAll, 118
- toArray, 119
- tryLock, 119
- unlock, 120
- wait, 120, 121
- decaf::util::AbstractCollection< E >, 106
- decaf::util::AbstractList
 - ~AbstractList, 124
 - AbstractList, 124
 - add, 125, 126
 - addAll, 126
 - clear, 127
 - indexOf, 128
 - iterator, 128, 129
 - lastIndexOf, 129
 - listIterator, 130–132
 - modCount, 134
 - removeAt, 132
 - removeRange, 133
 - set, 133
- decaf::util::AbstractList< E >, 123
- decaf::util::AbstractMap
 - ~AbstractMap, 135
- decaf::util::AbstractMap< K, V, COMPARE-
ATOR >, 134
- decaf::util::AbstractQueue
 - ~AbstractQueue, 139
 - AbstractQueue, 139
 - add, 139
 - addAll, 140
 - clear, 141
 - element, 141
 - remove, 142
- decaf::util::AbstractQueue< E >, 136
- decaf::util::AbstractSequentialList
 - ~AbstractSequentialList, 146
 - add, 146
 - addAll, 146
 - get, 148
 - iterator, 148, 149
 - listIterator, 149, 150
 - removeAt, 150
 - set, 151
- decaf::util::AbstractSequentialList< E >, 143
- decaf::util::AbstractSet
 - ~AbstractSet, 153
 - removeAll, 153
- decaf::util::AbstractSet< E >, 152
- decaf::util::ArrayList
 - ~ArrayList, 448
 - ArrayList, 448
 - add, 448, 449
 - addAll, 450, 451
 - clear, 451
 - contains, 452
 - ensureCapacity, 453
 - get, 453
 - indexOf, 453
 - isEmpty, 454
 - lastIndexOf, 454
 - operator=, 455
 - remove, 455
 - removeAt, 456
 - set, 456
 - size, 457
 - toArray, 457
 - toString, 458
 - trimToSize, 458
- decaf::util::ArrayList< E >, 445
- decaf::util::Arrays, 471
 - ~Arrays, 472
 - fill, 472
- decaf::util::Collection
 - ~Collection, 853
 - add, 853
 - addAll, 854
 - clear, 856
 - contains, 857
 - containsAll, 858
 - copy, 858
 - equals, 859
 - isEmpty, 860
 - remove, 861
 - removeAll, 862
 - retainAll, 863
 - size, 864
 - toArray, 865

- decaf::util::Collection< E >, 851
- decaf::util::Comparator
 - ~Comparator, 889
 - compare, 889
 - operator(), 890
- decaf::util::Comparator< T >, 888
- decaf::util::ConcurrentModificationException
 - Exception, 902
 - ~ConcurrentModificationException, 904
 - ConcurrentModificationException, 903, 904
 - clone, 904
- decaf::util::Date, 1139
 - ~Date, 1140
 - Date, 1140
 - after, 1140
 - before, 1141
 - compareTo, 1141
 - equals, 1141
 - getTime, 1141
 - operator<, 1141
 - operator=, 1141
 - operator==, 1141
 - setTime, 1142
 - toString, 1142
- decaf::util::Deque
 - ~Deque, 1198
 - addFirst, 1198
 - addLast, 1199
 - descendingIterator, 1199, 1200
 - getFirst, 1200, 1201
 - getLast, 1201, 1202
 - offerFirst, 1202
 - offerLast, 1203
 - peekFirst, 1203
 - peekLast, 1204
 - pollFirst, 1204
 - pollLast, 1205
 - pop, 1206
 - push, 1206
 - removeFirst, 1207
 - removeFirstOccurrence, 1208
 - removeLast, 1208
 - removeLastOccurrence, 1209
- decaf::util::Deque< E >, 1196
- decaf::util::Iterator
 - ~Iterator, 1559
 - hasNext, 1559
 - next, 1560
 - remove, 1560
- decaf::util::Iterator< E >, 1559
- decaf::util::LinkedList
 - ~LinkedList, 1639
 - LinkedList, 1639
 - add, 1639, 1640
 - addAll, 1641, 1642
 - addFirst, 1643
 - addLast, 1643
 - clear, 1643
 - contains, 1644
 - copy, 1645
 - descendingIterator, 1645
 - element, 1645
 - get, 1646
 - getFirst, 1646, 1647
 - getLast, 1647
 - indexOf, 1647
 - isEmpty, 1648
 - lastIndexOf, 1648
 - listIterator, 1649
 - offer, 1649
 - offerFirst, 1650
 - offerLast, 1650
 - operator=, 1651
 - peek, 1651
 - peekFirst, 1651
 - peekLast, 1652
 - poll, 1652
 - pollFirst, 1652
 - pollLast, 1653
 - pop, 1653
 - push, 1653
 - remove, 1654, 1655
 - removeFirst, 1655
 - removeFirstOccurrence, 1656
 - removeLast, 1656
 - removeLastOccurrence, 1657
 - set, 1657
 - size, 1658
 - toArray, 1658
- decaf::util::LinkedList< E >, 1633
- decaf::util::List
 - ~List, 1660
 - List, 1660
 - add, 1660
 - addAll, 1661
 - get, 1662
 - indexOf, 1663
 - lastIndexOf, 1664

- listIterator, 1665–1667
- removeAt, 1668
- set, 1669
- decaf::util::List< E >, 1658
- decaf::util::ListIterator
 - ~ListIterator, 1672
 - add, 1672
 - hasPrevious, 1672
 - nextIndex, 1672
 - previous, 1673
 - previousIndex, 1673
 - set, 1674
- decaf::util::ListIterator< E >, 1671
- decaf::util::Map
 - ~Map, 1770
 - Map, 1770
 - clear, 1770
 - containsKey, 1770
 - containsValue, 1771
 - copy, 1772
 - equals, 1772
 - get, 1773
 - isEmpty, 1774
 - keySet, 1775
 - put, 1776
 - putAll, 1777
 - remove, 1777
 - size, 1778
 - values, 1779
- decaf::util::Map< K, V, COMPARATOR >, 1768
- decaf::util::Map< K, V, COMPARATOR >::Entry, 1274
- decaf::util::Map::Entry
 - ~Entry, 1274
 - Entry, 1274
 - getKey, 1274
 - getValue, 1274
 - setValue, 1274
- decaf::util::NoSuchElementException, 1984
 - ~NoSuchElementException, 1986
 - NoSuchElementException, 1984, 1985
 - clone, 1986
- decaf::util::PriorityQueue
 - ~PriorityQueue, 2165
 - PriorityQueue, 2164, 2165
 - PriorityQueueIterator, 2171
 - add, 2165
 - clear, 2166
 - comparator, 2167
 - iterator, 2167
 - offer, 2167
 - operator=, 2168
 - peek, 2168
 - poll, 2169
 - remove, 2169, 2170
 - size, 2170
- decaf::util::PriorityQueue< E >, 2160
- decaf::util::Properties, 2200
 - ~Properties, 2202
 - Properties, 2201
 - clear, 2202
 - clone, 2202
 - copy, 2202
 - defaults, 2209
 - equals, 2202
 - getProperty, 2203
 - hasProperty, 2203
 - isEmpty, 2203
 - load, 2204
 - operator=, 2206
 - propertyNames, 2206
 - remove, 2207
 - setProperty, 2207
 - size, 2207
 - store, 2207, 2208
 - toArray, 2209
 - toString, 2209
- decaf::util::Queue
 - ~Queue, 2223
 - element, 2223
 - offer, 2224
 - peek, 2225
 - poll, 2225
 - remove, 2226
- decaf::util::Queue< E >, 2222
- decaf::util::Random, 2229
 - ~Random, 2231
 - Random, 2230, 2231
 - next, 2231
 - nextBoolean, 2232
 - nextBytes, 2232
 - nextDouble, 2233
 - nextFloat, 2233
 - nextGaussian, 2233
 - nextInt, 2233, 2234
 - nextLong, 2234
 - setSeed, 2234

- decaf::util::Set
 - ~Set, 2398
- decaf::util::Set< E >, 2397
- decaf::util::StlList
 - StlList, 2541
 - add, 2542
 - addAll, 2543, 2544
 - clear, 2545
 - contains, 2545
 - copy, 2546
 - equals, 2546
 - get, 2547
 - indexOf, 2547
 - isEmpty, 2548
 - iterator, 2548
 - lastIndexOf, 2548
 - listIterator, 2549
 - remove, 2550
 - removeAt, 2550
 - set, 2551
 - size, 2551
- decaf::util::StlList< E >, 2536
- decaf::util::StlMap
 - ~StlMap, 2556
 - StlMap, 2556
 - clear, 2556
 - containsKey, 2557
 - containsValue, 2557
 - copy, 2558
 - equals, 2558
 - get, 2558, 2559
 - isEmpty, 2559
 - keySet, 2560
 - lock, 2560
 - notify, 2560
 - notifyAll, 2561
 - put, 2561
 - putAll, 2561, 2562
 - remove, 2562
 - size, 2563
 - tryLock, 2563
 - unlock, 2563
 - values, 2563
 - wait, 2564
- decaf::util::StlMap< K, V, COMPARATOR >, 2552
- decaf::util::StlQueue
 - ~StlQueue, 2567
 - StlQueue, 2567
 - back, 2567, 2568
 - clear, 2568
 - empty, 2568
 - enqueueFront, 2568
 - front, 2568, 2569
 - getSafeValue, 2569
 - iterator, 2569
 - lock, 2569
 - notify, 2570
 - notifyAll, 2570
 - pop, 2570
 - push, 2570
 - reverse, 2571
 - size, 2571
 - toArray, 2571
 - tryLock, 2571
 - unlock, 2572
 - wait, 2572, 2573
- decaf::util::StlQueue< T >, 2565
- decaf::util::StlSet
 - ~StlSet, 2577
 - StlSet, 2576, 2577
 - add, 2577
 - clear, 2578
 - contains, 2578
 - copy, 2579
 - equals, 2579
 - isEmpty, 2580
 - iterator, 2580
 - remove, 2580
 - size, 2581
- decaf::util::StlSet< E >, 2574
- decaf::util::StringTokenizer, 2627
 - ~StringTokenizer, 2628
 - StringTokenizer, 2628
 - countTokens, 2628
 - hasMoreTokens, 2629
 - nextToken, 2629
 - reset, 2630
 - toArray, 2630
- decaf::util::Timer, 2739
 - ~Timer, 2741
 - Timer, 2741
 - cancel, 2741
 - purge, 2741
 - schedule, 2742–2746
 - scheduleAtFixedRate, 2747–2750
- decaf::util::TimerTask, 2751
 - ~TimerTask, 2751
 - Timer, 2753
 - TimerImpl, 2753

- TimerTask, 2751
- cancel, 2752
- decaf::internal::util::TimerTaskHeap, 2753
- getWhen, 2752
- isScheduled, 2752
- scheduledExecutionTime, 2752
- setScheduledTime, 2753
- decaf::util::UUID, 2877
 - ~UUID, 2879
 - UUID, 2879
 - clockSequence, 2879
 - compareTo, 2879
 - equals, 2880
 - fromString, 2880
 - getLeastSignificantBits, 2880
 - getMostSignificantBits, 2881
 - nameUUIDFromBytes, 2881
 - node, 2881
 - operator<, 2882
 - operator==, 2882
 - randomUUID, 2882
 - timestamp, 2883
 - toString, 2883
 - variant, 2883
 - version, 2884
- decaf::util::comparators, 98
- decaf::util::comparators::Less
 - ~Less, 1613
 - Less, 1613
 - compare, 1613
 - operator(), 1614
- decaf::util::comparators::Less< E >, 1612
- decaf::util::concurrent, 98
- decaf::util::concurrent::AbstractExecutorService, 122
 - ~AbstractExecutorService, 122
 - AbstractExecutorService, 122
- decaf::util::concurrent::BlockingQueue
 - ~BlockingQueue, 541
 - drainTo, 541, 542
 - offer, 543
 - poll, 543
 - put, 544
 - remainingCapacity, 544
 - take, 544
- decaf::util::concurrent::BlockingQueue< E >, 538
- decaf::util::concurrent::BrokenBarrierException, 556
 - ~BrokenBarrierException, 558
- BrokenBarrierException, 556, 557
- clone, 558
- decaf::util::concurrent::Callable
 - ~Callable, 746
 - call, 746
- decaf::util::concurrent::Callable< V >, 746
- decaf::util::concurrent::CancellationException, 748
 - ~CancellationException, 750
- CancellationException, 749, 750
- clone, 750
- decaf::util::concurrent::ConcurrentMap
 - ~ConcurrentMap, 898
 - putIfAbsent, 898
 - remove, 899
 - replace, 900, 901
- decaf::util::concurrent::ConcurrentMap< K, V, COMPARATOR >, 898
- decaf::util::concurrent::ConcurrentStlMap
 - ~ConcurrentStlMap, 909
 - ConcurrentStlMap, 909
 - clear, 909
 - containsKey, 910
 - containsValue, 910
 - copy, 911
 - equals, 911
 - get, 912
 - isEmpty, 913
 - keySet, 913
 - lock, 913
 - notify, 914
 - notifyAll, 914
 - put, 914
 - putAll, 915
 - putIfAbsent, 915
 - remove, 916, 917
 - replace, 917, 918
 - size, 918
 - tryLock, 919
 - unlock, 919
 - values, 919
 - wait, 920
- decaf::util::concurrent::ConcurrentStlMap< K, V, COMPARATOR >, 905
- decaf::util::concurrent::ConditionHandle, 928
 - ~ConditionHandle, 928

- ConditionHandle, 928
- condition, 928
- criticalSection, 928
- generation, 928
- mutex, 928
- numWaiting, 928
- numWake, 929
- semaphore, 929
- decaf::util::concurrent::CopyOnWrite-
 - ArrayList
 - ~CopyOnWriteArrayList, 1032
 - CopyOnWriteArrayList, 1032
 - CopyOnWriteArraySet, 1049
 - add, 1032, 1033
 - addAll, 1034, 1035
 - addAllAbsent, 1036
 - addIfAbsent, 1036
 - clear, 1036
 - contains, 1037
 - containsAll, 1037
 - copy, 1037
 - equals, 1038
 - get, 1038
 - indexOf, 1039
 - isEmpty, 1040
 - iterator, 1040
 - lastIndexOf, 1040, 1041
 - listIterator, 1041, 1042
 - lock, 1042
 - notify, 1042
 - notifyAll, 1043
 - operator=, 1043
 - remove, 1043
 - removeAll, 1044
 - removeAt, 1044
 - retainAll, 1045
 - set, 1046
 - size, 1046
 - toArray, 1047
 - toString, 1047
 - tryLock, 1047
 - unlock, 1048
 - wait, 1048, 1049
- decaf::util::concurrent::CopyOnWrite-
 - ArrayList< E >, 1030
- decaf::util::concurrent::CopyOnWrite-
 - ArrayList< E >::ArrayList-
 - Iterator, 458
- decaf::util::concurrent::CopyOnWrite-
 - ArrayList::ArrayListIterator
 - ~ArrayListIterator, 459
 - ArrayListIterator, 459
 - add, 459
 - hasNext, 459
 - hasPrevious, 459
 - next, 460
 - nextIndex, 460
 - previous, 461
 - previousIndex, 461
 - remove, 461
 - set, 462
- decaf::util::concurrent::CopyOnWrite-
 - ArraySet
 - ~CopyOnWriteArraySet, 1054
 - CopyOnWriteArraySet, 1053
 - add, 1054
 - addAll, 1055
 - clear, 1055
 - contains, 1056
 - containsAll, 1056
 - copy, 1057
 - equals, 1057
 - isEmpty, 1058
 - iterator, 1058
 - remove, 1058
 - removeAll, 1059
 - retainAll, 1060
 - size, 1061
 - toArray, 1062
- decaf::util::concurrent::CopyOnWrite-
 - ArraySet< E >, 1050
- decaf::util::concurrent::CountDownLatch,
 - 1062
 - ~CountDownLatch, 1063
 - CountDownLatch, 1063
 - await, 1063, 1064
 - countDown, 1065
 - getCount, 1065
- decaf::util::concurrent::Delayed, 1194
 - ~Delayed, 1194
 - getDelay, 1194
- decaf::util::concurrent::ExecutionException,
 - 1294
 - ~ExecutionException, 1296
 - ExecutionException, 1295, 1296
 - clone, 1297
- decaf::util::concurrent::Executor, 1297
 - ~Executor, 1299
 - execute, 1299

- decaf::util::concurrent::ExecutorService, 1302
 - ~ExecutorService, 1303
 - awaitTermination, 1303
 - isShutdown, 1304
 - isTerminated, 1304
 - shutdown, 1304
 - shutdownNow, 1304
- decaf::util::concurrent::Executors, 1299
 - ~Executors, 1300
 - decaf::lang::Thread, 1302
 - getDefaultThreadFactory, 1300
 - newFixedThreadPool, 1300, 1301
- decaf::util::concurrent::Future
 - ~Future, 1390
 - cancel, 1390
 - get, 1391
 - isCancelled, 1392
 - isDone, 1392
- decaf::util::concurrent::Future< V >, 1390
- decaf::util::concurrent::LinkedBlockingQueue
 - ~LinkedBlockingQueue, 1624
 - LinkedBlockingQueue, 1623, 1624
 - clear, 1624
 - drainTo, 1625, 1626
 - iterator, 1626
 - offer, 1627
 - operator=, 1628
 - peek, 1628
 - poll, 1629
 - put, 1630
 - remainingCapacity, 1630
 - remove, 1631
 - size, 1632
 - take, 1632
 - toArray, 1632
 - toString, 1633
- decaf::util::concurrent::LinkedBlockingQueue< E >, 1621
- decaf::util::concurrent::Lock, 1682
 - ~Lock, 1683
 - Lock, 1683
 - isLocked, 1683
 - lock, 1683
 - unlock, 1683
- decaf::util::concurrent::Mutex, 1960
 - ~Mutex, 1961
 - Mutex, 1961
 - getName, 1962
 - lock, 1962
 - notify, 1962
 - notifyAll, 1962
 - toString, 1963
 - tryLock, 1963
 - unlock, 1963
 - wait, 1964
- decaf::util::concurrent::MutexHandle, 1965
 - ~MutexHandle, 1966
 - MutexHandle, 1966
 - lock_count, 1966
 - lock_owner, 1966
 - mutex, 1966
- decaf::util::concurrent::RejectedExecutionException, 2263
 - ~RejectedExecutionException, 2265
 - RejectedExecutionException, 2264, 2265
 - clone, 2265
- decaf::util::concurrent::RejectedExecutionHandler, 2266
 - ~RejectedExecutionHandler, 2267
 - RejectedExecutionHandler, 2267
 - rejectedExecution, 2267
- decaf::util::concurrent::Semaphore, 2331
 - ~Semaphore, 2334
 - Semaphore, 2333, 2334
 - acquire, 2334, 2335
 - acquireUninterruptibly, 2335, 2336
 - availablePermits, 2336
 - drainPermits, 2337
 - isFair, 2337
 - release, 2337
 - toString, 2338
 - tryAcquire, 2338–2340
- decaf::util::concurrent::Synchronizable, 2639
 - ~Synchronizable, 2640
 - lock, 2640
 - notify, 2641
 - notifyAll, 2642
 - tryLock, 2643
 - unlock, 2645
 - wait, 2646–2648
- decaf::util::concurrent::SynchronousQueue
 - ~SynchronousQueue, 2657
 - SynchronousQueue, 2657
 - clear, 2657

- contains, 2658
- containsAll, 2658
- drainTo, 2659
- equals, 2660
- isEmpty, 2660
- iterator, 2661
- offer, 2661
- peek, 2662
- poll, 2662, 2663
- put, 2663
- remainingCapacity, 2663
- remove, 2664
- removeAll, 2664
- retainAll, 2664
- size, 2664
- take, 2664
- toArray, 2665
- decaf::util::concurrent::Synchronous-
Queue< E >, 2655
- decaf::util::concurrent::ThreadFactory,
2714
 - ~ThreadFactory, 2714
 - newThread, 2714
- decaf::util::concurrent::ThreadPool-
Executor, 2715
 - ~ThreadPoolExecutor, 2722
 - ExecutorKernel, 2732
 - ThreadPoolExecutor, 2719–2721
 - afterExecute, 2722
 - allowCoreThreadTimeout, 2722
 - allowsCoreThreadTimeout, 2723
 - awaitTermination, 2723
 - beforeExecute, 2723
 - execute, 2724
 - getActiveCount, 2724
 - getCompletedTaskCount, 2725
 - getCorePoolSize, 2725
 - getKeepAliveTime, 2725
 - getLargestPoolSize, 2725
 - getMaximumPoolSize, 2726
 - getPoolSize, 2726
 - getQueue, 2726
 - getRejectedExecutionHandler, 2726
 - getTaskCount, 2726
 - getThreadFactory, 2727
 - isShutdown, 2727
 - isTerminated, 2727
 - isTerminating, 2727
 - prestartAllCoreThreads, 2728
 - prestartCoreThread, 2728
 - purge, 2728
 - remove, 2728
 - setCorePoolSize, 2729
 - setKeepAliveTime, 2729
 - setMaximumPoolSize, 2730
 - setRejectedExecutionHandler, 2730
 - setThreadFactory, 2730
 - shutdown, 2731
 - shutdownNow, 2731
 - terminated, 2731
- decaf::util::concurrent::ThreadPool-
Executor::AbortPolicy, 105
 - ~AbortPolicy, 106
 - AbortPolicy, 105
 - rejectedExecution, 106
- decaf::util::concurrent::ThreadPool-
Executor::CallerRunsPolicy,
747
 - ~CallerRunsPolicy, 748
 - CallerRunsPolicy, 748
 - rejectedExecution, 748
- decaf::util::concurrent::ThreadPool-
Executor::CallerRunsPolicy:-
DiscardOldestPolicy, 1224
 - ~DiscardOldestPolicy, 1224
 - DiscardOldestPolicy, 1224
 - rejectedExecution, 1225
- decaf::util::concurrent::ThreadPool-
Executor::CallerRunsPolicy:-
DiscardPolicy, 1225
 - ~DiscardPolicy, 1226
 - DiscardPolicy, 1226
 - rejectedExecution, 1226
- decaf::util::concurrent::TimeUnit, 2756
 - ~TimeUnit, 2758
 - DAYS, 2764
 - HOURS, 2764
 - MICROSECONDS, 2764
 - MILLISECONDS, 2764
 - MINUTES, 2764
 - NANOSECONDS, 2764
 - SECONDS, 2764
 - TimeUnit, 2758
 - compareTo, 2758
 - convert, 2758
 - equals, 2759
 - operator<, 2759
 - operator==, 2759
 - sleep, 2759
 - timedJoin, 2759

- timedWait, 2760
- toDays, 2760
- toHours, 2761
- toMicros, 2761
- toMillis, 2761
- toMinutes, 2762
- toNanos, 2762
- toSeconds, 2763
- toString, 2763
- valueOf, 2763
- values, 2764
- decaf::util::concurrent::TimeoutException, 2737
 - ~TimeoutException, 2739
 - TimeoutException, 2737, 2738
 - clone, 2739
- decaf::util::concurrent::atomic, 100
- decaf::util::concurrent::atomic::AtomicBoolean, 473
 - ~AtomicBoolean, 474
 - AtomicBoolean, 473
 - compareAndSet, 474
 - get, 474
 - getAndSet, 474
 - set, 475
 - toString, 475
- decaf::util::concurrent::atomic::AtomicInteger, 475
 - ~AtomicInteger, 477
 - AtomicInteger, 477
 - addAndGet, 477
 - compareAndSet, 477
 - decrementAndGet, 478
 - doubleValue, 478
 - floatValue, 478
 - get, 478
 - getAndAdd, 479
 - getAndDecrement, 479
 - getAndIncrement, 479
 - getAndSet, 479
 - incrementAndGet, 480
 - intValue, 480
 - longValue, 480
 - set, 480
 - toString, 481
- decaf::util::concurrent::atomic::AtomicReferenceCounter, 481
 - ~AtomicReferenceCounter, 482
 - AtomicReferenceCounter, 482
 - release, 483
- swap, 483
- decaf::util::concurrent::atomic::AtomicReference
 - ~AtomicReference, 485
 - AtomicReference, 485
 - compareAndSet, 485
 - get, 485
 - getAndSet, 485
 - set, 486
 - toString, 486
- decaf::util::concurrent::atomic::AtomicReference< T >, 484
- decaf::util::concurrent::locks, 100
- decaf::util::concurrent::locks::AbstractOwnableSynchronizer, 135
 - ~AbstractOwnableSynchronizer, 136
 - AbstractOwnableSynchronizer, 136
 - getExclusiveOwnerThread, 136
 - setExclusiveOwnerThread, 136
- decaf::util::concurrent::locks::Condition, 921
 - ~Condition, 923
 - await, 923, 924
 - awaitNanos, 925
 - awaitUninterruptibly, 926
 - awaitUntil, 927
 - signal, 927
 - signalAll, 927
- decaf::util::concurrent::locks::Lock, 1684
 - ~Lock, 1686
 - lock, 1686
 - lockInterruptibly, 1686
 - newCondition, 1687
 - tryLock, 1687, 1688
 - unlock, 1689
- decaf::util::concurrent::locks::LockSupport, 1690
 - ~LockSupport, 1691
- decaf::lang::Thread, 2713
- park, 1691
- parkNanos, 1692
- parkUntil, 1692
- unpark, 1692
- decaf::util::concurrent::locks::ReadWriteLock, 2246
 - ~ReadWriteLock, 2248
 - readLock, 2248
 - writeLock, 2248

- decaf::util::concurrent::locks::ReentrantLock, 2256
 - ~ReentrantLock, 2258
 - ReentrantLock, 2258
 - getHoldCount, 2258
 - isFair, 2258
 - isHeldByCurrentThread, 2258
 - isLocked, 2259
 - lock, 2259
 - lockInterruptibly, 2260
 - newCondition, 2260
 - toString, 2261
 - tryLock, 2261, 2262
 - unlock, 2263
- decaf::util::logging, 101
 - Debug, 102
 - Error, 102
 - Fatal, 102
 - Info, 102
 - Levels, 102
 - Markblock, 102
 - Null, 102
 - Off, 102
 - Throwing, 102
 - Warn, 102
- decaf::util::logging::ConsoleHandler, 990
 - ~ConsoleHandler, 991
 - ConsoleHandler, 991
 - close, 991
 - publish, 991
- decaf::util::logging::ErrorHandler, 1277
 - ~ErrorHandler, 1278
 - CLOSE_FAILURE, 1278
 - ErrorHandler, 1278
 - FLUSH_FAILURE, 1279
 - FORMAT_FAILURE, 1279
 - GENERIC_FAILURE, 1279
 - OPEN_FAILURE, 1279
 - WRITE_FAILURE, 1279
 - error, 1278
- decaf::util::logging::Filter, 1333
 - ~Filter, 1333
 - isLoggable, 1333
- decaf::util::logging::Formatter, 1387
 - ~Formatter, 1388
 - format, 1388
 - formatMessage, 1388
 - getHead, 1389
 - getTail, 1389
- decaf::util::logging::Handler, 1401
 - ~Handler, 1403
 - Handler, 1403
 - flush, 1403
 - getErrorHandler, 1403
 - getFilter, 1403
 - getFormatter, 1403
 - getLevel, 1403
 - isLoggable, 1404
 - publish, 1404
 - reportError, 1404
 - setErrorHandler, 1404
 - setFilter, 1405
 - setFormatter, 1405
 - setLevel, 1405
- decaf::util::logging::Level, 1616
 - ~Level, 1618
 - ALL, 1619
 - CONFIG, 1619
 - DEBUG, 1619
 - FINE, 1619
 - FINER, 1620
 - FINEST, 1620
 - INFO, 1620
 - INHERIT, 1620
 - Level, 1617
 - OFF, 1620
 - SEVERE, 1620
 - WARNING, 1620
 - compareTo, 1618
 - equals, 1618
 - getName, 1618
 - intValue, 1618
 - operator<, 1618
 - operator==, 1618
 - parse, 1618
 - toString, 1619
- decaf::util::logging::LogManager, 1712
 - ~LogManager, 1714
 - LogManager, 1715
 - addLogger, 1715
 - addPropertyChangeListener, 1715
 - decaf::lang::Runtime, 1718
 - getLogManager, 1716
 - getLogger, 1715
 - getLoggerNames, 1716
 - getProperties, 1716
 - getProperty, 1716
 - operator=, 1717
 - readConfiguration, 1717

- removePropertyChangeListener, 1718
- reset, 1718
- setProperties, 1718
- decaf::util::logging::LogRecord, 1719
 - ~LogRecord, 1720
 - LogRecord, 1720
 - getLevel, 1720
 - getLoggerName, 1720
 - getMessage, 1720
 - getSourceFile, 1721
 - getSourceFunction, 1721
 - getSourceLine, 1721
 - getThrown, 1721
 - getTimestamp, 1721
 - getTreadId, 1722
 - setLevel, 1722
 - setLoggerName, 1722
 - setMessage, 1722
 - setSourceFile, 1722
 - setSourceFunction, 1723
 - setSourceLine, 1723
 - setThrown, 1723
 - setTimestamp, 1723
 - setTreadId, 1724
- decaf::util::logging::LogWriter, 1724
 - ~LogWriter, 1725
 - LogWriter, 1725
 - destroy, 1725
 - getInstance, 1725
 - log, 1725
 - returnInstance, 1725
- decaf::util::logging::Logger, 1693
 - ~Logger, 1696
 - Logger, 1696
 - addHandler, 1696
 - config, 1697
 - debug, 1697
 - entering, 1697
 - exiting, 1698
 - fine, 1698
 - finer, 1698
 - finest, 1699
 - getAnonymousLogger, 1699
 - getFilter, 1699
 - getHandlers, 1700
 - getLevel, 1700
 - getLogger, 1700
 - getName, 1701
 - getParent, 1701
 - getUseParentHandlers, 1701
 - info, 1701
 - isLoggable, 1702
 - log, 1702, 1703
 - removeHandler, 1703
 - setFilter, 1703
 - setLevel, 1704
 - setParent, 1704
 - setUseParentHandlers, 1704
 - severe, 1704
 - throwing, 1705
 - warning, 1705
- decaf::util::logging::LoggerHierarchy, 1706
 - ~LoggerHierarchy, 1706
 - LoggerHierarchy, 1706
- decaf::util::logging::MarkBlockLogger, 1791
 - ~MarkBlockLogger, 1792
 - MarkBlockLogger, 1792
- decaf::util::logging::PropertiesChangeListener, 2210
 - ~PropertiesChangeListener, 2210
 - onPropertiesReset, 2210
 - onPropertyChanged, 2210
- decaf::util::logging::SimpleFormatter, 2441
 - ~SimpleFormatter, 2441
 - SimpleFormatter, 2441
 - format, 2441
- decaf::util::logging::SimpleLogger, 2442
 - ~SimpleLogger, 2443
 - SimpleLogger, 2443
 - debug, 2443
 - error, 2443
 - fatal, 2443
 - info, 2443
 - log, 2443
 - mark, 2443
 - warn, 2443
- decaf::util::logging::StreamHandler, 2603
 - ~StreamHandler, 2604
 - StreamHandler, 2604
 - close, 2604, 2605
 - flush, 2605
 - isLoggable, 2605
 - publish, 2605
 - setOutputStream, 2606
- decaf::util::logging::XMLFormatter, 2950
 - ~XMLFormatter, 2950

- XMLFormatter, 2950
- format, 2951
- getHead, 2951
- getTail, 2951
- decaf::util::zip, 102
- decaf::util::zip::Adler32, 439
 - ~Adler32, 440
 - Adler32, 440
 - getValue, 440
 - reset, 440
 - update, 440, 441
- decaf::util::zip::CRC32, 1065
 - ~CRC32, 1066
 - CRC32, 1066
 - getValue, 1066
 - reset, 1066
 - update, 1066, 1067
- decaf::util::zip::CheckedInputStream, 805
 - ~CheckedInputStream, 807
 - CheckedInputStream, 806
 - doReadArrayBounded, 807
 - doReadByte, 807
 - getChecksum, 807
 - skip, 807
- decaf::util::zip::CheckedOutputStream, 808
 - ~CheckedOutputStream, 809
 - CheckedOutputStream, 809
 - doWriteArrayBounded, 809
 - doWriteByte, 809
 - getChecksum, 810
- decaf::util::zip::Checksum, 810
 - ~Checksum, 811
 - getValue, 811
 - reset, 811
 - update, 811, 812
- decaf::util::zip::DataFormatException, 1076
 - ~DataFormatException, 1078
 - DataFormatException, 1076, 1077
 - clone, 1078
- decaf::util::zip::Deflater, 1180
 - ~Deflater, 1182
 - BEST_COMPRESSION, 1188
 - BEST_SPEED, 1188
 - DEFAULT_COMPRESSION, 1188
 - DEFAULT_STRATEGY, 1189
 - DEFLATED, 1189
 - Deflater, 1182
 - FILTERED, 1189
 - HUFFMAN_ONLY, 1189
 - NO_COMPRESSION, 1189
 - deflate, 1182, 1183
 - end, 1183
 - finish, 1184
 - finished, 1184
 - getAdler, 1184
 - getBytesRead, 1184
 - getBytesWritten, 1184
 - needsInput, 1185
 - reset, 1185
 - setDictionary, 1185, 1186
 - setInput, 1186, 1187
 - setLevel, 1188
 - setStrategy, 1188
- decaf::util::zip::DeflaterOutputStream, 1189
 - ~DeflaterOutputStream, 1192
 - DEFAULT_BUFFER_SIZE, 1193
 - DeflaterOutputStream, 1191, 1192
 - buf, 1193
 - close, 1192
 - deflate, 1193
 - deflater, 1193
 - doWriteArrayBounded, 1193
 - doWriteByte, 1193
 - finish, 1193
 - isDone, 1194
 - ownDeflater, 1194
- decaf::util::zip::Inflater, 1448
 - ~Inflater, 1450
 - Inflater, 1449
 - end, 1450
 - finish, 1450
 - finished, 1450
 - getAdler, 1450
 - getBytesRead, 1450
 - getBytesWritten, 1451
 - getRemaining, 1451
 - inflate, 1451, 1452
 - needsDictionary, 1453
 - needsInput, 1453
 - reset, 1453
 - setDictionary, 1453, 1454
 - setInput, 1455
- decaf::util::zip::InflaterInputStream, 1456
 - ~InflaterInputStream, 1460
 - DEFAULT_BUFFER_SIZE, 1464
 - InflaterInputStream, 1459
 - atEOF, 1464

- available, 1460
- buff, 1464
- close, 1461
- doReadArrayBounded, 1461
- doReadByte, 1461
- fill, 1461
- inflater, 1464
- length, 1464
- mark, 1461
- markSupported, 1462
- ownInflater, 1464
- reset, 1462
- skip, 1463
- decaf::util::zip::ZipException, 2953
 - ~ZipException, 2955
 - ZipException, 2953, 2954
 - clone, 2955
- decode
 - decaf::internal::net::URLEncoder-Decoder, 2842
 - decaf::lang::Byte, 617
 - decaf::lang::Integer, 1504
 - decaf::lang::Long, 1730
 - decaf::lang::Short, 2401
 - decaf::net::URLDecoder, 2870
- decreaseUsage
 - activemq::util::MemoryUsage, 1819
 - activemq::util::Usage, 2873
- decrementAndGet
 - decaf::util::concurrent::atomic:-AtomicInteger, 478
- defaults
 - decaf::util::Properties, 2209
- deflate
 - decaf::util::zip::Deflater, 1182, 1183
 - decaf::util::zip::DeflaterOutputStream, 1193
- deflate.h
 - BL_CODES, 3144
 - BUSY_STATE, 3144
 - COMMENT_STATE, 3144
 - Code, 3144
 - D_CODES, 3144
 - Dad, 3144
 - EXTRA_STATE, 3144
 - FINISH_STATE, 3144
 - Freq, 3144
 - GZIP, 3144
 - HCRC_STATE, 3144
 - HEAP_SIZE, 3145
 - INIT_STATE, 3145
 - IPos, 3145
 - LENGTH_CODES, 3145
 - LITERALS, 3145
 - L_CODES, 3145
 - Len, 3145
 - MAX_BITS, 3145
 - MAX_DIST, 3145
 - MIN_LOOKAHEAD, 3145
 - NAME_STATE, 3145
 - OF, 3145, 3146
 - Pos, 3145
 - Posf, 3145
 - WIN_INIT, 3145
 - _dist_code, 3146
 - _length_code, 3146
 - _tr_tally_dist, 3144
 - _tr_tally_lit, 3144
 - ct_data, 3145
 - d_code, 3144
 - deflate_state, 3145
 - max_insert_length, 3145
 - put_byte, 3145
 - static_tree_desc, 3145
 - tree_desc, 3145
- deflate_state
 - deflate.h, 3145
- deflateInit
 - zlib.h, 3156
- deflateInit2
 - zlib.h, 3156
- deflater
 - decaf::util::zip::DeflaterOutputStream, 1193
- deleteIfCancelled
 - decaf::internal::util::TimerTaskHeap, 2754
- deliverAcks
 - activemq::core::ActiveMQConsumer, 237
 - activemq::core::ActiveMQSession, 345
- deliverySequenceId
 - activemq::commands::MessageDispatchNotification, 1899
- depth
 - internal_state, 1525
- dequeue
 - activemq::core::ActiveMQConsumer, 238

- activemq::core::FifoMessageDispatch-Channel, 1324
- activemq::core::MessageDispatch-Channel, 1887
- activemq::core::SimplePriority-MessageDispatchChannel, 2446
- dequeueNoWait
 - activemq::core::FifoMessageDispatch-Channel, 1325
 - activemq::core::MessageDispatch-Channel, 1888
 - activemq::core::SimplePriority-MessageDispatchChannel, 2446
- descendingIterator
 - decaf::util::Deque, 1199, 1200
 - decaf::util::LinkedList, 1645
- descriptor
 - decaf::io::FileDescriptor, 1332
- destOptionMap
 - activemq::core::ActiveMQConstants::StaticInitializer, 2535
- destOptions
 - activemq::core::ActiveMQConstants::StaticInitializer, 2535
- destination
 - activemq::cmsutil::CmsTemplate::ProducerExecutor, 2181
 - activemq::cmsutil::CmsTemplate::ReceiveExecutor, 2250
 - activemq::commands::ConsumerControl, 996
 - activemq::commands::ConsumerInfo, 1016
 - activemq::commands::DestinationInfo, 1217
 - activemq::commands::JournalQueueAck, 1563
 - activemq::commands::JournalTopicAck, 1572
 - activemq::commands::Message, 1838
 - activemq::commands::MessageAck, 1872
 - activemq::commands::MessageDispatch, 1885
 - activemq::commands::MessageDispatchNotification, 1899
 - activemq::commands::MessagePull, 1943
- activemq::commands::ProducerInfo, 2194
- activemq::commands::SubscriptionInfo, 2634
- destroy
 - activemq::cmsutil::CmsAccessor, 821
 - activemq::cmsutil::CmsDestinationAccessor, 824
 - activemq::cmsutil::CmsTemplate, 840
 - activemq::cmsutil::DestinationResolver, 1222
 - activemq::cmsutil::DynamicDestinationResolver, 1272
 - activemq::cmsutil::ResourceLifecycleManager, 2296
 - activemq::commands::ActiveMQTempQueue, 388
 - activemq::commands::ActiveMQTempTopic, 398
 - cms::TemporaryQueue, 2699
 - cms::TemporaryTopic, 2701
 - decaf::internal::util::concurrent::ConditionImpl, 930
 - decaf::internal::util::concurrent::MutexImpl, 1967
 - decaf::util::logging::LogWriter, 1725
- destroyDestination
 - activemq::core::ActiveMQConnection, 196, 197
- destroyMarshalers
 - activemq::wireformat::openwire::OpenWireFormat, 2049
- destroyResources
 - decaf::internal::util::ResourceLifecycleManager, 2297
- digit
 - decaf::lang::Character, 767
- direct
 - gz_state, 1400
- disconnect
 - activemq::core::ActiveMQConnection, 197
- dispatch
 - activemq::core::ActiveMQConsumer, 238
 - activemq::core::ActiveMQSession, 346

- activemq::core::Dispatcher, 1235
- dispatchAsync
 - activemq::commands::Consumer-Info, 1016
 - activemq::commands::ProducerInfo, 2194
- dispose
 - activemq::core::ActiveMQConsumer, 238
 - activemq::core::ActiveMQProducer, 310
 - activemq::core::ActiveMQSession, 346
 - activemq::util::ServiceSupport, 2360
- distbits
 - inflate_state, 1446
- distcode
 - inflate_state, 1446
- dl
 - ct_data_s, 1068
- dmax
 - inflate_state, 1446
- doAppendChar
 - decaf::io::Writer, 2911
- doAppendCharSequence
 - decaf::io::Writer, 2911
- doAppendCharSequenceStartEnd
 - decaf::io::Writer, 2911
- doClose
 - activemq::core::ActiveMQConsumer, 239
 - activemq::core::ActiveMQSession, 346
- doCreateComposite
 - activemq::transport::failover::FailoverTransportFactory, 1320
 - activemq::transport::mock::MockTransportFactory, 1960
 - activemq::transport::tcp::SslTransportFactory, 2527
 - activemq::transport::tcp::TcpTransportFactory, 2698
- doInCms
 - activemq::cmsutil::CmsTemplate::ProducerExecutor, 2181
 - activemq::cmsutil::CmsTemplate::ReceiveExecutor, 2249
 - activemq::cmsutil::CmsTemplate::SendExecutor, 2342
- activemq::cmsutil::ProducerCallback, 2179
- activemq::cmsutil::SessionCallback, 2377
- doReadArray
 - decaf::io::FilterInputStream, 1337
 - decaf::io::InputStream, 1467
 - decaf::io::Reader, 2239
- doReadArrayBounded
 - activemq::io::LoggingInputStream, 1707
- decaf::internal::net::ssl::openssl::OpenSSLSocketInputStream, 2042
- decaf::internal::net::tcp::TcpSocketInputStream, 2690
- decaf::io::BlockingByteArrayInputStream, 537
- decaf::io::BufferedInputStream, 593
- decaf::io::ByteArrayInputStream, 684
- decaf::io::FilterInputStream, 1337
- decaf::io::InputStream, 1467
- decaf::io::InputStreamReader, 1476
- decaf::io::PushbackInputStream, 2217
- decaf::io::Reader, 2239
- decaf::util::zip::CheckedInputStream, 807
- decaf::util::zip::InflaterInputStream, 1461
- doReadByte
 - activemq::io::LoggingInputStream, 1707
- decaf::internal::io::StandardInputStream, 2531
- decaf::internal::net::ssl::openssl::OpenSSLSocketInputStream, 2042
- decaf::internal::net::tcp::TcpSocketInputStream, 2690
- decaf::io::BlockingByteArrayInputStream, 537
- decaf::io::BufferedInputStream, 593
- decaf::io::ByteArrayInputStream, 684
- decaf::io::FilterInputStream, 1338
- decaf::io::InputStream, 1467
- decaf::io::PushbackInputStream, 2217
- decaf::util::zip::CheckedInputStream, 807

- decaf::util::zip::InflaterInputStream, 1461
- doReadChar
 - decaf::io::Reader, 2239
- doReadCharBuffer
 - decaf::io::Reader, 2239
- doReadVector
 - decaf::io::Reader, 2239
- doStart
 - activemq::threads::Scheduler, 2318
 - activemq::util::ServiceSupport, 2360
- doStartTransaction
 - activemq::core::ActiveMQSession, 346
 - activemq::core::ActiveMQXASession, 437
- doStop
 - activemq::threads::Scheduler, 2318
 - activemq::util::ServiceSupport, 2360
- doUnmarshal
 - activemq::wireformat::openwire::OpenWireFormat, 2049
- doWriteArray
 - decaf::io::BufferedOutputStream, 596
 - decaf::io::FilterOutputStream, 1343
 - decaf::io::OutputStream, 2068
 - decaf::io::Writer, 2911
- doWriteArrayBounded
 - activemq::io::LoggingOutputStream, 1708
 - decaf::internal::io::StandardErrorOutputStream, 2530
 - decaf::internal::io::StandardOutputStream, 2533
 - decaf::internal::net::ssl::openssl::OpenSSLSocketOutputStream, 2045
 - decaf::internal::net::tcp::TcpSocketOutputStream, 2692
 - decaf::io::BufferedOutputStream, 596
 - decaf::io::ByteArrayOutputStream, 689
 - decaf::io::DataOutputStream, 1110
 - decaf::io::FilterOutputStream, 1343
 - decaf::io::OutputStream, 2068
 - decaf::io::OutputStreamWriter, 2075
 - decaf::io::Writer, 2911
- decaf::util::zip::CheckedOutputStream, 809
- decaf::util::zip::DeflaterOutputStream, 1193
- doWriteByte
 - activemq::io::LoggingOutputStream, 1708
 - decaf::internal::io::StandardErrorOutputStream, 2530
 - decaf::internal::io::StandardOutputStream, 2533
 - decaf::internal::net::ssl::openssl::OpenSSLSocketOutputStream, 2045
 - decaf::internal::net::tcp::TcpSocketOutputStream, 2692
 - decaf::io::BufferedOutputStream, 596
 - decaf::io::ByteArrayOutputStream, 689
 - decaf::io::DataOutputStream, 1110
 - decaf::io::FilterOutputStream, 1343
 - decaf::io::OutputStream, 2069
 - decaf::util::zip::CheckedOutputStream, 809
 - decaf::util::zip::DeflaterOutputStream, 1193
- doWriteChar
 - decaf::io::Writer, 2911
- doWriteString
 - decaf::io::Writer, 2912
- doWriteStringBounded
 - decaf::io::Writer, 2912
- doWriteVector
 - decaf::io::Writer, 2912
- done
 - gz_header_s, 1399
- doubleToLongBits
 - decaf::lang::Double, 1239
- doubleToRawLongBits
 - decaf::lang::Double, 1240
- doubleValue
 - activemq::util::PrimitiveValueNode::PrimitiveValue, 2143
 - decaf::lang::Byte, 618
 - decaf::lang::Character, 768
 - decaf::lang::Double, 1241
 - decaf::lang::Float, 1348
 - decaf::lang::Integer, 1504
 - decaf::lang::Long, 1731

- decaf::lang::Number, 1993
- decaf::lang::Short, 2402
- decaf::util::concurrent::atomic::-AtomicInteger, 478
- drainPermits
 - decaf::util::concurrent::Semaphore, 2337
- drainTo
 - decaf::util::concurrent::BlockingQueue, 541, 542
 - decaf::util::concurrent::LinkedBlockingQueue, 1625, 1626
 - decaf::util::concurrent::SynchronousQueue, 2659
- droppable
 - activemq::commands::Message, 1838
- dummy
 - internal_state, 1525
- duplexConnection
 - activemq::commands::BrokerInfo, 578
- duplicate
 - decaf::internal::nio::ByteBuffer, 665
 - decaf::internal::nio::CharArrayBuffer, 781
 - decaf::internal::nio::DoubleArrayBuffer, 1255
 - decaf::internal::nio::FloatArrayBuffer, 1364
 - decaf::internal::nio::IntArrayBuffer, 1484
 - decaf::internal::nio::LongArrayBuffer, 1749
 - decaf::internal::nio::ShortArrayBuffer, 2415
 - decaf::nio::ByteBuffer, 699
 - decaf::nio::CharBuffer, 793
 - decaf::nio::DoubleBuffer, 1264
 - decaf::nio::FloatBuffer, 1372
 - decaf::nio::IntBuffer, 1492
 - decaf::nio::LongBuffer, 1757
 - decaf::nio::ShortBuffer, 2424
- dyn_dtree
 - internal_state, 1525
- dyn_ltree
 - internal_state, 1525
- dyn_tree
 - tree_desc_s, 2815
- dynamicCast
 - decaf::lang::Pointer, 2087
- element
 - decaf::util::AbstractQueue, 141
 - decaf::util::LinkedList, 1645
 - decaf::util::Queue, 2223
- empty
 - decaf::util::StlQueue, 2568
- encode
 - decaf::net::URLEncoder, 2871
- encodeOthers
 - decaf::internal::net::URLEncoder-Decoder, 2842
- end
 - activemq::core::ActiveMQTransactionContext, 426
 - cms::XAResource, 2929
 - decaf::util::zip::Deflater, 1183
 - decaf::util::zip::Inflater, 1450
- enqueue
 - activemq::core::FifoMessageDispatchChannel, 1325
 - activemq::core::MessageDispatchChannel, 1888
 - activemq::core::SimplePriorityMessageDispatchChannel, 2446
- enqueueFirst
 - activemq::core::FifoMessageDispatchChannel, 1325
 - activemq::core::MessageDispatchChannel, 1888
 - activemq::core::SimplePriorityMessageDispatchChannel, 2447
- enqueueFront
 - decaf::util::StlQueue, 2568
- enqueueUsage
 - activemq::util::MemoryUsage, 1819
 - activemq::util::Usage, 2873
- ensureCapacity
 - decaf::util::ArrayList, 453
- ensureConnectionInfoSent
 - activemq::core::ActiveMQConnection, 197
- ensureCreated
 - decaf::net::ServerSocket, 2348
 - decaf::net::Socket, 2459
- entering

- decaf::util::logging::Logger, 1697
- eof
- gz_state, 1400
- equals
- activemq::commands::ActiveMQ-BlobMessage, 159
- activemq::commands::ActiveMQ-BytesMessage, 169
- activemq::commands::ActiveMQ-Destination, 250
- activemq::commands::ActiveMQ-MapMessage, 272
- activemq::commands::ActiveMQ-Message, 289
- activemq::commands::ActiveMQ-MessageTemplate, 297
- activemq::commands::ActiveMQ-ObjectMessage, 302
- activemq::commands::ActiveMQ-Queue, 323
- activemq::commands::ActiveMQ-StreamMessage, 362
- activemq::commands::ActiveMQ-TempDestination, 381
- activemq::commands::ActiveMQ-TempQueue, 389
- activemq::commands::ActiveMQ-TempTopic, 398
- activemq::commands::ActiveMQ-TextMessage, 407
- activemq::commands::ActiveMQ-Topic, 416
- activemq::commands::BaseCommand, 493
- activemq::commands::BaseData-Structure, 530
- activemq::commands::Boolean-Expression, 552
- activemq::commands::BrokerId, 566
- activemq::commands::BrokerInfo, 574
- activemq::commands::Connection-Control, 940
- activemq::commands::Connection-Error, 949
- activemq::commands::ConnectionId, 962
- activemq::commands::Connection-Info, 970
- activemq::commands::Consumer-Control, 993
- activemq::commands::ConsumerId, 1002, 1003
- activemq::commands::Consumer-Info, 1012
- activemq::commands::Control-Command, 1024
- activemq::commands::DataArray-Response, 1070
- activemq::commands::DataResponse, 1114
- activemq::commands::DataStructure, 1135
- activemq::commands::Destination-Info, 1215
- activemq::commands::Discovery-Event, 1228
- activemq::commands::Exception-Response, 1289
- activemq::commands::FlushCommand, 1382
- activemq::commands::Integer-Response, 1518
- activemq::commands::Journal-QueueAck, 1562
- activemq::commands::JournalTopic-Ack, 1570
- activemq::commands::JournalTrace, 1578
- activemq::commands::Journal-Transaction, 1585
- activemq::commands::KeepAliveInfo, 1592
- activemq::commands::LastPartial-Command, 1607
- activemq::commands::LocalTransaction-Id, 1676, 1677
- activemq::commands::Message, 1827
- activemq::commands::MessageAck, 1869
- activemq::commands::Message-Dispatch, 1883
- activemq::commands::Message-DispatchNotification, 1896
- activemq::commands::MessageId, 1910
- activemq::commands::MessagePull, 1941

- activemq::commands::Network-BridgeFilter, 1973
- activemq::commands::Partial-Command, 2077
- activemq::commands::ProducerAck, 2173
- activemq::commands::ProducerId, 2184
- activemq::commands::ProducerInfo, 2192
- activemq::commands::RemoveInfo, 2269
- activemq::commands::Remove-SubscriptionInfo, 2277
- activemq::commands::Replay-Command, 2285
- activemq::commands::Response, 2299
- activemq::commands::SessionId, 2380
- activemq::commands::SessionInfo, 2388
- activemq::commands::ShutdownInfo, 2433
- activemq::commands::Subscription-Info, 2632
- activemq::commands::TransactionId, 2768, 2769
- activemq::commands::Transaction-Info, 2776
- activemq::commands::WireFormat-Info, 2892
- activemq::commands::XATransaction-Id, 2939
- cms::Destination, 1212
- cms::Xid, 2948
- decaf::lang::Boolean, 547
- decaf::lang::Byte, 618
- decaf::lang::Character, 768
- decaf::lang::Comparable, 887
- decaf::lang::Double, 1241
- decaf::lang::Float, 1348, 1349
- decaf::lang::Integer, 1505
- decaf::lang::Long, 1731
- decaf::lang::Short, 2402
- decaf::net::URI, 2834
- decaf::nio::ByteBuffer, 700
- decaf::nio::CharBuffer, 793
- decaf::nio::DoubleBuffer, 1264
- decaf::nio::FloatBuffer, 1373
- decaf::nio::IntBuffer, 1493
- decaf::nio::LongBuffer, 1758
- decaf::nio::ShortBuffer, 2424
- decaf::security::cert::Certificate, 752
- decaf::security::Principal, 2160
- decaf::util::AbstractCollection, 114
- decaf::util::Collection, 859
- decaf::util::concurrent::Concurrent-StlMap, 911
- decaf::util::concurrent::CopyOn-WriteArrayList, 1038
- decaf::util::concurrent::CopyOn-WriteArraySet, 1057
- decaf::util::concurrent::Synchronous-Queue, 2660
- decaf::util::concurrent::TimeUnit, 2759
- decaf::util::Date, 1141
- decaf::util::logging::Level, 1618
- decaf::util::Map, 1772
- decaf::util::Properties, 2202
- decaf::util::StlList, 2546
- decaf::util::StlMap, 2558
- decaf::util::StlSet, 2579
- decaf::util::UUID, 2880
- err
 - decaf::io::FileDescriptor, 1332
 - gz_state, 1400
- error
 - decaf::util::logging::ErrorManager, 1278
 - decaf::util::logging::SimpleLogger, 2443
- exception
 - activemq::commands::Connection-Error, 951
 - activemq::commands::Exception-Response, 1290
- exclusive
 - activemq::commands::ActiveMQ-Destination, 256
 - activemq::commands::Consumer-Info, 1016
- execute
 - activemq::cmsutil::CmsTemplate, 840, 841
 - activemq::core::ActiveMQSession-Executor, 357
 - decaf::util::concurrent::Executor, 1299

- decaf::util::concurrent::ThreadPool-Executor, 2724
- executeAfterDelay
 - activemq::threads::Scheduler, 2319
- executeFirst
 - activemq::core::ActiveMQSession-Executor, 357
- executePeriodically
 - activemq::threads::Scheduler, 2319
- executor
 - activemq::core::ActiveMQSession, 354
- exit
 - activemq::commands::Connection-Control, 943
- exiting
 - decaf::util::logging::Logger, 1698
- expiration
 - activemq::commands::Message, 1838
- extra
 - gz_header_s, 1399
 - inflate_state, 1446
- extra_len
 - gz_header_s, 1399
- extra_max
 - gz_header_s, 1399
- faillfReadOnlyBody
 - activemq::commands::ActiveMQ-MessageTemplate, 297
- faillfReadOnlyProperties
 - activemq::commands::ActiveMQ-MessageTemplate, 297
- faillfWriteOnlyBody
 - activemq::commands::ActiveMQ-MessageTemplate, 297
- failoverReconnect
 - activemq::commands::Connection-Info, 973
- fatal
 - decaf::util::logging::SimpleLogger, 2443
- faultTolerant
 - activemq::commands::Connection-Control, 943
 - activemq::commands::Connection-Info, 973
- faultTolerantConfiguration
- activemq::commands::BrokerInfo, 578
- fc
 - ct_data_s, 1068
- fd
 - decaf::net::SocketImpl, 2487
 - gz_state, 1400
- fill
 - decaf::util::Arrays, 472
 - decaf::util::zip::InflaterInputStream, 1461
- find
 - decaf::internal::util::TimerTaskHeap, 2754
- findFactory
 - activemq::transport::Transport-Registry, 2813
 - activemq::wireformat::WireFormat-Registry, 2906
- fine
 - decaf::util::logging::Logger, 1698
- finer
 - decaf::util::logging::Logger, 1698
- finest
 - decaf::util::logging::Logger, 1699
- finish
 - decaf::util::zip::Deflater, 1184
 - decaf::util::zip::DeflaterOutput-Stream, 1193
 - decaf::util::zip::Inflater, 1450
- finished
 - decaf::util::zip::Deflater, 1184
 - decaf::util::zip::Inflater, 1450
- fire
 - activemq::core::ActiveMQConnection, 197
 - activemq::core::ActiveMQSession, 346
 - activemq::transport::TransportFilter, 2803
- fireCommand
 - activemq::transport::mock::Mock-Transport, 1951
- fireException
 - activemq::transport::mock::Mock-Transport, 1951
- firstMessageId
 - activemq::commands::MessageAck, 1872
- firstNakNumber

- activemq::commands::Replay-Command, 2287
- flags
 - inflate_state, 1446
- flip
 - decaf::nio::Buffer, 585
- floatToIntBits
 - decaf::lang::Float, 1349
- floatToRawIntBits
 - decaf::lang::Float, 1349
- floatValue
 - activemq::util::PrimitiveValueNode::PrimitiveValue, 2143
 - decaf::lang::Byte, 618
 - decaf::lang::Character, 768
 - decaf::lang::Double, 1241
 - decaf::lang::Float, 1350
 - decaf::lang::Integer, 1505
 - decaf::lang::Long, 1731
 - decaf::lang::Number, 1993
 - decaf::lang::Short, 2402
 - decaf::util::concurrent::atomic::AtomicInteger, 478
- floor
 - decaf::lang::Math, 1805
- flush
 - activemq::commands::Consumer-Control, 996
 - decaf::internal::io::StandardError-OutputStream, 2530
 - decaf::internal::io::StandardOutputStream, 2533
 - decaf::io::BufferedOutputStream, 597
 - decaf::io::FilterOutputStream, 1343
 - decaf::io::Flushable, 1380
 - decaf::io::OutputStream, 2069
 - decaf::io::OutputStreamWriter, 2075
 - decaf::util::logging::Handler, 1403
 - decaf::util::logging::StreamHandler, 2605
- forget
 - activemq::core::ActiveMQTransaction-Context, 427
 - cms::XAResource, 2929
- format
 - decaf::util::logging::Formatter, 1388
 - decaf::util::logging::SimpleFormatter, 2441
- decaf::util::logging::XMLFormatter, 2951
- formatId
 - activemq::commands::XATransaction-Id, 2942
- formatMessage
 - decaf::util::logging::Formatter, 1388
- freq
 - ct_data_s, 1068
- fromStream
 - activemq::wireformat::stomp::Stomp-Frame, 2589
- fromString
 - decaf::util::UUID, 2880
- front
 - decaf::util::StlQueue, 2568, 2569
- generateId
 - activemq::util::IdGenerator, 1412
- generation
 - decaf::util::concurrent::Condition-Handle, 928
- get
 - decaf::internal::nio::ByteArrayBuffer, 665
 - decaf::internal::nio::CharArrayBuffer, 781
 - decaf::internal::nio::DoubleArray-Buffer, 1256
 - decaf::internal::nio::FloatArrayBuffer, 1364, 1365
 - decaf::internal::nio::IntArrayBuffer, 1485
 - decaf::internal::nio::LongArrayBuffer, 1749, 1750
 - decaf::internal::nio::ShortArray-Buffer, 2416
 - decaf::internal::util::ByteArray-Adapter, 630
 - decaf::lang::ArrayPointer, 466
 - decaf::lang::Pointer, 2088
 - decaf::nio::ByteBuffer, 700, 701
 - decaf::nio::CharBuffer, 793, 794
 - decaf::nio::DoubleBuffer, 1264–1266
 - decaf::nio::FloatBuffer, 1373, 1374
 - decaf::nio::IntBuffer, 1493, 1494
 - decaf::nio::LongBuffer, 1758, 1759
 - decaf::nio::ShortBuffer, 2424, 2425
 - decaf::util::AbstractSequentialList, 148

- decaf::util::ArrayList, 453
- decaf::util::concurrent::atomic::-
 - AtomicBoolean, 474
- decaf::util::concurrent::atomic::-
 - AtomicInteger, 478
- decaf::util::concurrent::atomic::-
 - AtomicReference, 485
- decaf::util::concurrent::Concurrent-
 - StlMap, 912
- decaf::util::concurrent::CopyOn-
 - WriteArrayList, 1038
- decaf::util::concurrent::Future, 1391
- decaf::util::LinkedList, 1646
- decaf::util::List, 1662
- decaf::util::Map, 1773
- decaf::util::StlList, 2547
- decaf::util::StlMap, 2558, 2559
- getAckHandler
 - activemq::commands::Message, 1828
- getAckMode
 - activemq::commands::SessionInfo, 2388
- getAckType
 - activemq::commands::MessageAck, 1870
- getAcknowledgeMode
 - activemq::cmsutil::PooledSession, 2104
 - activemq::core::ActiveMQSession, 347
 - cms::Session, 2374
- getActiveCount
 - decaf::util::concurrent::ThreadPool-
 - Executor, 2724
- getAdditionalPredicate
 - activemq::commands::Consumer-
 - Info, 1012
- getAddress
 - decaf::net::DatagramPacket, 1082
 - decaf::net::InetAddress, 1440
- getAdler
 - decaf::util::zip::Deflater, 1184
 - decaf::util::zip::Inflater, 1450
- getAlgorithm
 - decaf::security::Key, 1599
- getAndAdd
 - decaf::util::concurrent::atomic::-
 - AtomicInteger, 479
- getAndDecrement
 - decaf::util::concurrent::atomic::-
 - AtomicInteger, 479
- getAndIncrement
 - decaf::util::concurrent::atomic::-
 - AtomicInteger, 479
- getAndSet
 - decaf::util::concurrent::atomic::-
 - AtomicBoolean, 474
 - decaf::util::concurrent::atomic::-
 - AtomicInteger, 479
 - decaf::util::concurrent::atomic::-
 - AtomicReference, 485
- getAnonymousLogger
 - decaf::util::logging::Logger, 1699
- getAnyAddress
 - decaf::net::InetAddress, 1440
- getAprPool
 - decaf::internal::AprPool, 445
- getArrival
 - activemq::commands::Message, 1828
- getAuthority
 - decaf::internal::net::URIType, 2862
 - decaf::net::URI, 2834
- getBackOffMultiplier
 - activemq::core::policies::Default-
 - RedeliveryPolicy, 1151
 - activemq::core::RedeliveryPolicy, 2252
 - activemq::transport::failover::Failover-
 - Transport, 1308
- getBackup
 - activemq::transport::failover::Backup-
 - TransportPool, 490
- getBackupPoolSize
 - activemq::transport::failover::Backup-
 - TransportPool, 490
 - activemq::transport::failover::Failover-
 - Transport, 1309
- getBasicConstraints
 - decaf::security::cert::X509Certificate, 2916
- getBody
 - activemq::wireformat::stomp::Stomp-
 - Frame, 2590
- getBodyBytes
 - activemq::commands::ActiveMQ-
 - BytesMessage, 169
 - cms::BytesMessage, 721
- getBodyLength

- activemq::commands::ActiveMQ-BytesMessage, 170
- activemq::wireformat::stomp::Stomp-Frame, 2590
- cms::BytesMessage, 721
- getBool
 - activemq::util::PrimitiveList, 2117
 - activemq::util::PrimitiveMap, 2128
 - activemq::util::PrimitiveValueNode, 2152
- getBoolean
 - activemq::commands::ActiveMQ-MapMessage, 273
 - cms::MapMessage, 1782
- getBooleanProperty
 - activemq::commands::ActiveMQ-MessageTemplate, 297
 - activemq::wireformat::openwire::utils::MessageProperty-Interceptor, 1934
 - cms::Message, 1844
- getBranchQualifier
 - activemq::commands::XATransaction-Id, 2939, 2940
 - cms::Xid, 2948
- getBrokerId
 - activemq::commands::BrokerInfo, 574
- getBrokerInTime
 - activemq::commands::Message, 1828
- getBrokerName
 - activemq::commands::BrokerInfo, 574, 575
 - activemq::commands::Discovery-Event, 1228
- getBrokerOutTime
 - activemq::commands::Message, 1828
- getBrokerPath
 - activemq::commands::Connection-Info, 970
 - activemq::commands::Consumer-Info, 1012
 - activemq::commands::Destination-Info, 1215, 1216
 - activemq::commands::Message, 1828
 - activemq::commands::ProducerInfo, 2192
- getBrokerSequenceId
 - activemq::commands::MessageId, 1910
- getBrokerURI
 - activemq::core::ActiveMQConnection-Factory, 219
- getBrokerURL
 - activemq::commands::BrokerInfo, 575
 - activemq::core::ActiveMQConnection, 198
- getBrokerUploadUrl
 - activemq::commands::BrokerInfo, 575
- getByAddress
 - decaf::net::InetAddress, 1440
- getByte
 - activemq::commands::ActiveMQ-MapMessage, 273
 - activemq::util::PrimitiveList, 2117
 - activemq::util::PrimitiveMap, 2128
 - activemq::util::PrimitiveValueNode, 2152
 - cms::MapMessage, 1782
- getByteArray
 - activemq::util::PrimitiveList, 2117
 - activemq::util::PrimitiveMap, 2129
 - activemq::util::PrimitiveValueNode, 2152
 - decaf::internal::util::ByteArray-Adapter, 630
- getByteProperty
 - activemq::commands::ActiveMQ-MessageTemplate, 297
 - activemq::wireformat::openwire::utils::MessageProperty-Interceptor, 1935
 - cms::Message, 1845
- getBytes
 - activemq::commands::ActiveMQ-MapMessage, 273
 - cms::MapMessage, 1782
- getBytesRead
 - decaf::util::zip::Deflater, 1184
 - decaf::util::zip::Inflater, 1450
- getBytesWritten
 - decaf::util::zip::Deflater, 1184
 - decaf::util::zip::Inflater, 1451
- getCMSCorrelationID

- activemq::commands::ActiveMQ-MessageTemplate, 297
- cms::Message, 1846
- getCMSDeliveryMode
 - activemq::commands::ActiveMQ-MessageTemplate, 297
 - cms::Message, 1846
- getCMSDestination
 - activemq::commands::ActiveMQ-Destination, 251
 - activemq::commands::ActiveMQ-MessageTemplate, 298
 - activemq::commands::ActiveMQ-Queue, 324
 - activemq::commands::ActiveMQ-TempQueue, 389
 - activemq::commands::ActiveMQ-TempTopic, 399
 - activemq::commands::ActiveMQ-Topic, 417
 - cms::Message, 1847
- getCMSExpiration
 - activemq::commands::ActiveMQ-MessageTemplate, 298
 - cms::Message, 1847
- getCMSMajorVersion
 - activemq::core::ActiveMQConnection-MetaData, 228
 - cms::ConnectionMetaData, 979
- getCMSMessageID
 - activemq::commands::ActiveMQ-MessageTemplate, 298
 - cms::Message, 1848
- getCMSMinorVersion
 - activemq::core::ActiveMQConnection-MetaData, 228
 - cms::ConnectionMetaData, 979
- getCMSPriority
 - activemq::commands::ActiveMQ-MessageTemplate, 298
 - cms::Message, 1849
- getCMSProperties
 - activemq::commands::ActiveMQ-Queue, 324
 - activemq::commands::ActiveMQ-TempQueue, 390
 - activemq::commands::ActiveMQ-TempTopic, 399
 - activemq::commands::ActiveMQ-Topic, 417
 - cms::Destination, 1212
- getCMSProviderName
 - activemq::core::ActiveMQConnection-MetaData, 228
 - cms::ConnectionMetaData, 979
- getCMSRedelivered
 - activemq::commands::ActiveMQ-MessageTemplate, 298
 - cms::Message, 1849
- getCMSReplyTo
 - activemq::commands::ActiveMQ-MessageTemplate, 298
 - cms::Message, 1850
- getCMSTimestamp
 - activemq::commands::ActiveMQ-MessageTemplate, 298
 - cms::Message, 1850
- getCMSType
 - activemq::commands::ActiveMQ-MessageTemplate, 298
 - cms::Message, 1851
- getCMSVersion
 - activemq::core::ActiveMQConnection-MetaData, 229
 - cms::ConnectionMetaData, 980
- getCMSXPropertyNames
 - activemq::core::ActiveMQConnection-MetaData, 229
 - cms::ConnectionMetaData, 980
- getCacheSize
 - activemq::commands::WireFormat-Info, 2893
 - activemq::wireformat::openwire::OpenWireFormat, 2050
- getCapacity
 - decaf::internal::util::ByteArray-Adapter, 631
- getCause
 - activemq::commands::BrokerError, 560
 - cms::CMSException, 827
 - decaf::lang::Exception, 1283
 - decaf::lang::Throwable, 2734
- getChar
 - activemq::commands::ActiveMQ-MapMessage, 274
 - activemq::util::PrimitiveList, 2118
 - activemq::util::PrimitiveMap, 2129
 - activemq::util::PrimitiveValueNode, 2152

- cms::MapMessage, 1783
- decaf::internal::nio::ByteBuffer, 666
- decaf::internal::util::ByteArray-Adapter, 631
- decaf::nio::ByteBuffer, 702
- getCharArray
 - decaf::internal::util::ByteArray-Adapter, 631
- getCharCapacity
 - decaf::internal::util::ByteArray-Adapter, 631
- getChecksum
 - decaf::util::zip::CheckedInputStream, 807
 - decaf::util::zip::CheckedOutputStream, 810
- getCipherSuites
 - decaf::net::ssl::SSLParameters, 2503
- getClientID
 - activemq::core::ActiveMQConnection, 198
 - cms::Connection, 936
- getClientId
 - activemq::commands::ActiveMQ-Destination, 250
 - activemq::commands::Connection-Info, 970
 - activemq::commands::JournalTopic-Ack, 1570
 - activemq::commands::Remove-SubscriptionInfo, 2277
 - activemq::commands::Subscription-Info, 2633
 - activemq::core::ActiveMQConnection-Factory, 219
- getCloseTimeout
 - activemq::core::ActiveMQConnection, 198
 - activemq::core::ActiveMQConnection-Factory, 219
- getCluster
 - activemq::commands::Message, 1828
- getCollisionAvoidancePercent
 - activemq::core::policies::Default-RedeliveryPolicy, 1151
 - activemq::core::RedeliveryPolicy, 2253
- getCommand
 - activemq::commands::Control-Command, 1024
 - activemq::wireformat::stomp::Stomp-Frame, 2590
- getCommandId
 - activemq::commands::BaseCommand, 494
 - activemq::commands::Command, 867
 - activemq::commands::Partial-Command, 2078
- getCommands
 - activemq::state::TransactionState, 2785
- getCompletedTaskCount
 - decaf::util::concurrent::ThreadPool-Executor, 2725
- getComponents
 - activemq::util::CompositeData, 891
- getCompressionLevel
 - activemq::core::ActiveMQConnection, 198
 - activemq::core::ActiveMQConnection-Factory, 219
- getConnectedBrokers
 - activemq::commands::Connection-Control, 940, 941
- getConnection
 - activemq::commands::Message, 1828
 - activemq::core::ActiveMQSession, 347
- getConnectionFactory
 - activemq::cmsutil::CmsAccessor, 822
- getConnectionId
 - activemq::commands::BrokerInfo, 575
 - activemq::commands::Connection-Error, 949, 950
 - activemq::commands::Connection-Info, 970, 971
 - activemq::commands::ConsumerId, 1003
 - activemq::commands::Destination-Info, 1216
 - activemq::commands::LocalTransaction-Id, 1677

- activemq::commands::ProducerId, 2184
- activemq::commands::Remove-SubscriptionInfo, 2277, 2278
- activemq::commands::SessionId, 2380
- activemq::commands::Transaction-Info, 2776
- activemq::core::ActiveMQConnection, 198
- getConnectionInfo
 - activemq::core::ActiveMQConnection, 199
- getConsumerId
 - activemq::commands::Consumer-Control, 994
 - activemq::commands::Consumer-Info, 1012
 - activemq::commands::MessageAck, 1870
 - activemq::commands::Message-Dispatch, 1883, 1884
 - activemq::commands::Message-DispatchNotification, 1897
 - activemq::commands::MessagePull, 1941, 1942
 - activemq::core::ActiveMQConsumer, 239
 - activemq::core::DispatchData, 1234
- getConsumerInfo
 - activemq::core::ActiveMQConsumer, 239
- getConsumerState
 - activemq::state::SessionState, 2396
- getConsumerStates
 - activemq::state::SessionState, 2396
- getContent
 - activemq::commands::Message, 1828, 1829
- getContext
 - decaf::internal::net::ssl::DefaultSSL-Context, 1165
- getCorePoolSize
 - decaf::util::concurrent::ThreadPool-Executor, 2725
- getCorrelationId
 - activemq::commands::Message, 1829
 - activemq::commands::MessagePull, 1942
- activemq::commands::Response, 2300
- getCount
 - decaf::util::concurrent::CountDown-Latch, 1065
- getData
 - activemq::commands::DataArray-Response, 1070, 1071
 - activemq::commands::DataResponse, 1114
 - activemq::commands::Partial-Command, 2078
 - decaf::net::DatagramPacket, 1082
- getDataStructure
 - activemq::commands::Message, 1829
- getDataStructureType
 - activemq::commands::ActiveMQ-BlobMessage, 159
 - activemq::commands::ActiveMQ-BytesMessage, 170
 - activemq::commands::ActiveMQ-Destination, 251
 - activemq::commands::ActiveMQ-MapMessage, 274
 - activemq::commands::ActiveMQ-Message, 290
 - activemq::commands::ActiveMQ-ObjectMessage, 303
 - activemq::commands::ActiveMQ-Queue, 324
 - activemq::commands::ActiveMQ-StreamMessage, 362
 - activemq::commands::ActiveMQ-TempDestination, 381
 - activemq::commands::ActiveMQ-TempQueue, 390
 - activemq::commands::ActiveMQ-TempTopic, 399
 - activemq::commands::ActiveMQ-TextMessage, 408
 - activemq::commands::ActiveMQ-Topic, 417
- activemq::commands::BrokerError, 560
- activemq::commands::BrokerId, 566
- activemq::commands::BrokerInfo, 575
- activemq::commands::Connection-Control, 941

- activemq::commands::Connection-Error, 950
- activemq::commands::ConnectionId, 962
- activemq::commands::Connection-Info, 971
- activemq::commands::Consumer-Control, 994
- activemq::commands::ConsumerId, 1003
- activemq::commands::Consumer-Info, 1013
- activemq::commands::Control-Command, 1024
- activemq::commands::DataArray-Response, 1071
- activemq::commands::DataResponse, 1114
- activemq::commands::DataStructure, 1137
- activemq::commands::Destination-Info, 1216
- activemq::commands::Discovery-Event, 1228
- activemq::commands::Exception-Response, 1289
- activemq::commands::FlushCommand, 1382
- activemq::commands::Integer-Response, 1518
- activemq::commands::Journal-QueueAck, 1562
- activemq::commands::JournalTopic-Ack, 1570
- activemq::commands::JournalTrace, 1578
- activemq::commands::Journal-Transaction, 1585
- activemq::commands::KeepAliveInfo, 1593
- activemq::commands::LastPartial-Command, 1607
- activemq::commands::LocalTransaction-Id, 1677
- activemq::commands::Message, 1829
- activemq::commands::MessageAck, 1870
- activemq::commands::Message-Dispatch, 1884
- activemq::commands::Message-DispatchNotification, 1897
- activemq::commands::MessageId, 1910
- activemq::commands::MessagePull, 1942
- activemq::commands::Network-BridgeFilter, 1973
- activemq::commands::Partial-Command, 2078
- activemq::commands::ProducerAck, 2173
- activemq::commands::ProducerId, 2184
- activemq::commands::ProducerInfo, 2192
- activemq::commands::RemoveInfo, 2269
- activemq::commands::Remove-SubscriptionInfo, 2278
- activemq::commands::Replay-Command, 2285
- activemq::commands::Response, 2300
- activemq::commands::SessionId, 2381
- activemq::commands::SessionInfo, 2388
- activemq::commands::ShutdownInfo, 2433
- activemq::commands::Subscription-Info, 2633
- activemq::commands::TransactionId, 2769
- activemq::commands::Transaction-Info, 2776
- activemq::commands::WireFormat-Info, 2893
- activemq::commands::XATransaction-Id, 2940
- activemq::wireformat::openwire-::marshal::DataStreamMarshaller, 1122
- activemq::wireformat::openwire-::marshal::generated::Active-MQBlobMessageMarshaller, 163
- activemq::wireformat::openwire-::marshal::generated::Active-MQBytesMessageMarshaller,

- 185
- activemq::wireformat::openwire-
::marshal::generated::Active-
MQMapMessageMarshaller, 285
- activemq::wireformat::openwire-
::marshal::generated::Active-
MQMessageMarshaller, 292
- activemq::wireformat::openwire-
::marshal::generated::Active-
MQObjectMessageMarshaller, 305
- activemq::wireformat::openwire-
::marshal::generated::Active-
MQQueueMarshaller, 330
- activemq::wireformat::openwire-
::marshal::generated::Active-
MQStreamMessageMarshaller, 376
- activemq::wireformat::openwire-
::marshal::generated::Active-
MQTempQueueMarshaller, 393
- activemq::wireformat::openwire-
::marshal::generated::Active-
MQTempTopicMarshaller, 402
- activemq::wireformat::openwire-
::marshal::generated::ActiveM-
QTextMessageMarshaller, 411
- activemq::wireformat::openwire-
::marshal::generated::Active-
MQTopicMarshaller, 420
- activemq::wireformat::openwire-
::marshal::generated::Broker-
IdMarshaller, 569
- activemq::wireformat::openwire-
::marshal::generated::Broker-
InfoMarshaller, 580
- activemq::wireformat::openwire-
::marshal::generated::Connection-
ControlMarshaller, 945
- activemq::wireformat::openwire-
::marshal::generated::Connection-
ErrorMarshaller, 952
- activemq::wireformat::openwire-
::marshal::generated::Connection-
IdMarshaller, 965
- activemq::wireformat::openwire-
::marshal::generated::Connection-
InfoMarshaller, 975
- activemq::wireformat::openwire-
::marshal::generated::Consumer-
ControlMarshaller, 998
- activemq::wireformat::openwire-
::marshal::generated::Consumer-
IdMarshaller, 1006
- activemq::wireformat::openwire-
::marshal::generated::Consumer-
InfoMarshaller, 1019
- activemq::wireformat::openwire-
::marshal::generated::Control-
CommandMarshaller, 1027
- activemq::wireformat::openwire-
::marshal::generated::Data-
ArrayResponseMarshaller, 1073
- activemq::wireformat::openwire-
::marshal::generated::Data-
ResponseMarshaller, 1116
- activemq::wireformat::openwire-
::marshal::generated::Destination-
InfoMarshaller, 1219
- activemq::wireformat::openwire-
::marshal::generated::Discovery-
EventMarshaller, 1231
- activemq::wireformat::openwire-
::marshal::generated::Exception-
ResponseMarshaller, 1292
- activemq::wireformat::openwire-
::marshal::generated::Flush-
CommandMarshaller, 1385
- activemq::wireformat::openwire-
::marshal::generated::Integer-
ResponseMarshaller, 1520
- activemq::wireformat::openwire-
::marshal::generated::Journal-
QueueAckMarshaller, 1565
- activemq::wireformat::openwire-
::marshal::generated::Journal-
TopicAckMarshaller, 1574
- activemq::wireformat::openwire-
::marshal::generated::Journal-
TraceMarshaller, 1581
- activemq::wireformat::openwire-
::marshal::generated::Journal-
TransactionMarshaller, 1588
- activemq::wireformat::openwire-
::marshal::generated::Keep-
AliveInfoMarshaller, 1595
- activemq::wireformat::openwire-
::marshal::generated::Last-

- PartialCommandMarshaller, 1609
- activemq::wireformat::openwire-
::marshal::generated::Local-
TransactionIdMarshaller, 1680
- activemq::wireformat::openwire-
::marshal::generated::Message-
AckMarshaller, 1874
- activemq::wireformat::openwire-
::marshal::generated::Message-
DispatchMarshaller, 1892
- activemq::wireformat::openwire-
::marshal::generated::Message-
DispatchNotificationMarshaller,
1901
- activemq::wireformat::openwire-
::marshal::generated::Message-
IdMarshaller, 1914
- activemq::wireformat::openwire-
::marshal::generated::Message-
PullMarshaller, 1945
- activemq::wireformat::openwire-
::marshal::generated::Network-
BridgeFilterMarshaller, 1976
- activemq::wireformat::openwire-
::marshal::generated::Partial-
CommandMarshaller, 2080
- activemq::wireformat::openwire-
::marshal::generated::Producer-
AckMarshaller, 2176
- activemq::wireformat::openwire-
::marshal::generated::Producer-
IdMarshaller, 2187
- activemq::wireformat::openwire-
::marshal::generated::Producer-
InfoMarshaller, 2196
- activemq::wireformat::openwire-
::marshal::generated::Remove-
InfoMarshaller, 2272
- activemq::wireformat::openwire-
::marshal::generated::Remove-
SubscriptionInfoMarshaller,
2281
- activemq::wireformat::openwire-
::marshal::generated::Replay-
CommandMarshaller, 2288
- activemq::wireformat::openwire-
::marshal::generated::Response-
Marshaller, 2308
- activemq::wireformat::openwire-
::marshal::generated::Session-
IdMarshaller, 2383
- activemq::wireformat::openwire-
::marshal::generated::Session-
InfoMarshaller, 2391
- activemq::wireformat::openwire-
::marshal::generated::Shutdown-
InfoMarshaller, 2435
- activemq::wireformat::openwire-
::marshal::generated::Subscription-
InfoMarshaller, 2636
- activemq::wireformat::openwire-
::marshal::generated::Transaction-
InfoMarshaller, 2779
- activemq::wireformat::openwire-
::marshal::generated::Wire-
FormatInfoMarshaller, 2901
- activemq::wireformat::openwire-
::marshal::generated::XA-
TransactionIdMarshaller, 2944
- getDefault
 - decaf::net::ServerSocketFactory,
2355
 - decaf::net::SocketFactory, 2477
 - decaf::net::ssl::SSLContext, 2496
 - decaf::net::ssl::SSLServerSocket-
Factory, 2511
 - decaf::net::ssl::SSLSocketFactory,
2523
- getDefaultBacklog
 - decaf::net::ServerSocket, 2348
- getDefaultCipherSuites
 - decaf::internal::net::ssl::DefaultSSL-
ServerSocketFactory, 1169
 - decaf::internal::net::ssl::DefaultSSL-
SocketFactory, 1177
 - decaf::internal::net::ssl::openssl:-
OpenSSLServerSocketFactory,
2013
 - decaf::internal::net::ssl::openssl:-
OpenSSLSocketFactory, 2039
 - decaf::net::ssl::SSLServerSocket-
Factory, 2512
 - decaf::net::ssl::SSLSocketFactory,
2524
- getDefaultDestination
 - activemq::cmsutil::CmsTemplate,
841, 842
- getDefaultDestinationName
 - activemq::cmsutil::CmsTemplate,

- 842
- getDefaultSSLParameters
 - decaf::net::ssl::SSLContext, 2497
- getDefaultThreadFactory
 - decaf::util::concurrent::Executors, 1300
- getDelay
 - decaf::util::concurrent::Delayed, 1194
- getDeliveryMode
 - activemq::cmsutil::CachedProducer, 740
 - activemq::cmsutil::CmsTemplate, 842
 - activemq::core::ActiveMQProducer, 310
 - cms::MessageProducer, 1926
- getDeliverySequenceId
 - activemq::commands::MessageDispatchNotification, 1897
- getDestination
 - activemq::cmsutil::CmsTemplate::ProducerExecutor, 2181
 - activemq::cmsutil::CmsTemplate::ReceiveExecutor, 2249
 - activemq::cmsutil::CmsTemplate::ResolveProducerExecutor, 2291
 - activemq::cmsutil::CmsTemplate::ResolveReceiveExecutor, 2292
- activemq::commands::ConsumerControl, 994
- activemq::commands::ConsumerInfo, 1013
- activemq::commands::DestinationInfo, 1216
- activemq::commands::JournalQueueAck, 1563
- activemq::commands::JournalTopicAck, 1570
- activemq::commands::Message, 1829
- activemq::commands::MessageAck, 1870
- activemq::commands::MessageDispatch, 1884
- activemq::commands::MessageDispatchNotification, 1897
- activemq::commands::MessagePull, 1942
- activemq::commands::ProducerInfo, 2193
- activemq::commands::SubscriptionInfo, 2633
- getDestinationResolver
 - activemq::cmsutil::CmsDestinationAccessor, 824, 825
- getDestinationType
 - activemq::commands::ActiveMQDestination, 251
 - activemq::commands::ActiveMQQueue, 324
 - activemq::commands::ActiveMQTempQueue, 390
 - activemq::commands::ActiveMQTempTopic, 399
 - activemq::commands::ActiveMQTopic, 417
 - cms::Destination, 1212
- getDisableMessageId
 - activemq::cmsutil::CachedProducer, 740
 - activemq::core::ActiveMQProducer, 310
 - cms::MessageProducer, 1926
- getDisableMessageTimeStamp
 - activemq::cmsutil::CachedProducer, 740
 - activemq::core::ActiveMQProducer, 310
 - cms::MessageProducer, 1927
- getDouble
 - activemq::commands::ActiveMQMapMessage, 275
 - activemq::util::PrimitiveList, 2118
 - activemq::util::PrimitiveMap, 2129
 - activemq::util::PrimitiveValueNode, 2153
 - cms::MapMessage, 1783
 - decaf::internal::nio::ByteBuffer, 667
 - decaf::internal::util::ByteArrayAdapter, 632
 - decaf::nio::ByteBuffer, 703
- getDoubleArray
 - decaf::internal::util::ByteArrayAdapter, 632
- getDoubleAt
 - decaf::internal::util::ByteArrayAdapter, 632

- getDoubleCapacity
 - decaf::internal::util::ByteArray-Adapter, 633
- getDoubleProperty
 - activemq::commands::ActiveMQ-MessageTemplate, 298
 - activemq::wireformat::openwire::utils::MessageProperty-Interceptor, 1935
 - cms::Message, 1852
- getDurableTopicPrefetch
 - activemq::core::policies::Default-PrefetchPolicy, 1147
 - activemq::core::PrefetchPolicy, 2112
- getEnabledCipherSuites
 - decaf::internal::net::ssl::openssl::OpenSSLParameters, 2002
 - decaf::internal::net::ssl::openssl::OpenSSLServerSocket, 2006
 - decaf::internal::net::ssl::openssl::OpenSSLSocket, 2021
 - decaf::net::ssl::SSLServerSocket, 2508
 - decaf::net::ssl::SSLSocket, 2517
- getEnabledProtocols
 - decaf::internal::net::ssl::openssl::OpenSSLParameters, 2002
 - decaf::internal::net::ssl::openssl::OpenSSLServerSocket, 2006
 - decaf::internal::net::ssl::openssl::OpenSSLSocket, 2021
 - decaf::net::ssl::SSLServerSocket, 2508
 - decaf::net::ssl::SSLSocket, 2517
- getEncoded
 - decaf::security::auth::x500::X500-Principal, 2914
 - decaf::security::cert::Certificate, 752
 - decaf::security::Key, 1600
- getEnumeration
 - activemq::core::ActiveMQQueue-Browser, 327
 - cms::QueueBrowser, 2228
- getErrorCode
 - cms::XAException, 2923
 - decaf::net::SocketError, 2471
- getErrorMessage
 - decaf::util::logging::Handler, 1403
- getErrorString
 - decaf::internal::net::ssl::openssl::OpenSSLSocketException, 2032
 - decaf::net::SocketError, 2471
- getException
 - activemq::commands::Connection-Error, 950
 - activemq::commands::Exception-Response, 1289, 1290
- getExceptionClass
 - activemq::commands::BrokerError, 561
- getExceptionListener
 - activemq::core::ActiveMQConnection, 199
 - activemq::core::ActiveMQConnection-Factory, 220
 - activemq::core::ActiveMQSession, 347
 - cms::Connection, 936
- getExclusiveOwnerThread
 - decaf::util::concurrent::locks::AbstractOwnableSynchronizer, 136
- getExpiration
 - activemq::commands::Message, 1829
- getFailureError
 - activemq::core::ActiveMQConsumer, 239
- getFileDescriptor
 - decaf::net::SocketImpl, 2483
- getFilter
 - decaf::util::logging::Handler, 1403
 - decaf::util::logging::Logger, 1699
- getFirst
 - decaf::util::Deque, 1200, 1201
 - decaf::util::LinkedList, 1646, 1647
- getFirstFailureError
 - activemq::core::ActiveMQConnection, 199
- getFirstMessageId
 - activemq::commands::MessageAck, 1870
- getFirstNakNumber
 - activemq::commands::Replay-Command, 2286
- getFloat
 - activemq::commands::ActiveMQ-MapMessage, 275

- activemq::util::PrimitiveList, 2119
- activemq::util::PrimitiveMap, 2130
- activemq::util::PrimitiveValueNode, 2153
- cms::MapMessage, 1784
- decaf::internal::nio::ByteBuffer, 668
- decaf::internal::util::ByteArray-Adapter, 633
- decaf::nio::ByteBuffer, 703, 704
- getFloatArray
 - decaf::internal::util::ByteArray-Adapter, 633
- getFloatAt
 - decaf::internal::util::ByteArray-Adapter, 634
- getFloatCapacity
 - decaf::internal::util::ByteArray-Adapter, 634
- getFloatProperty
 - activemq::commands::ActiveMQ-MessageTemplate, 298
 - activemq::wireformat::openwire-:utils::MessageProperty-Interceptor, 1936
 - cms::Message, 1852
- getFormat
 - decaf::security::Key, 1600
- getFormatId
 - activemq::commands::XATransaction-Id, 2940
 - cms::Xid, 2949
- getFormatter
 - decaf::util::logging::Handler, 1403
- getFragment
 - activemq::util::CompositeData, 891
 - decaf::internal::net::URIType, 2862
 - decaf::net::URI, 2834
- getGlobalLock
 - decaf::internal::DecafRuntime, 1143
- getGlobalPool
 - decaf::internal::AprPool, 445
 - decaf::internal::DecafRuntime, 1144
- getGlobalTransactionId
 - activemq::commands::XATransaction-Id, 2940, 2941
 - cms::Xid, 2949
- getGroupId
 - activemq::commands::Message, 1829, 1830
- getGroupSequence
 - activemq::commands::Message, 1830
- getHandlers
 - decaf::util::logging::Logger, 1700
- getHead
 - decaf::util::logging::Formatter, 1389
 - decaf::util::logging::XMLFormatter, 2951
- getHoldCount
 - decaf::util::concurrent::locks:-ReentrantLock, 2258
- getHost
 - activemq::util::CompositeData, 891
 - decaf::internal::net::URIType, 2863
 - decaf::net::URI, 2834
- getHostAddress
 - decaf::net::InetAddress, 1441
- getHostName
 - decaf::net::InetAddress, 1441
- getHostname
 - activemq::util::IdGenerator, 1412
- getId
 - activemq::state::TransactionState, 2785
 - decaf::lang::Thread, 2708
- getIndex
 - decaf::net::URISyntaxException, 2860
- getInetAddress
 - decaf::net::Socket, 2459
 - decaf::net::SocketImpl, 2483
- getInfo
 - activemq::state::ConnectionState, 983
 - activemq::state::ConsumerState, 1022
 - activemq::state::ProducerState, 2199
 - activemq::state::SessionState, 2396
- getInitialDelayTime
 - activemq::transport::inactivity:-InactivityMonitor, 1427
- getInitialReconnectDelay
 - activemq::transport::failover::Failover-Transport, 1309
- getInitialRedeliveryDelay
 - activemq::core::policies::Default-RedeliveryPolicy, 1151
 - activemq::core::RedeliveryPolicy, 2253

- getInput
 - decaf::net::URISyntaxException, 2860
- getInputStream
 - decaf::internal::net::ssl::openssl::OpenSSLSocket, 2021
 - decaf::internal::net::tcp::TcpSocket, 2683
 - decaf::net::Socket, 2459
 - decaf::net::SocketImpl, 2483
- getInstance
 - activemq::transport::mock::MockTransport, 1951
 - activemq::transport::TransportRegistry, 2813
 - activemq::wireformat::WireFormatRegistry, 2906
 - decaf::util::logging::LogWriter, 1725
- getInt
 - activemq::commands::ActiveMQMapMessage, 275
 - activemq::util::PrimitiveList, 2119
 - activemq::util::PrimitiveMap, 2130
 - activemq::util::PrimitiveValueNode, 2153
 - cms::MapMessage, 1784
 - decaf::internal::nio::ByteBuffer, 668, 669
 - decaf::internal::util::ByteArrayAdapter, 634
 - decaf::nio::ByteBuffer, 704, 705
- getIntArray
 - decaf::internal::util::ByteArrayAdapter, 635
- getIntAt
 - decaf::internal::util::ByteArrayAdapter, 635
- getIntCapacity
 - decaf::internal::util::ByteArrayAdapter, 635
- getIntProperty
 - activemq::commands::ActiveMQMessageTemplate, 298
 - activemq::wireformat::openwire::utils::MessagePropertyInterceptor, 1936
 - cms::Message, 1853
- getIssuerUniqueID
 - decaf::security::cert::X509Certificate, 2916
- getIssuerX500Principal
 - decaf::security::cert::X509Certificate, 2916
- getKeepAlive
 - decaf::net::Socket, 2460
- getKeepAliveTime
 - decaf::util::concurrent::ThreadPoolExecutor, 2725
- getKey
 - decaf::util::Map::Entry, 1274
- getKeyUsage
 - decaf::security::cert::X509Certificate, 2916
- getLargestPoolSize
 - decaf::util::concurrent::ThreadPoolExecutor, 2725
- getLast
 - decaf::util::Deque, 1201, 1202
 - decaf::util::LinkedList, 1647
- getLastDeliveredSequenceId
 - activemq::commands::RemoveInfo, 2269
 - activemq::core::ActiveMQConsumer, 239
 - activemq::core::ActiveMQSession, 347
- getLastMessageId
 - activemq::commands::MessageAck, 1870
- getLastNakNumber
 - activemq::commands::ReplayCommand, 2286
- getLastSequenceId
 - activemq::util::LongSequenceGenerator, 1765
- getLeastSignificantBits
 - decaf::util::UUID, 2880
- getLength
 - decaf::net::DatagramPacket, 1082
- getLevel
 - decaf::util::logging::Handler, 1403
 - decaf::util::logging::Logger, 1700
 - decaf::util::logging::LogRecord, 1720
- getLimit
 - activemq::util::MemoryUsage, 1820
- getList
 - activemq::util::PrimitiveValueNode, 2154
- getLocalAddress

- decaf::internal::net::tcp::TcpSocket, 2684
- decaf::net::Socket, 2460
- decaf::net::SocketImpl, 2483
- getLocalHost
 - decaf::net::InetAddress, 1441
- getLocalPort
 - decaf::net::ServerSocket, 2348
 - decaf::net::Socket, 2460
 - decaf::net::SocketImpl, 2484
- getLogManager
 - decaf::util::logging::LogManager, 1716
- getLogger
 - decaf::util::logging::Logger, 1700
 - decaf::util::logging::LogManager, 1715
- getLoggerName
 - decaf::util::logging::LogRecord, 1720
- getLoggerNames
 - decaf::util::logging::LogManager, 1716
- getLong
 - activemq::commands::ActiveMQ-MapMessage, 276
 - activemq::util::PrimitiveList, 2120
 - activemq::util::PrimitiveMap, 2131
 - activemq::util::PrimitiveValueNode, 2154
 - cms::MapMessage, 1784
 - decaf::internal::nio::ByteBuffer, 669, 670
 - decaf::internal::util::ByteArray-Adapter, 636
 - decaf::nio::ByteBuffer, 705
- getLongArray
 - decaf::internal::util::ByteArray-Adapter, 636
- getLongAt
 - decaf::internal::util::ByteArray-Adapter, 636
- getLongCapacity
 - decaf::internal::util::ByteArray-Adapter, 637
- getLongProperty
 - activemq::commands::ActiveMQ-MessageTemplate, 298
 - activemq::wireformat::openwire::utils::MessageProperty-Interceptor, 1936
 - cms::Message, 1854
- getLoopbackAddress
 - decaf::net::InetAddress, 1442
- getMagic
 - activemq::commands::WireFormat-Info, 2893
- getManaged
 - decaf::internal::util::GenericResource, 1398
- getMap
 - activemq::commands::ActiveMQ-MapMessage, 276, 277
 - activemq::util::PrimitiveValueNode, 2154
- getMapNames
 - activemq::commands::ActiveMQ-MapMessage, 277
 - cms::MapMessage, 1785
- getMarshaledForm
 - activemq::commands::BaseData-Structure, 530
 - activemq::wireformat::MarshalAware, 1794
- getMarshaledProperties
 - activemq::commands::Message, 1830
 - activemq::commands::WireFormat-Info, 2893
- getMaxCacheSize
 - activemq::state::ConnectionState-Tracker, 986
 - activemq::transport::failover::Failover-Transport, 1309
- getMaxInactivityDuration
 - activemq::commands::WireFormat-Info, 2893
 - activemq::wireformat::openwire::OpenWireFormat, 2050
- getMaxInactivityDurationInitalDelay
 - activemq::commands::WireFormat-Info, 2894
- getMaxInactivityDurationInitialDelay
 - activemq::wireformat::openwire::OpenWireFormat, 2050
- getMaxPrefetchLimit
 - activemq::core::policies::Default-PrefetchPolicy, 1147
 - activemq::core::PrefetchPolicy, 2112
- getMaxReconnectAttempts

- activemq::transport::failover::Failover-Transport, 1309
- getMaxReconnectDelay
 - activemq::transport::failover::Failover-Transport, 1309
- getMaximumPendingMessageLimit
 - activemq::commands::Consumer-Info, 1013
- getMaximumPoolSize
 - decaf::util::concurrent::ThreadPool-Executor, 2726
- getMaximumRedeliveries
 - activemq::core::policies::Default-RedeliveryPolicy, 1152
 - activemq::core::RedeliveryPolicy, 2253
- getMessage
 - activemq::cmsutil::CmsTemplate::ReceiveExecutor, 2249
 - activemq::commands::BrokerError, 561
 - activemq::commands::JournalTrace, 1578, 1579
 - activemq::commands::Message-Dispatch, 1884
 - activemq::core::DispatchData, 1234
 - cms::CMSException, 827
 - decaf::lang::Exception, 1284
 - decaf::lang::Throwable, 2735
 - decaf::util::logging::LogRecord, 1720
- getMessageAck
 - activemq::commands::Journal-QueueAck, 1563
- getMessageAvailableCount
 - activemq::core::ActiveMQConsumer, 240
- getMessageCount
 - activemq::commands::MessageAck, 1871
- getMessageId
 - activemq::commands::JournalTopic-Ack, 1571
 - activemq::commands::Message, 1830
 - activemq::commands::Message-DispatchNotification, 1897
 - activemq::commands::MessagePull, 1942
- getMessageListener
 - activemq::cmsutil::CachedConsumer, 736
 - activemq::core::ActiveMQConsumer, 240
 - cms::MessageConsumer, 1878
- getMessageProperties
 - activemq::commands::Message, 1830
- getMessageSelector
 - activemq::cmsutil::CachedConsumer, 736
 - activemq::core::ActiveMQConsumer, 240
 - activemq::core::ActiveMQQueue-Browser, 327
 - cms::MessageConsumer, 1878
 - cms::QueueBrowser, 2228
- getMessageSequencedId
 - activemq::commands::JournalTopic-Ack, 1571
- getMetaData
 - activemq::core::ActiveMQConnection, 199
 - cms::Connection, 937
- getMimeType
 - activemq::commands::ActiveMQ-BlobMessage, 159
- getMostSignificantBits
 - decaf::util::UUID, 2881
- getName
 - activemq::commands::ActiveMQ-BlobMessage, 159
 - activemq::transport::mock::Mock-Transport, 1951
 - decaf::lang::Thread, 2708
 - decaf::security::auth::x500::X500-Principal, 2914
 - decaf::security::Principal, 2160
 - decaf::util::concurrent::Mutex, 1962
 - decaf::util::logging::Level, 1618
 - decaf::util::logging::Logger, 1701
- getNeedClientAuth
 - decaf::internal::net::ssl::openssl::OpenSSLParameters, 2002
 - decaf::internal::net::ssl::openssl::OpenSSLServerSocket, 2007
 - decaf::internal::net::ssl::openssl::OpenSSLSocket, 2022
 - decaf::net::ssl::SSLParameters, 2503

- decaf::net::ssl::SSLServerSocket, 2508
- decaf::net::ssl::SSLSocket, 2517
- getNetworkBrokerId
 - activemq::commands::Network-BridgeFilter, 1973
- getNetworkConsumerPath
 - activemq::commands::Consumer-Info, 1013
- getNetworkProperties
 - activemq::commands::BrokerInfo, 575
- getNetworkRuntime
 - decaf::internal::net::Network, 1970
- getNetworkTTL
 - activemq::commands::Network-BridgeFilter, 1973
- getNextConsumerId
 - activemq::core::ActiveMQSession, 347
- getNextLocalTransactionId
 - activemq::core::ActiveMQConnection, 200
- getNextProducerId
 - activemq::core::ActiveMQSession, 348
- getNextRedeliveryDelay
 - activemq::core::policies::Default-RedeliveryPolicy, 1152
 - activemq::core::RedeliveryPolicy, 2253
- getNextSequenceId
 - activemq::util::LongSequence-Generator, 1765
- getNextSessionId
 - activemq::core::ActiveMQConnection, 200
- getNextTempDestinationId
 - activemq::core::ActiveMQConnection, 200
- getNotAfter
 - decaf::security::cert::X509Certificate, 2916
- getNotBefore
 - decaf::security::cert::X509Certificate, 2916
- getNumReceivedMessageBeforeFail
 - activemq::transport::mock::Mock-Transport, 1951
- getNumReceivedMessages
 - activemq::transport::mock::Mock-Transport, 1951
- getNumSentKeepAlives
 - activemq::transport::mock::Mock-Transport, 1951
- getNumSentKeepAlivesBeforeFail
 - activemq::transport::mock::Mock-Transport, 1952
- getNumSentMessageBeforeFail
 - activemq::transport::mock::Mock-Transport, 1952
- getNumSentMessages
 - activemq::transport::mock::Mock-Transport, 1952
- getOOBInline
 - decaf::net::Socket, 2461
- getObjectId
 - activemq::commands::RemoveInfo, 2270
- getOffset
 - decaf::net::DatagramPacket, 1083
- getOperationType
 - activemq::commands::Destination-Info, 1216
- getOption
 - decaf::internal::net::tcp::TcpSocket, 2684
 - decaf::net::SocketImpl, 2484
- getOptions
 - activemq::commands::ActiveMQ-Destination, 252
- getOrderedTarget
 - activemq::commands::ActiveMQ-Destination, 252
- getOriginalDestination
 - activemq::commands::Message, 1830
- getOriginalTransactionId
 - activemq::commands::Message, 1830
- getOutputStream
 - decaf::internal::net::ssl::openssl::OpenSSLSocket, 2022
 - decaf::internal::net::tcp::TcpSocket, 2684
 - decaf::net::Socket, 2461
 - decaf::net::SocketImpl, 2484
- getParameters
 - activemq::util::CompositeData, 891
- getParent

- decaf::util::logging::Logger, 1701
- getParentId
 - activemq::commands::ConsumerId, 1003
 - activemq::commands::ProducerId, 2184
 - activemq::commands::SessionId, 2381
- getPassword
 - activemq::commands::Connection-Info, 971
 - activemq::core::ActiveMQConnection, 201
 - activemq::core::ActiveMQConnection-Factory, 220
- getPath
 - activemq::util::CompositeData, 891
 - decaf::internal::net::URIType, 2863
 - decaf::net::URI, 2834
- getPeerBrokerInfos
 - activemq::commands::BrokerInfo, 575
- getPhysicalName
 - activemq::commands::ActiveMQ-Destination, 252
- getPoolSize
 - decaf::util::concurrent::ThreadPool-Executor, 2726
- getPort
 - decaf::internal::net::URIType, 2863
 - decaf::net::DatagramPacket, 1083
 - decaf::net::Socket, 2461
 - decaf::net::SocketImpl, 2485
 - decaf::net::URI, 2834
- getPreferredWireFormatInfo
 - activemq::wireformat::openwire::OpenWireFormat, 2050
- getPrefetch
 - activemq::commands::Consumer-Control, 994
- getPrefetchPolicy
 - activemq::core::ActiveMQConnection, 201
 - activemq::core::ActiveMQConnection-Factory, 220
- getPrefetchSize
 - activemq::commands::Consumer-Info, 1013
- getPreparedResult
 - activemq::state::TransactionState, 2786
- getPriority
 - activemq::cmsutil::CachedProducer, 741
 - activemq::cmsutil::CmsTemplate, 842
 - activemq::commands::Consumer-Info, 1013
 - activemq::commands::Message, 1831
 - activemq::core::ActiveMQProducer, 311
 - cms::MessageProducer, 1927
 - decaf::lang::Thread, 2709
- getProducerId
 - activemq::commands::Message, 1831
 - activemq::commands::MessageId, 1911
 - activemq::commands::ProducerAck, 2173
 - activemq::commands::ProducerInfo, 2193
 - activemq::core::ActiveMQProducer, 311
- getProducerInfo
 - activemq::core::ActiveMQProducer, 311
- getProducerSequenceId
 - activemq::commands::MessageId, 1911
- getProducerState
 - activemq::state::SessionState, 2397
- getProducerStates
 - activemq::state::SessionState, 2397
 - activemq::state::TransactionState, 2786
- getProducerWindowSize
 - activemq::core::ActiveMQConnection, 201
 - activemq::core::ActiveMQConnection-Factory, 220
- getProperties
 - activemq::commands::WireFormat-Info, 2894
 - activemq::core::ActiveMQConnection, 201
 - activemq::util::ActiveMQProperties, 318

- activemq::wireformat::stomp::Stomp-Frame, 2590
- decaf::lang::System, 2673
- decaf::util::logging::LogManager, 1716
- getProperty
 - activemq::util::ActiveMQProperties, 318
 - activemq::wireformat::stomp::Stomp-Frame, 2591
 - cms::CMSProperties, 832
 - decaf::lang::System, 2673, 2674
 - decaf::util::logging::LogManager, 1716
 - decaf::util::Properties, 2203
- getPropertyNames
 - activemq::commands::ActiveMQ-MessageTemplate, 299
 - cms::Message, 1854
- getProtocols
 - decaf::net::ssl::SSLParameters, 2503
- getProviderMajorVersion
 - activemq::core::ActiveMQConnection-MetaData, 229
 - cms::ConnectionMetaData, 981
- getProviderMinorVersion
 - activemq::core::ActiveMQConnection-MetaData, 230
 - cms::ConnectionMetaData, 981
- getProviderVersion
 - activemq::core::ActiveMQConnection-MetaData, 230
 - cms::ConnectionMetaData, 981
- getPublicKey
 - decaf::security::cert::Certificate, 752
- getQuery
 - decaf::internal::net::URIType, 2863
 - decaf::net::URI, 2834
- getQueue
 - activemq::core::ActiveMQQueue-Browser, 327
 - cms::QueueBrowser, 2228
 - decaf::util::concurrent::ThreadPool-Executor, 2726
- getQueueBrowserPrefetch
 - activemq::core::policies::Default-PrefetchPolicy, 1147
 - activemq::core::PrefetchPolicy, 2112
- getQueueName
 - activemq::commands::ActiveMQ-Queue, 325
 - activemq::commands::ActiveMQ-TempQueue, 390
 - cms::Queue, 2222
 - cms::TemporaryQueue, 2699
- getQueuePrefetch
 - activemq::core::policies::Default-PrefetchPolicy, 1148
 - activemq::core::PrefetchPolicy, 2112
- getRawAuthority
 - decaf::net::URI, 2834
- getRawFragment
 - decaf::net::URI, 2835
- getRawPath
 - decaf::net::URI, 2835
- getRawQuery
 - decaf::net::URI, 2835
- getRawSchemeSpecificPart
 - decaf::net::URI, 2835
- getRawUserInfo
 - decaf::net::URI, 2836
- getReadCheckTime
 - activemq::transport::inactivity::InactivityMonitor, 1427
- getReason
 - decaf::net::URISyntaxException, 2860
- getReceiveBufferSize
 - decaf::net::ServerSocket, 2348
 - decaf::net::Socket, 2461
- getReceiveTimeout
 - activemq::cmsutil::CmsTemplate, 842
- getReconnectDelay
 - activemq::transport::failover::Failover-Transport, 1309
- getReconnectTo
 - activemq::commands::Connection-Control, 941
- getRecoveringPullConsumers
 - activemq::state::ConnectionState, 983
- getRedeliveryCounter
 - activemq::commands::Message, 1831
 - activemq::commands::Message-Dispatch, 1884
- getRedeliveryDelay

- activemq::core::policies::Default-RedeliveryPolicy, 1152
- activemq::core::RedeliveryPolicy, 2254
- getRedeliveryPolicy
 - activemq::core::ActiveMQConnection, 201
 - activemq::core::ActiveMQConnectionFactory, 221
 - activemq::core::ActiveMQConsumer, 240
- getRejectedExecutionHandler
 - decaf::util::concurrent::ThreadPool-Executor, 2726
- getRemaining
 - decaf::util::zip::Inflater, 1451
- getRemoteAddress
 - activemq::transport::failover::Failover-Transport, 1309
 - activemq::transport::IOTransport, 1551
 - activemq::transport::mock::Mock-Transport, 1952
 - activemq::transport::Transport, 2792
 - activemq::transport::TransportFilter, 2803
- getRemoteBlobUrl
 - activemq::commands::ActiveMQ-BlobMessage, 160
- getReplyTo
 - activemq::commands::Message, 1831
- getResourceLifecycleManager
 - activemq::cmsutil::CmsAccessor, 822
 - activemq::cmsutil::SessionPool, 2395
- getResourceManagerId
 - activemq::core::ActiveMQConnection, 202
- getResponse
 - activemq::transport::correlator::-FutureResponse, 1393, 1394
- getResult
 - activemq::commands::Integer-Response, 1518
- getReuseAddress
 - decaf::net::ServerSocket, 2349
 - decaf::net::Socket, 2462
- getRuntime
 - decaf::lang::Runtime, 2313
- getRuntimeLock
 - decaf::internal::net::Network, 1970
- getSSLParameters
 - decaf::net::ssl::SSLSocket, 2517
- getSafeValue
 - decaf::util::StlQueue, 2569
- getScheduler
 - activemq::core::ActiveMQConnection, 202
 - activemq::core::ActiveMQSession, 348
- getScheme
 - activemq::util::CompositeData, 891
 - decaf::internal::net::URIType, 2863
 - decaf::net::URI, 2836
- getSchemeSpecificPart
 - decaf::internal::net::URIType, 2863
 - decaf::net::URI, 2836
- getSeedFromId
 - activemq::util::IdGenerator, 1412
- getSelector
 - activemq::commands::Consumer-Info, 1013
 - activemq::commands::Subscription-Info, 2633
- getSendBufferSize
 - decaf::net::Socket, 2462
- getSendTimeout
 - activemq::core::ActiveMQConnection, 202
 - activemq::core::ActiveMQConnectionFactory, 221
 - activemq::core::ActiveMQProducer, 311
- getSequenceFromId
 - activemq::util::IdGenerator, 1413
- getServerSocketFactory
 - decaf::net::ssl::SSLContext, 2497
- getServiceName
 - activemq::commands::Discovery-Event, 1229
- getSession
 - activemq::cmsutil::PooledSession, 2104, 2105
- getSessionAcknowledgeMode
 - activemq::cmsutil::CmsAccessor, 822
- getSessionId

- activemq::commands::ConsumerId, 1003
- activemq::commands::ProducerId, 2185
- activemq::commands::SessionInfo, 2388
- activemq::core::ActiveMQSession, 348
- getSessionInfo
 - activemq::core::ActiveMQSession, 348
- getSessionState
 - activemq::state::ConnectionState, 983
- getSessionStates
 - activemq::state::ConnectionState, 983
- getShort
 - activemq::commands::ActiveMQ-MapMessage, 277
 - activemq::util::PrimitiveList, 2120
 - activemq::util::PrimitiveMap, 2131
 - activemq::util::PrimitiveValueNode, 2155
 - cms::MapMessage, 1785
 - decaf::internal::nio::ByteBuffer, 670
 - decaf::internal::util::ByteArray-Adapter, 637
 - decaf::nio::ByteBuffer, 706
- getShortArray
 - decaf::internal::util::ByteArray-Adapter, 637
- getShortAt
 - decaf::internal::util::ByteArray-Adapter, 638
- getShortCapacity
 - decaf::internal::util::ByteArray-Adapter, 638
- getShortProperty
 - activemq::commands::ActiveMQ-MessageTemplate, 299
 - activemq::wireformat::openwire-::utils::MessageProperty-Interceptor, 1936
 - cms::Message, 1855
- getSigAlgName
 - decaf::security::cert::X509Certificate, 2916
- getSigAlgOID
 - decaf::security::cert::X509Certificate, 2916
- getSigAlgParams
 - decaf::security::cert::X509Certificate, 2916
- getSignature
 - decaf::security::cert::X509Certificate, 2916
- getSize
 - activemq::commands::ActiveMQ-TextMessage, 408
 - activemq::commands::Message, 1831
 - activemq::commands::ProducerAck, 2173
 - decaf::net::DatagramPacket, 1083
- getSoLinger
 - decaf::net::Socket, 2462
- getSoTimeout
 - decaf::net::ServerSocket, 2349
 - decaf::net::Socket, 2463
- getSocketAddress
 - decaf::net::DatagramPacket, 1083
- getSocketFactory
 - decaf::net::ssl::SSLContext, 2497
- getSocketHandle
 - decaf::internal::net::tcp::TcpSocket, 2685
- getSource
 - decaf::internal::net::URIType, 2864
- getSourceFile
 - decaf::util::logging::LogRecord, 1721
- getSourceFunction
 - decaf::util::logging::LogRecord, 1721
- getSourceLine
 - decaf::util::logging::LogRecord, 1721
- getStackTrace
 - cms::CMSException, 828
 - decaf::lang::Exception, 1284
 - decaf::lang::Throwable, 2735
- getStackTraceElements
 - activemq::commands::BrokerError, 561
- getStackTraceString
 - cms::CMSException, 828
 - decaf::lang::Exception, 1284
 - decaf::lang::Throwable, 2735
- getStartupMaxReconnectAttempts
 - activemq::transport::failover::Failover-Transport, 1309

- getState
 - decaf::lang::Thread, 2709
- getString
 - activemq::commands::ActiveMQ-MapMessage, 277
 - activemq::util::PrimitiveList, 2121
 - activemq::util::PrimitiveMap, 2132
 - activemq::util::PrimitiveValueNode, 2155
 - cms::MapMessage, 1786
- getStringProperty
 - activemq::commands::ActiveMQ-MessageTemplate, 299
 - activemq::wireformat::openwire::utils::MessageProperty-Interceptor, 1937
 - cms::Message, 1855
- getSubscriptionName
 - activemq::commands::Remove-SubscriptionInfo, 2278
 - activemq::commands::Subscription-Info, 2633
- getSubjectUniqueID
 - decaf::security::cert::X509Certificate, 2916
- getSubjectX500Principal
 - decaf::security::cert::X509Certificate, 2916
- getSubscribedDestination
 - activemq::commands::Subscription-Info, 2634
- getSubscriptionName
 - activemq::commands::Consumer-Info, 1013, 1014
- getSubscriptionName
 - activemq::commands::JournalTopic-Ack, 1571
- getSupportedCipherSuites
 - decaf::internal::net::ssl::DefaultSSL-ServerSocketFactory, 1169
 - decaf::internal::net::ssl::DefaultSSL-SocketFactory, 1177
 - decaf::internal::net::ssl::openssl:-OpenSSLParameters, 2002
 - decaf::internal::net::ssl::openssl:-OpenSSLServerSocket, 2007
 - decaf::internal::net::ssl::openssl:-OpenSSLServerSocketFactory, 2014
 - decaf::internal::net::ssl::openssl:-OpenSSLSocket, 2023
 - decaf::internal::net::ssl::openssl:-OpenSSLSocketFactory, 2039
 - decaf::net::ssl::SSLServerSocket, 2508
 - decaf::net::ssl::SSLServerSocket-Factory, 2512
 - decaf::net::ssl::SSLSocket, 2518
 - decaf::net::ssl::SSLSocketFactory, 2524
- getSupportedProtocols
 - decaf::internal::net::ssl::openssl:-OpenSSLParameters, 2002
 - decaf::internal::net::ssl::openssl:-OpenSSLServerSocket, 2007
 - decaf::internal::net::ssl::openssl:-OpenSSLSocket, 2023
 - decaf::net::ssl::SSLServerSocket, 2508
 - decaf::net::ssl::SSLSocket, 2518
- getSupportedSSLParameters
 - decaf::net::ssl::SSLContext, 2498
- getTBSCertificate
 - decaf::security::cert::X509Certificate, 2917
- getTail
 - decaf::util::logging::Formatter, 1389
 - decaf::util::logging::XMLFormatter, 2951
- getTargetConsumerId
 - activemq::commands::Message, 1831
- getTaskCount
 - decaf::util::concurrent::ThreadPool-Executor, 2726
- getTcpNoDelay
 - decaf::net::Socket, 2463
- getTempDesinations
 - activemq::state::ConnectionState, 983
- getText
 - activemq::commands::ActiveMQ-TextMessage, 408
 - cms::TextMessage, 2702
- getThreadFactory
 - decaf::util::concurrent::ThreadPool-Executor, 2727
- getThrown
 - decaf::util::logging::LogRecord, 1721

- getTime
 - decaf::util::Date, 1141
- getTimeToLive
 - activemq::cmsutil::CachedProducer, 741
 - activemq::cmsutil::CmsTemplate, 842
 - activemq::core::ActiveMQProducer, 311
 - cms::MessageProducer, 1928
- getTimeout
 - activemq::commands::Destination-Info, 1216
 - activemq::commands::MessagePull, 1942
 - activemq::transport::failover::Failover-Transport, 1309
- getTimestamp
 - activemq::commands::Message, 1831
 - decaf::util::logging::LogRecord, 1721
- getTopicName
 - activemq::commands::ActiveMQ-TempTopic, 400
 - activemq::commands::ActiveMQ-Topic, 418
 - cms::TemporaryTopic, 2701
 - cms::Topic, 2765
- getTopicPrefetch
 - activemq::core::policies::Default-PrefetchPolicy, 1148
 - activemq::core::PrefetchPolicy, 2113
- getTrafficClass
 - decaf::net::Socket, 2463
- getTransactionContext
 - activemq::core::ActiveMQSession, 348
- getTransactionId
 - activemq::commands::JournalTopic-Ack, 1571
 - activemq::commands::Journal-Transaction, 1586
 - activemq::commands::Message, 1831
 - activemq::commands::MessageAck, 1871
 - activemq::commands::Transaction-Info, 2776
 - activemq::core::ActiveMQTransaction-Context, 427
- getTransactionState
 - activemq::state::ConnectionState, 983
 - activemq::state::ProducerState, 2199
- getTransactionStates
 - activemq::state::ConnectionState, 983
- getTransactionTimeout
 - activemq::core::ActiveMQTransaction-Context, 427
 - cms::XAResource, 2930
- getTransport
 - activemq::core::ActiveMQConnection, 202
 - activemq::transport::failover::Backup-Transport, 487
- getTransportListener
 - activemq::transport::failover::Failover-Transport, 1309
 - activemq::transport::IOTransport, 1551
 - activemq::transport::mock::Mock-Transport, 1952
 - activemq::transport::Transport, 2792
 - activemq::transport::TransportFilter, 2803
- getTransportNames
 - activemq::transport::Transport-Registry, 2814
- getTreadId
 - decaf::util::logging::LogRecord, 1722
- getType
 - activemq::commands::Journal-Transaction, 1586
 - activemq::commands::Message, 1832
 - activemq::commands::Transaction-Info, 2776
 - activemq::util::PrimitiveValueNode, 2155
 - decaf::security::cert::Certificate, 753
- getURI
 - activemq::transport::failover::URI-Pool, 2853
- getUncaughtExceptionHandler
 - decaf::lang::Thread, 2709
- getUnconsumedMessages
 - activemq::core::ActiveMQSession-Executor, 357
- getUri

- activemq::transport::failover::Backup-Transport, 487
- getUsage
 - activemq::util::MemoryUsage, 1820
- getUseClientMode
 - decaf::internal::net::ssl::openssl::-OpenSSLParameters, 2002
 - decaf::internal::net::ssl::openssl::-OpenSSLSocket, 2023
 - decaf::net::ssl::SSLSocket, 2518
- getUseParentHandlers
 - decaf::util::logging::Logger, 1701
- getUserID
 - activemq::commands::Message, 1832
- getUserInfo
 - decaf::internal::net::URIType, 2864
 - decaf::net::URI, 2836
- getUserName
 - activemq::commands::Connection-Info, 971
- getUsername
 - activemq::core::ActiveMQConnection, 202
 - activemq::core::ActiveMQConnection-Factory, 221
- getValue
 - activemq::commands::BrokerId, 566, 567
 - activemq::commands::ConnectionId, 962
 - activemq::commands::ConsumerId, 1003
 - activemq::commands::LocalTransaction-Id, 1677
 - activemq::commands::ProducerId, 2185
 - activemq::commands::SessionId, 2381
 - activemq::util::PrimitiveValueNode, 2155
 - decaf::internal::net::SocketFile-Descriptor, 2479
 - decaf::util::Map::Entry, 1274
 - decaf::util::zip::Adler32, 440
 - decaf::util::zip::Checksum, 811
 - decaf::util::zip::CRC32, 1066
- getVersion
 - activemq::commands::WireFormat-Info, 2894
 - activemq::wireformat::openwire::-OpenWireFormat, 2051
 - activemq::wireformat::stomp::Stomp-WireFormat, 2599
 - activemq::wireformat::WireFormat, 2886
 - decaf::security::cert::X509Certificate, 2917
 - getWantClientAuth
 - decaf::internal::net::ssl::openssl::-OpenSSLParameters, 2002
 - decaf::internal::net::ssl::openssl::-OpenSSLServerSocket, 2007
 - decaf::internal::net::ssl::openssl::-OpenSSLSocket, 2023
 - decaf::net::ssl::SSLParameters, 2503
 - decaf::net::ssl::SSLServerSocket, 2509
 - decaf::net::ssl::SSLSocket, 2518
 - getWasPrepared
 - activemq::commands::Journal-Transaction, 1586
 - getWhen
 - decaf::util::TimerTask, 2752
 - getWindowSize
 - activemq::commands::ProducerInfo, 2193
 - getWireFormat
 - activemq::transport::failover::Failover-Transport, 1309
 - activemq::transport::IOTransport, 1551
 - activemq::transport::mock::Mock-Transport, 1952
 - activemq::transport::Transport, 2792
 - activemq::transport::TransportFilter, 2803
 - getWireFormatNames
 - activemq::wireformat::WireFormat-Registry, 2906
 - getWriteCheckTime
 - activemq::transport::inactivity::-InactivityMonitor, 1427
 - getXAResource
 - activemq::core::ActiveMQXASession, 438
 - cms::XASession, 2936
 - getenv
 - decaf::lang::System, 2672, 2673

- globalTransactionId
 - activemq::commands::XATransaction-Id, 2942
- good_match
 - internal_state, 1525
- groupID
 - activemq::commands::Message, 1838
- groupSequence
 - activemq::commands::Message, 1838
- gz_header
 - zlib.h, 3158
- gz_header_s, 1398
 - comm_max, 1399
 - comment, 1399
 - done, 1399
 - extra, 1399
 - extra_len, 1399
 - extra_max, 1399
 - hcrc, 1399
 - name, 1399
 - name_max, 1399
 - os, 1399
 - text, 1399
 - time, 1399
 - xflags, 1399
- gz_headerp
 - zlib.h, 3158
- gz_state, 1400
 - direct, 1400
 - eof, 1400
 - err, 1400
 - fd, 1400
 - have, 1400
 - how, 1400
 - in, 1400
 - level, 1400
 - mode, 1401
 - msg, 1401
 - next, 1401
 - out, 1401
 - path, 1401
 - pos, 1401
 - raw, 1401
 - seek, 1401
 - size, 1401
 - skip, 1401
 - start, 1401
 - strategy, 1401
 - strm, 1401
 - want, 1401
- gz_statep
 - gzguts.h, 3147
- gzFile
 - zlib.h, 3158
- gzguts.h
 - COPY, 3147
 - GT_OFF, 3147
 - GZBUFSIZE, 3147
 - GZIP, 3147
 - GZ_APPEND, 3147
 - GZ_NONE, 3147
 - GZ_READ, 3147
 - GZ_WRITE, 3147
 - LOOK, 3147
 - OF, 3147, 3148
 - ZLIB_INTERNAL, 3147
 - gz_statep, 3147
 - local, 3147
 - zstrerror, 3147
- gzhead
 - internal_state, 1525
- gzindex
 - internal_state, 1525
- handleConnectionControl
 - activemq::transport::failover::Failover-Transport, 1310
- handleTransportFailure
 - activemq::transport::failover::Failover-Transport, 1310
- hasArray
 - decaf::internal::nio::ByteBuffer, 671
 - decaf::internal::nio::CharArrayBuffer, 782
 - decaf::internal::nio::DoubleArray-Buffer, 1257
 - decaf::internal::nio::FloatArrayBuffer, 1365
 - decaf::internal::nio::IntArrayBuffer, 1485
 - decaf::internal::nio::LongArrayBuffer, 1750
 - decaf::internal::nio::ShortArray-Buffer, 2417
 - decaf::nio::ByteBuffer, 707
 - decaf::nio::CharBuffer, 795
 - decaf::nio::DoubleBuffer, 1266

- decaf::nio::FloatBuffer, 1375
- decaf::nio::IntBuffer, 1495
- decaf::nio::LongBuffer, 1760
- decaf::nio::ShortBuffer, 2426
- hasMoreMessages
 - activemq::core::ActiveMQQueue-
Browser, 328
 - cms::MessageEnumeration, 1904
- hasMoreTokens
 - decaf::util::StringTokenizer, 2629
- hasNegotiator
 - activemq::wireformat::openwire::-
OpenWireFormat, 2051
 - activemq::wireformat::stomp::Stomp-
WireFormat, 2599
 - activemq::wireformat::WireFormat,
2886
- hasNext
 - decaf::util::concurrent::CopyOn-
WriteArrayList::ArrayListIterator,
459
 - decaf::util::Iterator, 1559
- hasPrevious
 - decaf::util::concurrent::CopyOn-
WriteArrayList::ArrayListIterator,
459
 - decaf::util::ListIterator, 1672
- hasProperty
 - activemq::util::ActiveMQProperties,
318
 - activemq::wireformat::stomp::Stomp-
Frame, 2591
 - cms::CMSProperties, 832
 - decaf::util::Properties, 2203
- hasRemaining
 - decaf::nio::Buffer, 586
- hasUnconsumedMessages
 - activemq::core::ActiveMQSession-
Executor, 358
- hash_bits
 - internal_state, 1525
- hash_mask
 - internal_state, 1525
- hash_shift
 - internal_state, 1525
- hash_size
 - internal_state, 1525
- hashCode
 - decaf::security::auth::x500::X500-
Principal, 2914
- have
 - gz_state, 1400
 - inflate_state, 1447
- havedict
 - inflate_state, 1447
- hcrc
 - gz_header_s, 1399
- head
 - inflate_state, 1447
 - internal_state, 1525
- heap
 - internal_state, 1525
- heap_len
 - internal_state, 1525
- heap_max
 - internal_state, 1525
- high_water
 - internal_state, 1525
- highestOneBit
 - decaf::lang::Integer, 1505
 - decaf::lang::Long, 1731
- hold
 - inflate_state, 1447
- hostname
 - decaf::net::InetAddress, 1444
- how
 - gz_state, 1400
- impl
 - decaf::net::Socket, 2469
- implAccept
 - decaf::net::ServerSocket, 2349
- in
 - decaf::io::FileDescriptor, 1332
 - gz_state, 1400
- inProgressClearRequired
 - activemq::core::ActiveMQConsumer,
241
- inReceive
 - activemq::wireformat::openwire::-
OpenWireFormat, 2051
 - activemq::wireformat::stomp::Stomp-
WireFormat, 2599
 - activemq::wireformat::WireFormat,
2886
- increaseUsage
 - activemq::util::MemoryUsage, 1820
 - activemq::util::Usage, 2873
- incrementAndGet

- decaf::util::concurrent::atomic::-AtomicInteger, 480
- indexOf
 - decaf::util::AbstractList, 128
 - decaf::util::ArrayList, 453
 - decaf::util::concurrent::CopyOnWriteArrayList, 1039
 - decaf::util::LinkedList, 1647
 - decaf::util::List, 1663
 - decaf::util::StlList, 2547
- inffast.h
 - OF, 3148
- inflate
 - decaf::util::zip::Inflater, 1451, 1452
- inflate.h
 - BAD, 3149
 - CHECK, 3149
 - CODELENS, 3149
 - COMMENT, 3149
 - COPY, 3149
 - COPY_, 3149
 - DICT, 3149
 - DICTID, 3149
 - DIST, 3149
 - DISTEXT, 3149
 - DONE, 3149
 - EXLEN, 3149
 - EXTRA, 3149
 - FLAGS, 3148
 - GUNZIP, 3148
 - HCRC, 3149
 - HEAD, 3148
 - LEN, 3149
 - LENEXT, 3149
 - LENGTH, 3149
 - LENLENS, 3149
 - LEN_, 3149
 - LIT, 3149
 - MATCH, 3149
 - MEM, 3149
 - NAME, 3149
 - OS, 3149
 - STORED, 3149
 - SYNC, 3149
 - TABLE, 3149
 - TIME, 3149
 - TYPE, 3149
 - TYPEDO, 3149
 - inflate_mode, 3148
- inflate_mode
 - inflate.h, 3148
- inflate_state, 1445
 - back, 1446
 - bits, 1446
 - check, 1446
 - codes, 1446
 - distbits, 1446
 - distcode, 1446
 - dmax, 1446
 - extra, 1446
 - flags, 1446
 - have, 1447
 - havedict, 1447
 - head, 1447
 - hold, 1447
 - last, 1447
 - lenbits, 1447
 - lencode, 1447
 - length, 1447
 - lens, 1447
 - mode, 1447
 - ncode, 1447
 - ndist, 1447
 - next, 1447
 - nlen, 1447
 - offset, 1447
 - sane, 1447
 - total, 1447
 - was, 1447
 - wbits, 1447
 - whave, 1447
 - window, 1447
 - wnext, 1447
 - work, 1447
 - wrap, 1447
 - wsizes, 1448
- inflateBackInit
 - zlib.h, 3156
- inflateInit
 - zlib.h, 3156
- inflateInit2
 - zlib.h, 3157
- inflater
 - decaf::util::zip::InflaterInputStream, 1464
- info
 - decaf::util::logging::Logger, 1701
 - decaf::util::logging::SimpleLogger, 2443
- inftrees.h

- CODES, 3150
- DISTS, 3150
- ENOUGH, 3150
- ENOUGH_DISTS, 3150
- ENOUGH_LENS, 3150
- LENS, 3150
- OF, 3150
- codetype, 3150
- init
 - activemq::cmsutil::CmsAccessor, 822
 - activemq::cmsutil::CmsDestination-Accessor, 825
 - activemq::cmsutil::CmsTemplate, 842
 - activemq::cmsutil::Destination-Resolver, 1223
 - activemq::cmsutil::DynamicDestination-Resolver, 1273
- initCause
 - decaf::lang::Exception, 1284
 - decaf::lang::Throwable, 2735
- initSocketImpl
 - decaf::net::Socket, 2464
- initializeLibrary
 - activemq::library::ActiveMQCPP, 245
- initializeNetworking
 - decaf::internal::net::Network, 1971
- initializeRuntime
 - decaf::lang::Runtime, 2314
- inputStream
 - decaf::io::FilterInputStream, 1340
- ins_h
 - internal_state, 1525
- insert
 - decaf::internal::util::TimerTaskHeap, 2755
- intBitsToFloat
 - decaf::lang::Float, 1350
- intValue
 - activemq::util::PrimitiveValueNode::PrimitiveValue, 2143
 - decaf::lang::Byte, 618
 - decaf::lang::Character, 769
 - decaf::lang::Double, 1242
 - decaf::lang::Float, 1351
 - decaf::lang::Integer, 1506
 - decaf::lang::Long, 1732
 - decaf::lang::Number, 1993
 - decaf::lang::Short, 2403
 - decaf::util::concurrent::atomic::AtomicInteger, 480
 - decaf::util::logging::Level, 1618
 - internal_state, 1523
 - bi_buf, 1524
 - bi_valid, 1524
 - bl_count, 1524
 - bl_desc, 1524
 - bl_tree, 1524
 - block_start, 1524
 - d_buf, 1524
 - d_desc, 1525
 - depth, 1525
 - dummy, 1525
 - dyn_dtree, 1525
 - dyn_ltree, 1525
 - good_match, 1525
 - gzhead, 1525
 - gzindex, 1525
 - hash_bits, 1525
 - hash_mask, 1525
 - hash_shift, 1525
 - hash_size, 1525
 - head, 1525
 - heap, 1525
 - heap_len, 1525
 - heap_max, 1525
 - high_water, 1525
 - ins_h, 1525
 - l_buf, 1525
 - l_desc, 1525
 - last_eob_len, 1525
 - last_flush, 1525
 - last_lit, 1525
 - level, 1525
 - lit_bufsize, 1526
 - lookahead, 1526
 - match_available, 1526
 - match_length, 1526
 - match_start, 1526
 - matches, 1526
 - max_chain_length, 1526
 - max_lazy_match, 1526
 - method, 1526
 - nice_match, 1526
 - opt_len, 1526
 - pending, 1526
 - pending_buf, 1526
 - pending_buf_size, 1526
 - pending_out, 1526

- prev, 1526
- prev_length, 1526
- prev_match, 1526
- static_len, 1526
- status, 1526
- strategy, 1526
- strm, 1526
- strstart, 1526
- w_bits, 1526
- w_mask, 1527
- w_size, 1527
- window, 1527
- window_size, 1527
- wrap, 1527
- intf
 - zconf.h, 3153
- isAbsolute
 - decaf::internal::net::URIType, 2864
 - decaf::net::URI, 2836
- isAdvisory
 - activemq::commands::ActiveMQ-Destination, 252
- isAlive
 - decaf::lang::Thread, 2709
- isAlwaysSyncSend
 - activemq::core::ActiveMQConnection, 203
 - activemq::core::ActiveMQConnection-Factory, 221
- isAnyLocalAddress
 - decaf::net::Inet4Address, 1433
 - decaf::net::InetAddress, 1442
- isAutoAcknowledge
 - activemq::core::ActiveMQSession, 349
 - activemq::core::ActiveMQXASession, 438
- isBackup
 - activemq::transport::failover::Failover-Transport, 1310
- isBound
 - decaf::net::ServerSocket, 2350
 - decaf::net::Socket, 2464
- isBrokerInfo
 - activemq::commands::BaseCommand, 495
 - activemq::commands::BrokerInfo, 575
 - activemq::commands::Command, 867
- isBrokerMasterConnector
 - activemq::commands::Connection-Info, 971
- isBrowser
 - activemq::commands::Consumer-Info, 1014
- isCacheEnabled
 - activemq::commands::WireFormat-Info, 2894
 - activemq::wireformat::openwire::OpenWireFormat, 2051
- isCancelled
 - decaf::util::concurrent::Future, 1392
- isClientAcknowledge
 - activemq::core::ActiveMQSession, 349
- isClientMaster
 - activemq::commands::Connection-Info, 971
- isClose
 - activemq::commands::Connection-Control, 941
 - activemq::commands::Consumer-Control, 994
- isClosed
 - activemq::core::ActiveMQConnection, 203
 - activemq::core::ActiveMQConsumer, 241
 - activemq::core::ActiveMQProducer, 312
 - activemq::core::FifoMessageDispatch-Channel, 1325
 - activemq::core::MessageDispatch-Channel, 1888
 - activemq::core::SimplePriority-MessageDispatchChannel, 2447
 - activemq::transport::failover::Backup-Transport, 487
 - activemq::transport::failover::Failover-Transport, 1310
 - activemq::transport::IOTransport, 1551
 - activemq::transport::mock::Mock-Transport, 1952
 - activemq::transport::tcp::TcpTransport, 2696
 - activemq::transport::Transport, 2793

- activemq::transport::TransportFilter, 2804
- decaf::internal::net::tcp::TcpSocket, 2685
- decaf::io::FilterInputStream, 1338
- decaf::io::FilterOutputStream, 1344
- decaf::net::ServerSocket, 2350
- decaf::net::Socket, 2464
- isComposite
 - activemq::commands::ActiveMQ-Destination, 252
- isCompressed
 - activemq::commands::Message, 1832
- isConnected
 - activemq::transport::failover::Failover-Transport, 1311
 - activemq::transport::IOTransport, 1552
 - activemq::transport::mock::Mock-Transport, 1953
 - activemq::transport::tcp::TcpTransport, 2696
 - activemq::transport::Transport, 2793
 - activemq::transport::TransportFilter, 2804
 - decaf::internal::net::tcp::TcpSocket, 2685
 - decaf::net::Socket, 2464
- isConnectionAdvisory
 - activemq::commands::ActiveMQ-Destination, 253
- isConnectionControl
 - activemq::commands::BaseCommand, 495
 - activemq::commands::Command, 867
 - activemq::commands::Connection-Control, 941
- isConnectionInfo
 - activemq::commands::BaseCommand, 495
 - activemq::commands::Command, 867
 - activemq::commands::Connection-Info, 971
- isConnectionInterruptProcessingComplete
 - activemq::state::ConnectionState, 984
- isConsumerAdvisory
 - activemq::commands::ActiveMQ-Destination, 253
- isConsumerInfo
 - activemq::commands::BaseCommand, 495
 - activemq::commands::Command, 867
 - activemq::commands::Consumer-Info, 1014
- isDaemon
 - decaf::lang::Thread, 2709
- isDeletedByBroker
 - activemq::commands::ActiveMQ-BlobMessage, 160
- isDigit
 - decaf::lang::Character, 769
- isDispatchAsync
 - activemq::commands::Consumer-Info, 1014
 - activemq::commands::ProducerInfo, 2193
 - activemq::core::ActiveMQConnection, 203
 - activemq::core::ActiveMQConnection-Factory, 221
- isDone
 - decaf::util::concurrent::Future, 1392
 - decaf::util::zip::DeflaterOutput-Stream, 1194
- isDroppable
 - activemq::commands::Message, 1832
- isDuplexConnection
 - activemq::commands::BrokerInfo, 576
- isDupsOkAcknowledge
 - activemq::core::ActiveMQSession, 349
- isEmpty
 - activemq::commands::ActiveMQ-MapMessage, 278
 - activemq::core::ActiveMQSession-Executor, 358
 - activemq::core::FifoMessageDispatch-Channel, 1326
 - activemq::core::MessageDispatch-Channel, 1889
 - activemq::core::SimplePriority-MessageDispatchChannel, 2447

- activemq::util::ActiveMQProperties, 319
- cms::CMSProperties, 833
- cms::MapMessage, 1786
- decaf::internal::util::TimerTaskHeap, 2755
- decaf::lang::String, 2624
- decaf::util::AbstractCollection, 114
- decaf::util::ArrayList, 454
- decaf::util::Collection, 860
- decaf::util::concurrent::Concurrent-StlMap, 913
- decaf::util::concurrent::CopyOn-WriteArrayList, 1040
- decaf::util::concurrent::CopyOn-WriteArraySet, 1058
- decaf::util::concurrent::Synchronous-Queue, 2660
- decaf::util::LinkedList, 1648
- decaf::util::Map, 1774
- decaf::util::Properties, 2203
- decaf::util::StlList, 2548
- decaf::util::StlMap, 2559
- decaf::util::StlSet, 2580
- isEnabled
 - activemq::transport::failover::Backup-TransportPool, 490
- isExclusive
 - activemq::commands::ActiveMQ-Destination, 253
 - activemq::commands::Consumer-Info, 1014
- isExit
 - activemq::commands::Connection-Control, 941
- isExpired
 - activemq::commands::Message, 1832
- isExplicitQosEnabled
 - activemq::cmsutil::CmsTemplate, 843
- isFailOnClose
 - activemq::transport::mock::Mock-Transport, 1953
- isFailOnKeepAliveSends
 - activemq::transport::mock::Mock-Transport, 1953
- isFailOnReceiveMessage
 - activemq::transport::mock::Mock-Transport, 1953
- isFailOnSendMessage
 - activemq::transport::mock::Mock-Transport, 1953
- isFailOnStart
 - activemq::transport::mock::Mock-Transport, 1953
- isFailOnStop
 - activemq::transport::mock::Mock-Transport, 1953
- isFailoverReconnect
 - activemq::commands::Connection-Info, 971
- isFair
 - decaf::util::concurrent::locks:-ReentrantLock, 2258
 - decaf::util::concurrent::Semaphore, 2337
- isFaultTolerant
 - activemq::commands::Connection-Control, 941
 - activemq::commands::Connection-Info, 971
 - activemq::transport::failover::Failover-Transport, 1311
 - activemq::transport::IOTransport, 1552
 - activemq::transport::mock::Mock-Transport, 1953
 - activemq::transport::tcp::TcpTransport, 2696
 - activemq::transport::Transport, 2793
 - activemq::transport::TransportFilter, 2804
- isFaultTolerantConfiguration
 - activemq::commands::BrokerInfo, 576
- isFlush
 - activemq::commands::Consumer-Control, 994
- isFull
 - activemq::util::MemoryUsage, 1820
 - activemq::util::Usage, 2873
- isHeldByCurrentThread
 - decaf::util::concurrent::locks:-ReentrantLock, 2258
- isISOControl
 - decaf::lang::Character, 769
- isInLocalTransaction
 - activemq::core::ActiveMQTransaction-Context, 428

- isInTransaction
 - activemq::core::ActiveMQTransaction-Context, 428
- isInXATransaction
 - activemq::core::ActiveMQTransaction-Context, 428
- isIndividualAcknowledge
 - activemq::core::ActiveMQSession, 349
- isInfinite
 - decaf::lang::Double, 1242
 - decaf::lang::Float, 1351
- isInitialized
 - activemq::transport::failover::Failover-Transport, 1311
- isInputShutdown
 - decaf::net::Socket, 2464
- isKeepAliveInfo
 - activemq::commands::BaseCommand, 495
 - activemq::commands::Command, 868
 - activemq::commands::KeepAliveInfo, 1593
- isKeepAliveResponseRequired
 - activemq::transport::inactivity::InactivityMonitor, 1427
- isLetter
 - decaf::lang::Character, 769
- isLetterOrDigit
 - decaf::lang::Character, 769
- isLinkLocalAddress
 - decaf::net::Inet4Address, 1433
 - decaf::net::InetAddress, 1442
- isLocalTransactionId
 - activemq::commands::LocalTransaction-Id, 1677
 - activemq::commands::TransactionId, 2769
- isLocked
 - decaf::util::concurrent::Lock, 1683
 - decaf::util::concurrent::locks::ReentrantLock, 2259
- isLoggable
 - decaf::util::logging::Filter, 1333
 - decaf::util::logging::Handler, 1404
 - decaf::util::logging::Logger, 1702
 - decaf::util::logging::StreamHandler, 2605
- isLoopbackAddress
 - decaf::net::Inet4Address, 1433
 - decaf::net::InetAddress, 1442
- isLowerCase
 - decaf::lang::Character, 770
- isMCGlobal
 - decaf::net::Inet4Address, 1433
 - decaf::net::InetAddress, 1442
- isMCLinkLocal
 - decaf::net::Inet4Address, 1434
 - decaf::net::InetAddress, 1443
- isMCNodeLocal
 - decaf::net::Inet4Address, 1434
 - decaf::net::InetAddress, 1443
- isMCOrgLocal
 - decaf::net::Inet4Address, 1434
 - decaf::net::InetAddress, 1443
- isMCSiteLocal
 - decaf::net::Inet4Address, 1434
 - decaf::net::InetAddress, 1443
- isManageable
 - activemq::commands::Connection-Info, 972
- isMarshalAware
 - activemq::commands::ActiveMQ-MapMessage, 278
 - activemq::commands::BaseData-Structure, 530
 - activemq::commands::Message, 1832
 - activemq::commands::WireFormat-Info, 2895
 - activemq::wireformat::MarshalAware, 1795
- isMasterBroker
 - activemq::commands::BrokerInfo, 576
- isMessage
 - activemq::commands::BaseCommand, 495
 - activemq::commands::Command, 868
 - activemq::commands::Message, 1832
- isMessageAck
 - activemq::commands::BaseCommand, 496
 - activemq::commands::Command, 868
 - activemq::commands::MessageAck, 1871

- isMessageDispatch
 - activemq::commands::BaseCommand, 496
 - activemq::commands::Command, 868
 - activemq::commands::MessageDispatch, 1884
- isMessageDispatchNotification
 - activemq::commands::BaseCommand, 496
 - activemq::commands::Command, 868
 - activemq::commands::MessageDispatchNotification, 1897
- isMessageIdEnabled
 - activemq::cmsutil::CmsTemplate, 843
- isMessagePrioritySupported
 - activemq::core::ActiveMQConnection, 203
 - activemq::core::ActiveMQConnectionFactory, 222
- isMessageTimestampEnabled
 - activemq::cmsutil::CmsTemplate, 843
- isMulticastAddress
 - decaf::net::Inet4Address, 1435
 - decaf::net::InetAddress, 1443
- isNaN
 - decaf::lang::Double, 1242
 - decaf::lang::Float, 1351
- isNetworkConnection
 - activemq::commands::BrokerInfo, 576
- isNetworkSubscription
 - activemq::commands::ConsumerInfo, 1014
- isNoLocal
 - activemq::cmsutil::CmsTemplate, 843
 - activemq::commands::ConsumerInfo, 1014
- isNoRangeAcks
 - activemq::commands::ConsumerInfo, 1014
- isOpaque
 - decaf::internal::net::URIType, 2864
 - decaf::net::URI, 2837
- isOptimizedAcknowledge
 - activemq::commands::ConsumerInfo, 1014
- isOrdered
 - activemq::commands::ActiveMQDestination, 253
- isOutputShutdown
 - decaf::net::Socket, 2464
- isPending
 - activemq::threads::CompositeTask, 893
 - activemq::transport::failover::BackupTransportPool, 490
 - activemq::transport::failover::CloseTransportsTask, 818
 - activemq::transport::failover::FailoverTransport, 1311
- isPersistent
 - activemq::commands::Message, 1833
- isPrepared
 - activemq::state::TransactionState, 2786
- isProducerAck
 - activemq::commands::BaseCommand, 496
 - activemq::commands::Command, 868
 - activemq::commands::ProducerAck, 2173
- isProducerAdvisory
 - activemq::commands::ActiveMQDestination, 253
- isProducerInfo
 - activemq::commands::BaseCommand, 496
 - activemq::commands::Command, 868
 - activemq::commands::ProducerInfo, 2193
- isPubSubDomain
 - activemq::cmsutil::CmsDestinationAccessor, 825
- isQueue
 - activemq::commands::ActiveMQDestination, 253
- isRandomize
 - activemq::transport::failover::FailoverTransport, 1311
 - activemq::transport::failover::URIPool, 2853

- isReadOnly
 - decaf::internal::nio::ByteBuffer, 671
 - decaf::internal::nio::CharArrayBuffer, 782
 - decaf::internal::nio::DoubleArrayBuffer, 1257
 - decaf::internal::nio::FloatArrayBuffer, 1365
 - decaf::internal::nio::IntArrayBuffer, 1486
 - decaf::internal::nio::LongArrayBuffer, 1750
 - decaf::internal::nio::ShortArrayBuffer, 2417
 - decaf::nio::Buffer, 586
 - decaf::nio::ByteBuffer, 707
- isReadOnlyBody
 - activemq::commands::Message, 1833
- isReadOnlyProperties
 - activemq::commands::Message, 1833
- isRebalanceConnection
 - activemq::commands::ConnectionControl, 941
- isRecievedByDFBridge
 - activemq::commands::Message, 1833
- isReconnectSupported
 - activemq::transport::failover::FailoverTransport, 1312
 - activemq::transport::IOTransport, 1552
 - activemq::transport::mock::MockTransport, 1953
 - activemq::transport::Transport, 2794
 - activemq::transport::TransportFilter, 2805
- isRemoveInfo
 - activemq::commands::BaseCommand, 496
 - activemq::commands::Command, 869
 - activemq::commands::RemoveInfo, 2270
- isRemoveSubscriptionInfo
 - activemq::commands::BaseCommand, 496
- activemq::commands::Command, 869
- activemq::commands::RemoveSubscriptionInfo, 2278
- isResponse
 - activemq::commands::BaseCommand, 497
 - activemq::commands::Command, 869
 - activemq::commands::Response, 2300
- isResponseRequired
 - activemq::commands::BaseCommand, 497
 - activemq::commands::Command, 869
- isRestoreConsumers
 - activemq::state::ConnectionStateTracker, 986
- isRestoreProducers
 - activemq::state::ConnectionStateTracker, 986
- isRestoreSessions
 - activemq::state::ConnectionStateTracker, 986
- isRestoreTransaction
 - activemq::state::ConnectionStateTracker, 986
- isResume
 - activemq::commands::ConnectionControl, 941
- isRetroactive
 - activemq::commands::ConsumerInfo, 1014
- isRunning
 - activemq::core::ActiveMQSessionExecutor, 358
 - activemq::core::FifoMessageDispatchChannel, 1326
 - activemq::core::MessageDispatchChannel, 1889
 - activemq::core::SimplePriorityMessageDispatchChannel, 2447
- isSameRM
 - activemq::core::ActiveMQTransactionContext, 428
 - cms::XAResource, 2930
- isScheduled
 - decaf::util::TimerTask, 2752

- isServerAuthority
 - decaf::internal::net::URIType, 2864
- isShutdown
 - decaf::util::concurrent::Executor-Service, 1304
 - decaf::util::concurrent::ThreadPool-Executor, 2727
- isShutdownInfo
 - activemq::commands::BaseCommand, 497
 - activemq::commands::Command, 869
 - activemq::commands::ShutdownInfo, 2433
- isSiteLocalAddress
 - decaf::net::Inet4Address, 1435
 - decaf::net::InetAddress, 1444
- isSizePrefixDisabled
 - activemq::commands::WireFormat-Info, 2895
 - activemq::wireformat::openwire::OpenWireFormat, 2052
- isSlaveBroker
 - activemq::commands::BrokerInfo, 576
- isStackTraceEnabled
 - activemq::commands::WireFormat-Info, 2895
 - activemq::wireformat::openwire::OpenWireFormat, 2052
- isStart
 - activemq::commands::Consumer-Control, 994
- isStarted
 - activemq::core::ActiveMQConnection, 203
 - activemq::core::ActiveMQSession, 349
 - activemq::util::ServiceSupport, 2360
- isStop
 - activemq::commands::Consumer-Control, 995
- isStopped
 - activemq::util::ServiceSupport, 2361
- isStopping
 - activemq::util::ServiceSupport, 2361
- isSuspend
 - activemq::commands::Connection-Control, 941
- isSynchronizationRegistered
 - activemq::core::ActiveMQConsumer, 241
- isTcpNoDelayEnabled
 - activemq::commands::WireFormat-Info, 2895
 - activemq::wireformat::openwire::OpenWireFormat, 2052
- isTemporary
 - activemq::commands::ActiveMQ-Destination, 254
- isTerminated
 - decaf::util::concurrent::Executor-Service, 1304
 - decaf::util::concurrent::ThreadPool-Executor, 2727
- isTerminating
 - decaf::util::concurrent::ThreadPool-Executor, 2727
- isTightEncodingEnabled
 - activemq::commands::WireFormat-Info, 2896
 - activemq::wireformat::openwire::OpenWireFormat, 2052
- isTopic
 - activemq::commands::ActiveMQ-Destination, 254
- isTrackMessages
 - activemq::state::ConnectionState-Tracker, 986
 - activemq::transport::failover::Failover-Transport, 1312
- isTrackTransactionProducers
 - activemq::state::ConnectionState-Tracker, 986
 - activemq::transport::failover::Failover-Transport, 1312
- isTrackTransactions
 - activemq::state::ConnectionState-Tracker, 986
- isTransacted
 - activemq::cmsutil::PooledSession, 2105
 - activemq::core::ActiveMQSession, 349
 - activemq::core::ActiveMQXASession, 438
 - cms::Session, 2375
- isTransactionInfo
 - activemq::commands::BaseCommand, 497

- activemq::commands::Command, 869
- activemq::commands::Transaction-Info, 2776
- isTransportFailed
 - activemq::core::ActiveMQConnection, 203
- isUpdateURIsSupported
 - activemq::transport::failover::Failover-Transport, 1312
 - activemq::transport::IOTransport, 1552
 - activemq::transport::mock::Mock-Transport, 1954
 - activemq::transport::Transport, 2794
 - activemq::transport::TransportFilter, 2805
- isUpperCase
 - decaf::lang::Character, 770
- isUseAsyncSend
 - activemq::core::ActiveMQConnection, 204
 - activemq::core::ActiveMQConnection-Factory, 222
- isUseCollisionAvoidance
 - activemq::core::policies::Default-RedeliveryPolicy, 1153
 - activemq::core::RedeliveryPolicy, 2254
- isUseCompression
 - activemq::core::ActiveMQConnection, 204
 - activemq::core::ActiveMQConnection-Factory, 222
- isUseExponentialBackOff
 - activemq::core::policies::Default-RedeliveryPolicy, 1153
 - activemq::core::RedeliveryPolicy, 2254
 - activemq::transport::failover::Failover-Transport, 1312
- isValid
 - activemq::commands::WireFormat-Info, 2896
 - decaf::internal::net::URIType, 2865
- isValidDomainName
 - decaf::internal::net::URIHelper, 2846
- isValidHexChar
 - decaf::internal::net::URIHelper, 2846
- isValidHost
 - decaf::internal::net::URIHelper, 2846
- isValidIP4Word
 - decaf::internal::net::URIHelper, 2847
- isValidIP6Address
 - decaf::internal::net::URIHelper, 2847
- isValidIPv4Address
 - decaf::internal::net::URIHelper, 2847
- isWaitingForResponse
 - activemq::state::Tracked, 2766
- isWhitespace
 - decaf::lang::Character, 770
- isWildcard
 - activemq::commands::ActiveMQ-Destination, 254
- isWireFormatInfo
 - activemq::commands::BaseCommand, 497
 - activemq::commands::Command, 869
 - activemq::commands::WireFormat-Info, 2896
- isXATransactionId
 - activemq::commands::TransactionId, 2769
 - activemq::commands::XATransaction-Id, 2941
- itemExists
 - activemq::commands::ActiveMQ-MapMessage, 278
 - cms::MapMessage, 1786
- iterate
 - activemq::core::ActiveMQConsumer, 241
 - activemq::core::ActiveMQSession-Executor, 358
 - activemq::threads::CompositeTask-Runner, 895
 - activemq::threads::Task, 2676
 - activemq::transport::failover::Backup-TransportPool, 491
 - activemq::transport::failover::Close-TransportsTask, 818
 - activemq::transport::failover::Failover-Transport, 1312
- iterator
 - decaf::lang::Iterable, 1557, 1558
 - decaf::util::AbstractList, 128, 129
 - decaf::util::AbstractSequentialList, 148, 149

- decaf::util::concurrent::CopyOn-WriteArrayList, 1040
- decaf::util::concurrent::CopyOn-WriteArraySet, 1058
- decaf::util::concurrent::LinkedBlockingQueue, 1626
- decaf::util::concurrent::SynchronousQueue, 2661
- decaf::util::PriorityQueue, 2167
- decaf::util::StlList, 2548
- decaf::util::StlQueue, 2569
- decaf::util::StlSet, 2580
- join
 - decaf::lang::Thread, 2709, 2710
- keySet
 - decaf::util::concurrent::ConcurrentStlMap, 913
 - decaf::util::Map, 1775
 - decaf::util::StlMap, 2560
- l_buf
 - internal_state, 1525
- l_desc
 - internal_state, 1525
- last
 - inflate_state, 1447
- last_eob_len
 - internal_state, 1525
- last_flush
 - internal_state, 1525
- last_lit
 - internal_state, 1525
- lastDeliveredSequenceId
 - activemq::commands::RemoveInfo, 2271
 - activemq::core::ActiveMQSession, 354
- lastIndexOf
 - decaf::util::AbstractList, 129
 - decaf::util::ArrayList, 454
 - decaf::util::concurrent::CopyOn-WriteArrayList, 1040, 1041
 - decaf::util::LinkedList, 1648
 - decaf::util::List, 1664
 - decaf::util::StlList, 2548
- lastMessageId
 - activemq::commands::MessageAck, 1872
- lastNakNumber
 - activemq::commands::Replay-Command, 2287
- len
 - ct_data_s, 1068
- lenbits
 - inflate_state, 1447
- lencode
 - inflate_state, 1447
- length
 - decaf::internal::nio::CharArrayBuffer, 785
 - decaf::lang::ArrayPointer, 467
 - decaf::lang::CharSequence, 804
 - decaf::lang::String, 2624
 - decaf::nio::CharBuffer, 795
 - decaf::util::zip::InflaterInputStream, 1464
 - inflate_state, 1447
- lens
 - inflate_state, 1447
- level
 - gz_state, 1400
 - internal_state, 1525
- limit
 - decaf::nio::Buffer, 586
- listIterator
 - decaf::util::AbstractList, 130–132
 - decaf::util::AbstractSequentialList, 149, 150
 - decaf::util::concurrent::CopyOn-WriteArrayList, 1041, 1042
 - decaf::util::LinkedList, 1649
 - decaf::util::List, 1665–1667
 - decaf::util::StlList, 2549
- listValue
 - activemq::util::PrimitiveValueNode:-PrimitiveValue, 2143
- listen
 - decaf::internal::net::tcp::TcpSocket, 2685
 - decaf::net::SocketImpl, 2485
- listener
 - activemq::transport::TransportFilter, 2810
- lit_bufsize
 - internal_state, 1526
- load
 - decaf::util::Properties, 2204
- local

- gzguts.h, 3147
- localPort
 - decaf::net::SocketImpl, 2487
- lock
 - activemq::core::FifoMessageDispatchChannel, 1326
 - activemq::core::SimplePriorityMessageDispatchChannel, 2447
 - decaf::internal::util::concurrent::MutexImpl, 1967
 - decaf::internal::util::concurrent::SynchronizableImpl, 2651
 - decaf::io::InputStream, 1467
 - decaf::io::OutputStream, 2069
 - decaf::util::AbstractCollection, 115
 - decaf::util::concurrent::ConcurrentStlMap, 913
 - decaf::util::concurrent::CopyOnWriteArrayList, 1042
 - decaf::util::concurrent::Lock, 1683
 - decaf::util::concurrent::locks::Lock, 1686
 - decaf::util::concurrent::locks::ReentrantLock, 2259
 - decaf::util::concurrent::Mutex, 1962
 - decaf::util::concurrent::Synchronizable, 2640
 - decaf::util::StlMap, 2560
 - decaf::util::StlQueue, 2569
- lock_count
 - decaf::util::concurrent::MutexHandle, 1966
- lock_owner
 - decaf::util::concurrent::MutexHandle, 1966
- lockInterruptibly
 - decaf::util::concurrent::locks::Lock, 1686
 - decaf::util::concurrent::locks::ReentrantLock, 2260
- log
 - decaf::util::logging::Logger, 1702, 1703
 - decaf::util::logging::LogWriter, 1725
 - decaf::util::logging::SimpleLogger, 2443
- longBitsToDouble
 - decaf::lang::Double, 1243
- longValue
 - activemq::util::PrimitiveValueNode::PrimitiveValue, 2143
- decaf::lang::Byte, 618
- decaf::lang::Character, 770
- decaf::lang::Double, 1243
- decaf::lang::Float, 1352
- decaf::lang::Integer, 1506
- decaf::lang::Long, 1732
- decaf::lang::Number, 1994
- decaf::lang::Short, 2403
- decaf::util::concurrent::atomic::AtomicInteger, 480
- lookahead
 - internal_state, 1526
- loopbackBytes
 - decaf::net::InetAddress, 1444
- looseMarshal
 - activemq::wireformat::openwire::marshal::BaseDataStreamMarshaller, 511
 - activemq::wireformat::openwire::marshal::DataStreamMarshaller, 1123
 - activemq::wireformat::openwire::marshal::generated::ActiveMQBlobMessageMarshaller, 163
 - activemq::wireformat::openwire::marshal::generated::ActiveMQBytesMessageMarshaller, 185
 - activemq::wireformat::openwire::marshal::generated::ActiveMQDestinationMarshaller, 259
 - activemq::wireformat::openwire::marshal::generated::ActiveMQMapMessageMarshaller, 285
 - activemq::wireformat::openwire::marshal::generated::ActiveMQMessageMarshaller, 292
 - activemq::wireformat::openwire::marshal::generated::ActiveMQObjectMessageMarshaller, 305
 - activemq::wireformat::openwire::marshal::generated::ActiveMQQueueMarshaller, 330
 - activemq::wireformat::openwire::marshal::generated::Active-

- MQStreamMessageMarshaller, 376
- activemq::wireformat::openwire-
::marshal::generated::Active-
MQTempDestinationMarshaller, 383
- activemq::wireformat::openwire-
::marshal::generated::Active-
MQTempQueueMarshaller, 393
- activemq::wireformat::openwire-
::marshal::generated::Active-
MQTempTopicMarshaller, 402
- activemq::wireformat::openwire-
::marshal::generated::ActiveM-
QTextMessageMarshaller, 412
- activemq::wireformat::openwire-
::marshal::generated::Active-
MQTopicMarshaller, 420
- activemq::wireformat::openwire-
::marshal::generated::Base-
CommandMarshaller, 500
- activemq::wireformat::openwire-
::marshal::generated::Broker-
IdMarshaller, 569
- activemq::wireformat::openwire-
::marshal::generated::Broker-
InfoMarshaller, 580
- activemq::wireformat::openwire-
::marshal::generated::Connection-
ControlMarshaller, 945
- activemq::wireformat::openwire-
::marshal::generated::Connection-
ErrorMarshaller, 953
- activemq::wireformat::openwire-
::marshal::generated::Connection-
IdMarshaller, 965
- activemq::wireformat::openwire-
::marshal::generated::Connection-
InfoMarshaller, 975
- activemq::wireformat::openwire-
::marshal::generated::Consumer-
ControlMarshaller, 998
- activemq::wireformat::openwire-
::marshal::generated::Consumer-
IdMarshaller, 1006
- activemq::wireformat::openwire-
::marshal::generated::Consumer-
InfoMarshaller, 1019
- activemq::wireformat::openwire-
::marshal::generated::Control-
CommandMarshaller, 1027
- activemq::wireformat::openwire-
::marshal::generated::Data-
ArrayResponseMarshaller, 1073
- activemq::wireformat::openwire-
::marshal::generated::Data-
ResponseMarshaller, 1117
- activemq::wireformat::openwire-
::marshal::generated::Destination-
InfoMarshaller, 1219
- activemq::wireformat::openwire-
::marshal::generated::Discovery-
EventMarshaller, 1231
- activemq::wireformat::openwire-
::marshal::generated::Exception-
ResponseMarshaller, 1292
- activemq::wireformat::openwire-
::marshal::generated::Flush-
CommandMarshaller, 1385
- activemq::wireformat::openwire-
::marshal::generated::Integer-
ResponseMarshaller, 1520
- activemq::wireformat::openwire-
::marshal::generated::Journal-
QueueAckMarshaller, 1565
- activemq::wireformat::openwire-
::marshal::generated::Journal-
TopicAckMarshaller, 1574
- activemq::wireformat::openwire-
::marshal::generated::Journal-
TraceMarshaller, 1581
- activemq::wireformat::openwire-
::marshal::generated::Journal-
TransactionMarshaller, 1589
- activemq::wireformat::openwire-
::marshal::generated::Keep-
AliveInfoMarshaller, 1596
- activemq::wireformat::openwire-
::marshal::generated::Last-
PartialCommandMarshaller, 1610
- activemq::wireformat::openwire-
::marshal::generated::Local-
TransactionIdMarshaller, 1680
- activemq::wireformat::openwire-
::marshal::generated::Message-
AckMarshaller, 1874
- activemq::wireformat::openwire-
::marshal::generated::Message-

- DispatchMarshaller, 1892
- activemq::wireformat::openwire-
::marshal::generated::Message-
DispatchNotificationMarshaller,
1901
- activemq::wireformat::openwire-
::marshal::generated::Message-
IdMarshaller, 1914
- activemq::wireformat::openwire-
::marshal::generated::Message-
Marshaller, 1918
- activemq::wireformat::openwire-
::marshal::generated::Message-
PullMarshaller, 1945
- activemq::wireformat::openwire-
::marshal::generated::Network-
BridgeFilterMarshaller, 1976
- activemq::wireformat::openwire-
::marshal::generated::Partial-
CommandMarshaller, 2081
- activemq::wireformat::openwire-
::marshal::generated::Producer-
AckMarshaller, 2176
- activemq::wireformat::openwire-
::marshal::generated::Producer-
IdMarshaller, 2188
- activemq::wireformat::openwire-
::marshal::generated::Producer-
InfoMarshaller, 2196
- activemq::wireformat::openwire-
::marshal::generated::Remove-
InfoMarshaller, 2273
- activemq::wireformat::openwire-
::marshal::generated::Remove-
SubscriptionInfoMarshaller,
2281
- activemq::wireformat::openwire-
::marshal::generated::Replay-
CommandMarshaller, 2289
- activemq::wireformat::openwire-
::marshal::generated::Response-
Marshaller, 2309
- activemq::wireformat::openwire-
::marshal::generated::Session-
IdMarshaller, 2384
- activemq::wireformat::openwire-
::marshal::generated::Session-
InfoMarshaller, 2391
- activemq::wireformat::openwire-
::marshal::generated::Shutdown-
InfoMarshaller, 2436
- activemq::wireformat::openwire-
::marshal::generated::Subscription-
InfoMarshaller, 2637
- activemq::wireformat::openwire-
::marshal::generated::Transaction-
IdMarshaller, 2771
- activemq::wireformat::openwire-
::marshal::generated::Transaction-
InfoMarshaller, 2780
- activemq::wireformat::openwire-
::marshal::generated::Wire-
FormatInfoMarshaller, 2901
- activemq::wireformat::openwire-
::marshal::generated::XA-
TransactionIdMarshaller, 2944
- looseMarshalBrokerError
- activemq::wireformat::openwire-
::marshal::BaseDataStream-
Marshaller, 511
- looseMarshalCachedObject
- activemq::wireformat::openwire-
::marshal::BaseDataStream-
Marshaller, 511
- looseMarshalLong
- activemq::wireformat::openwire-
::marshal::BaseDataStream-
Marshaller, 512
- looseMarshalNestedObject
- activemq::wireformat::openwire-
::marshal::BaseDataStream-
Marshaller, 512
- activemq::wireformat::openwire::
OpenWireFormat, 2052
- looseMarshalObjectArray
- activemq::wireformat::openwire-
::marshal::BaseDataStream-
Marshaller, 513
- looseMarshalString
- activemq::wireformat::openwire-
::marshal::BaseDataStream-
Marshaller, 513
- looseUnmarshal
- activemq::wireformat::openwire-
::marshal::BaseDataStream-
Marshaller, 513
- activemq::wireformat::openwire-
::marshal::DataStreamMarshaller,
1125
- activemq::wireformat::openwire-

- `::marshal::generated::ActiveMQBlobMessageMarshaller`, 164
- `activemq::wireformat::openwire::marshal::generated::ActiveMQBytesMessageMarshaller`, 185
- `activemq::wireformat::openwire::marshal::generated::ActiveMQDestinationMarshaller`, 259
- `activemq::wireformat::openwire::marshal::generated::ActiveMQMapMessageMarshaller`, 286
- `activemq::wireformat::openwire::marshal::generated::ActiveMQMessageMarshaller`, 293
- `activemq::wireformat::openwire::marshal::generated::ActiveMQObjectMessageMarshaller`, 306
- `activemq::wireformat::openwire::marshal::generated::ActiveMQQueueMarshaller`, 331
- `activemq::wireformat::openwire::marshal::generated::ActiveMQStreamMessageMarshaller`, 377
- `activemq::wireformat::openwire::marshal::generated::ActiveMQTempDestinationMarshaller`, 384
- `activemq::wireformat::openwire::marshal::generated::ActiveMQTempQueueMarshaller`, 394
- `activemq::wireformat::openwire::marshal::generated::ActiveMQTempTopicMarshaller`, 403
- `activemq::wireformat::openwire::marshal::generated::ActiveMQTextMessageMarshaller`, 412
- `activemq::wireformat::openwire::marshal::generated::ActiveMQTopicMarshaller`, 421
- `activemq::wireformat::openwire::marshal::generated::BaseCommandMarshaller`, 501
- `activemq::wireformat::openwire::marshal::generated::BrokerIdMarshaller`, 570
- `activemq::wireformat::openwire::marshal::generated::BrokerInfoMarshaller`, 580
- `activemq::wireformat::openwire::marshal::generated::ConnectionControlMarshaller`, 946
- `activemq::wireformat::openwire::marshal::generated::ConnectionErrorMarshaller`, 953
- `activemq::wireformat::openwire::marshal::generated::ConnectionIdMarshaller`, 966
- `activemq::wireformat::openwire::marshal::generated::ConnectionInfoMarshaller`, 976
- `activemq::wireformat::openwire::marshal::generated::ConsumerControlMarshaller`, 998
- `activemq::wireformat::openwire::marshal::generated::ConsumerIdMarshaller`, 1007
- `activemq::wireformat::openwire::marshal::generated::ConsumerInfoMarshaller`, 1020
- `activemq::wireformat::openwire::marshal::generated::ControlCommandMarshaller`, 1028
- `activemq::wireformat::openwire::marshal::generated::DataArrayResponseMarshaller`, 1074
- `activemq::wireformat::openwire::marshal::generated::DataResponseMarshaller`, 1117
- `activemq::wireformat::openwire::marshal::generated::DestinationInfoMarshaller`, 1220
- `activemq::wireformat::openwire::marshal::generated::DiscoveryEventMarshaller`, 1232
- `activemq::wireformat::openwire::marshal::generated::ExceptionResponseMarshaller`, 1292
- `activemq::wireformat::openwire::marshal::generated::FlushCommandMarshaller`, 1385
- `activemq::wireformat::openwire::marshal::generated::IntegerResponseMarshaller`, 1521
- `activemq::wireformat::openwire-`

- `::marshal::generated::Journal-QueueAckMarshaller`, 1566
- `activemq::wireformat::openwire-
::marshal::generated::Journal-TopicAckMarshaller`, 1575
- `activemq::wireformat::openwire-
::marshal::generated::Journal-TraceMarshaller`, 1581
- `activemq::wireformat::openwire-
::marshal::generated::Journal-TransactionMarshaller`, 1589
- `activemq::wireformat::openwire-
::marshal::generated::Keep-AliveInfoMarshaller`, 1596
- `activemq::wireformat::openwire-
::marshal::generated::Last-PartialCommandMarshaller`, 1610
- `activemq::wireformat::openwire-
::marshal::generated::Local-TransactionIdMarshaller`, 1680
- `activemq::wireformat::openwire-
::marshal::generated::Message-AckMarshaller`, 1875
- `activemq::wireformat::openwire-
::marshal::generated::Message-DispatchMarshaller`, 1892
- `activemq::wireformat::openwire-
::marshal::generated::Message-DispatchNotificationMarshaller`, 1901
- `activemq::wireformat::openwire-
::marshal::generated::Message-IdMarshaller`, 1914
- `activemq::wireformat::openwire-
::marshal::generated::Message-Marshaller`, 1919
- `activemq::wireformat::openwire-
::marshal::generated::Message-PullMarshaller`, 1946
- `activemq::wireformat::openwire-
::marshal::generated::Network-BridgeFilterMarshaller`, 1976
- `activemq::wireformat::openwire-
::marshal::generated::Partial-CommandMarshaller`, 2081
- `activemq::wireformat::openwire-
::marshal::generated::Producer-AckMarshaller`, 2177
- `activemq::wireformat::openwire-
::marshal::generated::Producer-IdMarshaller`, 2188
- `activemq::wireformat::openwire-
::marshal::generated::Producer-InfoMarshaller`, 2197
- `activemq::wireformat::openwire-
::marshal::generated::Remove-InfoMarshaller`, 2273
- `activemq::wireformat::openwire-
::marshal::generated::Remove-SubscriptionInfoMarshaller`, 2282
- `activemq::wireformat::openwire-
::marshal::generated::Replay-CommandMarshaller`, 2289
- `activemq::wireformat::openwire-
::marshal::generated::Response-Marshaller`, 2309
- `activemq::wireformat::openwire-
::marshal::generated::Session-IdMarshaller`, 2384
- `activemq::wireformat::openwire-
::marshal::generated::Session-InfoMarshaller`, 2392
- `activemq::wireformat::openwire-
::marshal::generated::Shutdown-InfoMarshaller`, 2436
- `activemq::wireformat::openwire-
::marshal::generated::Subscription-InfoMarshaller`, 2637
- `activemq::wireformat::openwire-
::marshal::generated::Transaction-IdMarshaller`, 2772
- `activemq::wireformat::openwire-
::marshal::generated::Transaction-InfoMarshaller`, 2780
- `activemq::wireformat::openwire-
::marshal::generated::Wire-FormatInfoMarshaller`, 2902
- `activemq::wireformat::openwire-
::marshal::generated::XA-TransactionIdMarshaller`, 2944
- `looseUnmarshalBrokerError`
- `activemq::wireformat::openwire-
::marshal::BaseDataStream-Marshaller`, 514
- `looseUnmarshalByteArray`
- `activemq::wireformat::openwire-
::marshal::BaseDataStream-Marshaller`, 514

- looseUnmarshalCachedObject
 - activemq::wireformat::openwire-
::marshal::BaseDataStream-
Marshaller, 515
- looseUnmarshalConstByteArray
 - activemq::wireformat::openwire-
::marshal::BaseDataStream-
Marshaller, 515
- looseUnmarshalLong
 - activemq::wireformat::openwire-
::marshal::BaseDataStream-
Marshaller, 516
- looseUnmarshalNestedObject
 - activemq::wireformat::openwire-
::marshal::BaseDataStream-
Marshaller, 516
 - activemq::wireformat::openwire::-
OpenWireFormat, 2053
- looseUnmarshalString
 - activemq::wireformat::openwire-
::marshal::BaseDataStream-
Marshaller, 517
- lowestOneBit
 - decaf::lang::Integer, 1506
 - decaf::lang::Long, 1732
- manageable
 - activemq::commands::Connection-
Info, 973
- mapValue
 - activemq::util::PrimitiveValueNode::-
PrimitiveValue, 2143
- mark
 - decaf::io::BufferedInputStream, 593
 - decaf::io::ByteArrayInputStream, 684
 - decaf::io::FilterInputStream, 1338
 - decaf::io::InputStream, 1468
 - decaf::io::PushbackInputStream,
2218
 - decaf::io::Reader, 2239
 - decaf::nio::Buffer, 587
 - decaf::util::logging::SimpleLogger,
2443
 - decaf::util::zip::InflaterInputStream,
1461
- markSupported
 - decaf::io::BufferedInputStream, 593
 - decaf::io::ByteArrayInputStream, 684
 - decaf::io::FilterInputStream, 1338
 - decaf::io::InputStream, 1468
- decaf::io::PushbackInputStream,
2218
- decaf::io::Reader, 2240
- decaf::util::zip::InflaterInputStream,
1462
- marshal
 - activemq::wireformat::openwire-
::marshal::PrimitiveTypes-
Marshaller, 2137
 - activemq::wireformat::openwire::-
OpenWireFormat, 2053
 - activemq::wireformat::openwire-
::utils::BooleanStream, 554
 - activemq::wireformat::stomp::Stomp-
WireFormat, 2599
 - activemq::wireformat::WireFormat,
2887
- marshalList
 - activemq::wireformat::openwire-
::marshal::PrimitiveTypes-
Marshaller, 2138
- marshalMap
 - activemq::wireformat::openwire-
::marshal::PrimitiveTypes-
Marshaller, 2138
- marshalPrimitive
 - activemq::wireformat::openwire-
::marshal::PrimitiveTypes-
Marshaller, 2138
- marshalPrimitiveList
 - activemq::wireformat::openwire-
::marshal::PrimitiveTypes-
Marshaller, 2139
- marshalPrimitiveMap
 - activemq::wireformat::openwire-
::marshal::PrimitiveTypes-
Marshaller, 2139
- marshalledProperties
 - activemq::commands::Message,
1838
- marshalledSize
 - activemq::wireformat::openwire-
::utils::BooleanStream, 554
- masterBroker
 - activemq::commands::BrokerInfo,
578
- match_available
 - internal_state, 1526
- match_length
 - internal_state, 1526

- match_start
 - internal_state, 1526
- matches
 - internal_state, 1526
- max
 - decaf::lang::Math, 1806, 1807
- max_chain_length
 - internal_state, 1526
- max_code
 - tree_desc_s, 2815
- max_insert_length
 - deflate.h, 3145
- max_lazy_match
 - internal_state, 1526
- maximumPendingMessageLimit
 - activemq::commands::Consumer-Info, 1017
- message
 - activemq::cmsutil::CmsTemplate::ReceiveExecutor, 2250
 - activemq::commands::JournalTrace, 1579
 - activemq::commands::MessageDispatch, 1885
 - decaf::lang::Exception, 1286
- messageAck
 - activemq::commands::JournalQueueAck, 1564
- messageCount
 - activemq::commands::MessageAck, 1872
- messageId
 - activemq::commands::JournalTopicAck, 1572
 - activemq::commands::Message, 1838
 - activemq::commands::MessageDispatchNotification, 1899
 - activemq::commands::MessagePull, 1943
- messageSequenceId
 - activemq::commands::JournalTopicAck, 1572
- method
 - internal_state, 1526
- min
 - decaf::lang::Math, 1807–1810
- modCount
 - decaf::util::AbstractList, 134
- mode
 - gz_state, 1401
 - inflate_state, 1447
- modifiedUtf8ToAscii
 - activemq::util::MarshallingSupport, 1798
- msg
 - gz_state, 1401
 - z_stream_s, 2952
- mutex
 - decaf::util::AbstractCollection, 121
 - decaf::util::concurrent::ConditionHandle, 928
 - decaf::util::concurrent::MutexHandle, 1966
- name
 - gz_header_s, 1399
- name_max
 - gz_header_s, 1399
- nameUUIDFromBytes
 - decaf::util::UUID, 2881
- nanoTime
 - decaf::lang::System, 2674
- narrow
 - activemq::transport::failover::FailoverTransport, 1312
 - activemq::transport::IOTransport, 1552
 - activemq::transport::mock::MockTransport, 1954
 - activemq::transport::Transport, 2794
 - activemq::transport::TransportFilter, 2805
- ncode
 - inflate_state, 1447
- ndist
 - inflate_state, 1447
- needsDictionary
 - decaf::util::zip::Inflater, 1453
- needsInput
 - decaf::util::zip::Deflater, 1185
 - decaf::util::zip::Inflater, 1453
- networkBrokerId
 - activemq::commands::NetworkBridgeFilter, 1974
- networkConnection
 - activemq::commands::BrokerInfo, 578
- networkConsumerPath

- activemq::commands::Consumer-Info, 1017
- networkProperties
 - activemq::commands::BrokerInfo, 578
- networkSubscription
 - activemq::commands::Consumer-Info, 1017
- networkTTL
 - activemq::commands::Network-BridgeFilter, 1974
- newCondition
 - decaf::util::concurrent::locks::Lock, 1687
 - decaf::util::concurrent::locks::ReentrantLock, 2260
- newFixedThreadPool
 - decaf::util::concurrent::Executors, 1300, 1301
- newThread
 - decaf::util::concurrent::Thread-Factory, 2714
- next
 - activemq::transport::TransportFilter, 2810
 - decaf::security::SecureRandom, 2323
 - decaf::util::concurrent::CopyOn-WriteArrayList::ArrayListIterator, 460
 - decaf::util::Iterator, 1560
 - decaf::util::Random, 2231
 - gz_state, 1401
 - inflate_state, 1447
- next_in
 - z_stream_s, 2952
- next_out
 - z_stream_s, 2952
- nextBoolean
 - decaf::util::Random, 2232
- nextBytes
 - decaf::security::SecureRandom, 2323, 2324
 - decaf::util::Random, 2232
- nextDouble
 - decaf::util::Random, 2233
- nextFloat
 - decaf::util::Random, 2233
- nextGaussian
 - decaf::util::Random, 2233
- nextIndex
 - decaf::util::concurrent::CopyOn-WriteArrayList::ArrayListIterator, 460
 - decaf::util::ListIterator, 1672
- nextInt
 - decaf::util::Random, 2233, 2234
- nextLong
 - decaf::util::Random, 2234
- nextMessage
 - activemq::core::ActiveMQQueue-Browser, 328
 - cms::MessageEnumeration, 1904
- nextToken
 - decaf::util::StringTokenizer, 2629
- nice_match
 - internal_state, 1526
- nlen
 - inflate_state, 1447
- noLocal
 - activemq::cmsutil::CmsTemplate:-ReceiveExecutor, 2250
 - activemq::commands::Consumer-Info, 1017
- noRangeAcks
 - activemq::commands::Consumer-Info, 1017
- node
 - decaf::util::UUID, 2881
- normalize
 - decaf::net::URI, 2837
- notify
 - activemq::core::FifoMessageDispatch-Channel, 1326
 - activemq::core::SimplePriority-MessageDispatchChannel, 2448
 - decaf::internal::util::concurrent:-ConditionImpl, 930
 - decaf::internal::util::concurrent:-SynchronizableImpl, 2651
 - decaf::io::InputStream, 1468
 - decaf::io::OutputStream, 2069
 - decaf::util::AbstractCollection, 115
 - decaf::util::concurrent::Concurrent-StlMap, 914
 - decaf::util::concurrent::CopyOn-WriteArrayList, 1042
 - decaf::util::concurrent::Mutex, 1962

- decaf::util::concurrent::Synchronizable, 2641
- decaf::util::StlMap, 2560
- decaf::util::StlQueue, 2570
- notifyAll
 - activemq::core::FifoMessageDispatchChannel, 1327
 - activemq::core::SimplePriorityMessageDispatchChannel, 2448
- decaf::internal::util::concurrent::ConditionImpl, 930
- decaf::internal::util::concurrent::SynchronizableImpl, 2651
- decaf::io::InputStream, 1469
- decaf::io::OutputStream, 2070
- decaf::util::AbstractCollection, 116
- decaf::util::concurrent::ConcurrentStlMap, 914
- decaf::util::concurrent::CopyOnWriteArrayList, 1043
- decaf::util::concurrent::Mutex, 1962
- decaf::util::concurrent::Synchronizable, 2642
- decaf::util::StlMap, 2561
- decaf::util::StlQueue, 2570
- numWaiting
 - decaf::util::concurrent::ConditionHandle, 928
- numWake
 - decaf::util::concurrent::ConditionHandle, 929
- numberOfLeadingZeros
 - decaf::lang::Integer, 1506
 - decaf::lang::Long, 1733
- numberOfTrailingZeros
 - decaf::lang::Integer, 1507
 - decaf::lang::Long, 1733
- objectId
 - activemq::commands::RemoveInfo, 2271
- offer
 - decaf::util::concurrent::BlockingQueue, 543
 - decaf::util::concurrent::LinkedBlockingQueue, 1627
 - decaf::util::concurrent::SynchronousQueue, 2661
 - decaf::util::LinkedList, 1649
 - decaf::util::PriorityQueue, 2167
 - decaf::util::Queue, 2224
 - offerFirst
 - decaf::util::Deque, 1202
 - decaf::util::LinkedList, 1650
 - offerLast
 - decaf::util::Deque, 1203
 - decaf::util::LinkedList, 1650
 - offset
 - decaf::internal::nio::CharArrayBuffer, 785
 - inflate_state, 1447
 - onAsyncException
 - activemq::core::ActiveMQConnection, 204
 - onCommand
 - activemq::core::ActiveMQConnection, 204
 - activemq::transport::correlator::ResponseCorrelator, 2305
 - activemq::transport::DefaultTransportListener, 1179
 - activemq::transport::failover::FailoverTransportListener, 1321
 - activemq::transport::inactivity::InactivityMonitor, 1427
 - activemq::transport::logging::LoggingTransport, 1710
 - activemq::transport::mock::InternalCommandListener, 1528
 - activemq::transport::TransportFilter, 2805
 - activemq::transport::TransportListener, 2811
 - activemq::wireformat::openwire::OpenWireFormatNegotiator, 2062
 - onException
 - activemq::core::ActiveMQConnection, 205
 - activemq::transport::DefaultTransportListener, 1179
 - activemq::transport::failover::BackupTransport, 488
 - activemq::transport::failover::FailoverTransportListener, 1322
 - activemq::transport::inactivity::InactivityMonitor, 1428
 - activemq::transport::TransportFilter, 2806

- activemq::transport::TransportListener, 2811
- activemq::util::ServiceStopper, 2358
- cms::ExceptionListener, 1287
- onMessage
 - cms::MessageListener, 1917
- onProducerAck
 - activemq::core::ActiveMQProducer, 312
- onPropertiesReset
 - decaf::util::logging::PropertiesChangeListener, 2210
- onPropertyChanged
 - decaf::util::logging::PropertiesChangeListener, 2210
- onResponse
 - activemq::state::Tracked, 2766
- onSend
 - activemq::commands::ActiveMQBytesMessage, 170
 - activemq::commands::ActiveMQMessageTemplate, 299
 - activemq::commands::ActiveMQStreamMessage, 363
 - activemq::commands::Message, 1833
- onTransportException
 - activemq::transport::correlator::ResponseCorrelator, 2305
 - activemq::wireformat::openwire::OpenWireFormatNegotiator, 2063
- oneway
 - activemq::core::ActiveMQConnection, 204
 - activemq::core::ActiveMQSession, 350
 - activemq::transport::correlator::ResponseCorrelator, 2305
 - activemq::transport::failover::FailoverTransport, 1313
 - activemq::transport::inactivity::InactivityMonitor, 1427
 - activemq::transport::IOTransport, 1553
 - activemq::transport::logging::LoggingTransport, 1710
 - activemq::transport::mock::MockTransport, 1954
 - activemq::transport::Transport, 2794
 - activemq::transport::TransportFilter, 2806
 - activemq::wireformat::openwire::OpenWireFormatNegotiator, 2062
- op
 - code, 851
- opaque
 - z_stream_s, 2952
- operationType
 - activemq::commands::DestinationInfo, 1217
- operator<
 - activemq::commands::BrokerId, 567
 - activemq::commands::ConnectionId, 963
 - activemq::commands::ConsumerId, 1003
 - activemq::commands::LocalTransactionId, 1677
 - activemq::commands::MessageId, 1911
 - activemq::commands::ProducerId, 2185
 - activemq::commands::SessionId, 2381
 - activemq::commands::TransactionId, 2769
 - activemq::commands::XATransactionId, 2941
 - decaf::lang::Boolean, 548
 - decaf::lang::Byte, 619
 - decaf::lang::Character, 770, 771
 - decaf::lang::Comparable, 887
 - decaf::lang::Double, 1243, 1244
 - decaf::lang::Float, 1352
 - decaf::lang::Integer, 1507, 1508
 - decaf::lang::Long, 1734
 - decaf::lang::Short, 2403, 2404
 - decaf::net::URI, 2837
 - decaf::nio::ByteBuffer, 707
 - decaf::nio::CharBuffer, 796
 - decaf::nio::DoubleBuffer, 1267
 - decaf::nio::FloatBuffer, 1375
 - decaf::nio::IntBuffer, 1495
 - decaf::nio::LongBuffer, 1760
 - decaf::nio::ShortBuffer, 2426
 - decaf::util::concurrent::TimeUnit, 2759
 - decaf::util::Date, 1141

- decaf::util::logging::Level, 1618
- decaf::util::UUID, 2882
- operator*
 - decaf::lang::Pointer, 2088, 2089
- operator()
 - decaf::lang::ArrayPointerComparator, 471
 - decaf::lang::PointerComparator, 2092
 - decaf::util::Comparator, 890
 - decaf::util::comparators::Less, 1614
 - std::less< decaf::lang::ArrayPointer< T > >, 1615
 - std::less< decaf::lang::Pointer< T > >, 1615
- operator->
 - decaf::lang::Pointer, 2089
- operator=
 - activemq::cmsutil::CmsAccessor, 822
 - activemq::cmsutil::ResourceLifecycleManager, 2296
 - activemq::commands::BrokerId, 567
 - activemq::commands::ConnectionId, 963
 - activemq::commands::ConsumerId, 1003
 - activemq::commands::LocalTransactionId, 1677
 - activemq::commands::MessageId, 1911
 - activemq::commands::ProducerId, 2185
 - activemq::commands::SessionId, 2381
 - activemq::commands::TransactionId, 2769
 - activemq::commands::XATransactionId, 2941
 - activemq::library::ActiveMQCPP, 246
 - activemq::util::PrimitiveValueNode, 2156
 - activemq::util::ServiceSupport, 2361
 - decaf::lang::ArrayPointer, 467, 468
 - decaf::lang::Exception, 1285
 - decaf::lang::Pointer, 2089
 - decaf::lang::String, 2624
 - decaf::util::AbstractCollection, 116
 - decaf::util::ArrayList, 455
 - decaf::util::concurrent::CopyOnWriteArrayList, 1043
 - decaf::util::concurrent::LinkedBlockingQueue, 1628
 - decaf::util::Date, 1141
 - decaf::util::LinkedList, 1651
 - decaf::util::logging::LogManager, 1717
 - decaf::util::PriorityQueue, 2168
 - decaf::util::Properties, 2206
- operator==
 - activemq::commands::BrokerId, 567
 - activemq::commands::ConnectionId, 963
 - activemq::commands::ConsumerId, 1003
 - activemq::commands::LocalTransactionId, 1677
 - activemq::commands::MessageId, 1911
 - activemq::commands::ProducerId, 2185
 - activemq::commands::SessionId, 2381
 - activemq::commands::TransactionId, 2769
 - activemq::commands::XATransactionId, 2941
 - activemq::util::PrimitiveValueNode, 2156
 - decaf::lang, 91, 92
 - decaf::lang::ArrayPointer, 468–470
 - decaf::lang::Boolean, 548, 549
 - decaf::lang::Byte, 619, 620
 - decaf::lang::Character, 771
 - decaf::lang::Comparable, 888
 - decaf::lang::Double, 1244
 - decaf::lang::Float, 1353
 - decaf::lang::Integer, 1508
 - decaf::lang::Long, 1734, 1735
 - decaf::lang::Pointer, 2089, 2091
 - decaf::lang::Short, 2404
 - decaf::net::URI, 2838
 - decaf::nio::ByteBuffer, 707
 - decaf::nio::CharBuffer, 796
 - decaf::nio::DoubleBuffer, 1267
 - decaf::nio::FloatBuffer, 1375
 - decaf::nio::IntBuffer, 1495
 - decaf::nio::LongBuffer, 1760
 - decaf::nio::ShortBuffer, 2426

- decaf::util::concurrent::TimeUnit, 2759
- decaf::util::Date, 1141
- decaf::util::logging::Level, 1618
- decaf::util::UUID, 2882
- operator[]
 - activemq::wireformat::openwire::utils::HexTable, 1408
 - decaf::internal::util::ByteArrayAdapter, 638, 639
 - decaf::lang::ArrayPointer, 468
- opt_len
 - internal_state, 1526
- optimizedAcknowledge
 - activemq::commands::ConsumerInfo, 1017
- options
 - activemq::commands::ActiveMQDestination, 256
- ordered
 - activemq::commands::ActiveMQDestination, 257
- orderedTarget
 - activemq::commands::ActiveMQDestination, 257
- originalDestination
 - activemq::commands::Message, 1838
- originalTransactionId
 - activemq::commands::Message, 1838
- os
 - gz_header_s, 1399
- out
 - decaf::io::FileDescriptor, 1332
 - gz_state, 1401
- outputStream
 - decaf::io::FilterOutputStream, 1344
- own
 - decaf::io::FilterInputStream, 1340
 - decaf::io::FilterOutputStream, 1344
- ownDeflater
 - decaf::util::zip::DeflaterOutputStream, 1194
- ownInflater
 - decaf::util::zip::InflaterInputStream, 1464
- parent
 - activemq::cmsutil::CmsTemplate::ProducerExecutor, 2181
 - activemq::cmsutil::CmsTemplate::ReceiveExecutor, 2250
- park
 - decaf::util::concurrent::locks::LockSupport, 1691
- parkNanos
 - decaf::util::concurrent::locks::LockSupport, 1692
- parkUntil
 - decaf::util::concurrent::locks::LockSupport, 1692
- parse
 - decaf::internal::util::HexStringParser, 1406
 - decaf::util::logging::Level, 1618
- parseAuthority
 - decaf::internal::net::URIHelper, 2848
- parseBoolean
 - decaf::lang::Boolean, 549
- parseByte
 - decaf::lang::Byte, 620, 621
- parseComposite
 - activemq::util::URISupport, 2855
- parseDouble
 - decaf::internal::util::HexStringParser, 1407
 - decaf::lang::Double, 1245
- parseFloat
 - decaf::internal::util::HexStringParser, 1407
 - decaf::lang::Float, 1353
- parseInt
 - decaf::lang::Integer, 1509
- parseLong
 - decaf::lang::Long, 1735, 1736
- parseQuery
 - activemq::util::URISupport, 2855, 2856
- parseServerAuthority
 - decaf::net::URI, 2838
- parseShort
 - decaf::lang::Short, 2405
- parseURI
 - decaf::internal::net::URIHelper, 2848
- parseURL
 - activemq::util::URISupport, 2856
- password

- activemq::commands::Connection-Info, 973
- path
 - gz_state, 1401
- peek
 - activemq::core::FifoMessageDispatch-Channel, 1327
 - activemq::core::MessageDispatch-Channel, 1889
 - activemq::core::SimplePriority-MessageDispatchChannel, 2448
 - decaf::internal::util::TimerTaskHeap, 2755
 - decaf::util::concurrent::Linked-BlockingQueue, 1628
 - decaf::util::concurrent::Synchronous-Queue, 2662
 - decaf::util::LinkedList, 1651
 - decaf::util::PriorityQueue, 2168
 - decaf::util::Queue, 2225
- peekFirst
 - decaf::util::Deque, 1203
 - decaf::util::LinkedList, 1651
- peekLast
 - decaf::util::Deque, 1204
 - decaf::util::LinkedList, 1652
- peerBrokerInfos
 - activemq::commands::BrokerInfo, 578
- pending
 - internal_state, 1526
- pending_buf
 - internal_state, 1526
- pending_buf_size
 - internal_state, 1526
- pending_out
 - internal_state, 1526
- persistent
 - activemq::commands::Message, 1838
- physicalName
 - activemq::commands::ActiveMQ-Destination, 257
- poll
 - decaf::util::concurrent::Blocking-Queue, 543
 - decaf::util::concurrent::Linked-BlockingQueue, 1629
 - decaf::util::concurrent::Synchronous-Queue, 2662, 2663
 - decaf::util::LinkedList, 1652
 - decaf::util::PriorityQueue, 2169
 - decaf::util::Queue, 2225
- pollFirst
 - decaf::util::Deque, 1204
 - decaf::util::LinkedList, 1652
- pollLast
 - decaf::util::Deque, 1205
 - decaf::util::LinkedList, 1653
- pop
 - decaf::util::Deque, 1206
 - decaf::util::LinkedList, 1653
 - decaf::util::StlQueue, 2570
- port
 - decaf::net::SocketImpl, 2487
- pos
 - gz_state, 1401
- position
 - decaf::nio::Buffer, 587
- pow
 - decaf::lang::Math, 1810
- prefetch
 - activemq::commands::Consumer-Control, 996
- prefetchSize
 - activemq::commands::Consumer-Info, 1017
- prepare
 - activemq::core::ActiveMQTransaction-Context, 429
 - cms::XAResource, 2931
- prestartAllCoreThreads
 - decaf::util::concurrent::ThreadPool-Executor, 2728
- prestartCoreThread
 - decaf::util::concurrent::ThreadPool-Executor, 2728
- prev
 - internal_state, 1526
- prev_length
 - internal_state, 1526
- prev_match
 - internal_state, 1526
- previous
 - decaf::util::concurrent::CopyOn-WriteArrayList::ArrayListIterator, 461
 - decaf::util::ListIterator, 1673

- previousIndex
 - decaf::util::concurrent::CopyOn-WriteArrayList::ArrayListIterator, 461
 - decaf::util::ListIterator, 1673
- printStackTrace
 - cms::CMSException, 828
 - decaf::lang::Exception, 1285
 - decaf::lang::Throwable, 2736
- priority
 - activemq::commands::ConsumerInfo, 1017
 - activemq::commands::Message, 1838
- processBeginTransaction
 - activemq::state::CommandVisitor, 874
 - activemq::state::CommandVisitor-Adapter, 881
 - activemq::state::ConnectionStateTracker, 986
- processBrokerError
 - activemq::state::CommandVisitor, 874
 - activemq::state::CommandVisitor-Adapter, 881
- processBrokerInfo
 - activemq::state::CommandVisitor, 874
 - activemq::state::CommandVisitor-Adapter, 881
- processCommitTransactionOnePhase
 - activemq::state::CommandVisitor, 874
 - activemq::state::CommandVisitor-Adapter, 881
 - activemq::state::ConnectionStateTracker, 987
- processCommitTransactionTwoPhase
 - activemq::state::CommandVisitor, 874
 - activemq::state::CommandVisitor-Adapter, 881
 - activemq::state::ConnectionStateTracker, 987
- processConnectionControl
 - activemq::state::CommandVisitor, 874
 - activemq::state::CommandVisitor-Adapter, 881
- processConnectionError
 - activemq::state::CommandVisitor, 875
 - activemq::state::CommandVisitor-Adapter, 881
- processConnectionInfo
 - activemq::state::CommandVisitor, 875
 - activemq::state::CommandVisitor-Adapter, 882
 - activemq::state::ConnectionStateTracker, 987
- processConsumerControl
 - activemq::state::CommandVisitor, 875
 - activemq::state::CommandVisitor-Adapter, 882
- processConsumerInfo
 - activemq::state::CommandVisitor, 875
 - activemq::state::CommandVisitor-Adapter, 882
 - activemq::state::ConnectionStateTracker, 987
- processControlCommand
 - activemq::state::CommandVisitor, 875
 - activemq::state::CommandVisitor-Adapter, 882
- processDestinationInfo
 - activemq::state::CommandVisitor, 875
 - activemq::state::CommandVisitor-Adapter, 882
 - activemq::state::ConnectionStateTracker, 987
- processEndTransaction
 - activemq::state::CommandVisitor, 875
 - activemq::state::CommandVisitor-Adapter, 882
 - activemq::state::ConnectionStateTracker, 987
- processFlushCommand
 - activemq::state::CommandVisitor, 875
 - activemq::state::CommandVisitor-Adapter, 882
- processForgetTransaction

- activemq::state::CommandVisitor, 875
- activemq::state::CommandVisitor-Adapter, 882
- processKeepAliveInfo
 - activemq::state::CommandVisitor, 876
 - activemq::state::CommandVisitor-Adapter, 882
- processMessage
 - activemq::state::CommandVisitor, 876
 - activemq::state::CommandVisitor-Adapter, 883
 - activemq::state::ConnectionStateTracker, 987
- processMessageAck
 - activemq::state::CommandVisitor, 876
 - activemq::state::CommandVisitor-Adapter, 883
 - activemq::state::ConnectionStateTracker, 988
- processMessageDispatch
 - activemq::state::CommandVisitor, 876
 - activemq::state::CommandVisitor-Adapter, 883
- processMessageDispatchNotification
 - activemq::state::CommandVisitor, 876
 - activemq::state::CommandVisitor-Adapter, 883
- processMessagePull
 - activemq::state::CommandVisitor, 876
 - activemq::state::CommandVisitor-Adapter, 883
- processPrepareTransaction
 - activemq::state::CommandVisitor, 876
 - activemq::state::CommandVisitor-Adapter, 883
 - activemq::state::ConnectionStateTracker, 988
- processProducerAck
 - activemq::state::CommandVisitor, 876
 - activemq::state::CommandVisitor-Adapter, 883
- processProducerInfo
 - activemq::state::CommandVisitor, 876
 - activemq::state::CommandVisitor-Adapter, 883
 - activemq::state::ConnectionStateTracker, 988
- processRecoverTransactions
 - activemq::state::CommandVisitor, 877
 - activemq::state::CommandVisitor-Adapter, 883
- processRemoveConnection
 - activemq::state::CommandVisitor, 877
 - activemq::state::CommandVisitor-Adapter, 883
 - activemq::state::ConnectionStateTracker, 988
- processRemoveConsumer
 - activemq::state::CommandVisitor, 877
 - activemq::state::CommandVisitor-Adapter, 884
 - activemq::state::ConnectionStateTracker, 988
- processRemoveDestination
 - activemq::state::CommandVisitor, 877
 - activemq::state::CommandVisitor-Adapter, 884
 - activemq::state::ConnectionStateTracker, 988
- processRemoveInfo
 - activemq::state::CommandVisitor, 877
 - activemq::state::CommandVisitor-Adapter, 884
- processRemoveProducer
 - activemq::state::CommandVisitor, 877
 - activemq::state::CommandVisitor-Adapter, 884
 - activemq::state::ConnectionStateTracker, 988
- processRemoveSession
 - activemq::state::CommandVisitor, 877
 - activemq::state::CommandVisitor-Adapter, 884

- activemq::state::ConnectionStateTracker, 989
- processRemoveSubscriptionInfo
 - activemq::state::CommandVisitor, 877
 - activemq::state::CommandVisitor-Adapter, 884
- processReplayCommand
 - activemq::state::CommandVisitor, 878
 - activemq::state::CommandVisitor-Adapter, 884
- processResponse
 - activemq::state::CommandVisitor, 878
 - activemq::state::CommandVisitor-Adapter, 884
- processRollbackTransaction
 - activemq::state::CommandVisitor, 878
 - activemq::state::CommandVisitor-Adapter, 884
 - activemq::state::ConnectionStateTracker, 989
- processSessionInfo
 - activemq::state::CommandVisitor, 878
 - activemq::state::CommandVisitor-Adapter, 885
 - activemq::state::ConnectionStateTracker, 989
- processShutdownInfo
 - activemq::state::CommandVisitor, 878
 - activemq::state::CommandVisitor-Adapter, 885
- processTransactionInfo
 - activemq::state::CommandVisitor, 878
 - activemq::state::CommandVisitor-Adapter, 885
- processWireFormat
 - activemq::state::CommandVisitor, 878
 - activemq::state::CommandVisitor-Adapter, 885
- producerId
 - activemq::commands::Message, 1838
 - activemq::commands::MessageId, 1912
 - activemq::commands::ProducerAck, 2174
 - activemq::commands::ProducerInfo, 2194
- producerIds
 - activemq::core::ActiveMQSession, 354
- producerSequenceId
 - activemq::commands::MessageId, 1912
- producerSequenceIds
 - activemq::core::ActiveMQSession, 355
- propertyExists
 - activemq::commands::ActiveMQ-MessageTemplate, 299
 - cms::Message, 1856
- propertyNames
 - activemq::util::ActiveMQProperties, 319
 - cms::CMSProperties, 833
 - decaf::util::Properties, 2206
- providerGenerateSeed
 - decaf::internal::security::SecureRandomImpl, 2327
 - decaf::security::SecureRandomSpi, 2330
- providerGetDefaultSSLParameters
 - decaf::net::ssl::SSLContextSpi, 2499
- providerGetServerSocketFactory
 - decaf::internal::net::ssl::openssl::OpenSSLContextSpi, 1999
 - decaf::net::ssl::SSLContextSpi, 2499
- providerGetSocketFactory
 - decaf::internal::net::ssl::openssl::OpenSSLContextSpi, 2000
 - decaf::net::ssl::SSLContextSpi, 2500
- providerGetSupportedSSLParameters
 - decaf::net::ssl::SSLContextSpi, 2500
- providerInit
 - decaf::internal::net::ssl::openssl::OpenSSLContextSpi, 2000
 - decaf::net::ssl::SSLContextSpi, 2501
- providerNextBytes
 - decaf::internal::security::SecureRandomImpl, 2327, 2328
 - decaf::security::SecureRandomSpi, 2330

- providerSetSeed
 - decaf::internal::security::SecureRandomImpl, 2328
 - decaf::security::SecureRandomSpi, 2330
- publish
 - decaf::util::logging::ConsoleHandler, 991
 - decaf::util::logging::Handler, 1404
 - decaf::util::logging::StreamHandler, 2605
- purge
 - decaf::util::concurrent::ThreadPoolExecutor, 2728
 - decaf::util::Timer, 2741
- push
 - decaf::util::Deque, 1206
 - decaf::util::LinkedList, 1653
 - decaf::util::StlQueue, 2570
- put
 - decaf::internal::nio::ByteBuffer, 671, 672
 - decaf::internal::nio::CharArrayBuffer, 782, 783
 - decaf::internal::nio::DoubleArrayBuffer, 1257, 1258
 - decaf::internal::nio::FloatArrayBuffer, 1366
 - decaf::internal::nio::IntArrayBuffer, 1486, 1487
 - decaf::internal::nio::LongArrayBuffer, 1751
 - decaf::internal::nio::ShortArrayBuffer, 2417, 2418
 - decaf::internal::util::ByteArrayAdapter, 639
 - decaf::nio::ByteBuffer, 707–709
 - decaf::nio::CharBuffer, 796–799
 - decaf::nio::DoubleBuffer, 1267–1269
 - decaf::nio::FloatBuffer, 1375–1377
 - decaf::nio::IntBuffer, 1495–1497
 - decaf::nio::LongBuffer, 1760–1763
 - decaf::nio::ShortBuffer, 2427–2429
 - decaf::util::concurrent::BlockingQueue, 544
 - decaf::util::concurrent::ConcurrentStlMap, 914
 - decaf::util::concurrent::LinkedBlockingQueue, 1630
 - decaf::util::concurrent::SynchronousQueue, 2663
 - decaf::util::Map, 1776
 - decaf::util::StlMap, 2561
- put_byte
 - deflate.h, 3145
- putAll
 - decaf::util::concurrent::ConcurrentStlMap, 915
 - decaf::util::Map, 1777
 - decaf::util::StlMap, 2561, 2562
- putChar
 - decaf::internal::nio::ByteBuffer, 672, 673
 - decaf::internal::util::ByteArrayAdapter, 639
 - decaf::nio::ByteBuffer, 710
- putDouble
 - decaf::internal::nio::ByteBuffer, 673, 674
 - decaf::internal::util::ByteArrayAdapter, 640
 - decaf::nio::ByteBuffer, 711
- putDoubleAt
 - decaf::internal::util::ByteArrayAdapter, 640
- putFloat
 - decaf::internal::nio::ByteBuffer, 675
 - decaf::internal::util::ByteArrayAdapter, 641
 - decaf::nio::ByteBuffer, 712
- putFloatAt
 - decaf::internal::util::ByteArrayAdapter, 641
- putIfAbsent
 - decaf::util::concurrent::ConcurrentMap, 898
 - decaf::util::concurrent::ConcurrentStlMap, 915
- putInt
 - decaf::internal::nio::ByteBuffer, 676
 - decaf::internal::util::ByteArrayAdapter, 642
 - decaf::nio::ByteBuffer, 713
- putIntAt
 - decaf::internal::util::ByteArrayAdapter, 642
- putLong

- decaf::internal::nio::ByteBuffer, 677
- decaf::internal::util::ByteArray-Adapter, 642
- decaf::nio::ByteBuffer, 714
- putLongAt
 - decaf::internal::util::ByteArray-Adapter, 643
- putShort
 - decaf::internal::nio::ByteBuffer, 678
 - decaf::internal::util::ByteArray-Adapter, 643
 - decaf::nio::ByteBuffer, 715
- putShortAt
 - decaf::internal::util::ByteArray-Adapter, 644
- quotelllegal
 - decaf::internal::net::URLEncoder-Decoder, 2843
- random
 - decaf::lang::Math, 1810
- randomUUID
 - decaf::util::UUID, 2882
- raw
 - gz_state, 1401
- reached
 - decaf::net::InetAddress, 1444
- read
 - decaf::internal::net::ssl::openssl::OpenSSLSocket, 2024
 - decaf::internal::net::tcp::TcpSocket, 2686
 - decaf::internal::util::ByteArray-Adapter, 644
 - decaf::io::InputStream, 1469, 1470
 - decaf::io::Reader, 2240–2242
 - decaf::lang::Readable, 2236
 - decaf::nio::CharBuffer, 800
- readAsciiString
 - activemq::wireformat::openwire::marshal::BaseDataStream-Marshaller, 517
- readBoolean
 - activemq::commands::ActiveMQ-BytesMessage, 171
 - activemq::commands::ActiveMQ-StreamMessage, 363
- activemq::wireformat::openwire::utils::BooleanStream, 554
- cms::BytesMessage, 721
- cms::StreamMessage, 2608
- decaf::io::DataInput, 1087
- decaf::io::DataInputStream, 1096
- readByte
 - activemq::commands::ActiveMQ-BytesMessage, 171
 - activemq::commands::ActiveMQ-StreamMessage, 363
 - cms::BytesMessage, 722
 - cms::StreamMessage, 2609
 - decaf::io::DataInput, 1088
 - decaf::io::DataInputStream, 1096
- readBytes
 - activemq::commands::ActiveMQ-BytesMessage, 171, 172
 - activemq::commands::ActiveMQ-StreamMessage, 364, 365
 - cms::BytesMessage, 722, 723
 - cms::StreamMessage, 2609, 2610
- readChar
 - activemq::commands::ActiveMQ-BytesMessage, 173
 - activemq::commands::ActiveMQ-StreamMessage, 365
 - cms::BytesMessage, 724
 - cms::StreamMessage, 2611
 - decaf::io::DataInput, 1088
 - decaf::io::DataInputStream, 1097
- readConfiguration
 - decaf::util::logging::LogManager, 1717
- readDouble
 - activemq::commands::ActiveMQ-BytesMessage, 173
 - activemq::commands::ActiveMQ-StreamMessage, 366
 - cms::BytesMessage, 724
 - cms::StreamMessage, 2612
 - decaf::io::DataInput, 1088
 - decaf::io::DataInputStream, 1097
- readFloat
 - activemq::commands::ActiveMQ-BytesMessage, 174
 - activemq::commands::ActiveMQ-StreamMessage, 366
 - cms::BytesMessage, 725
 - cms::StreamMessage, 2612

- decaf::io::DataInput, 1089
- decaf::io::DataInputStream, 1097
- readFully
 - decaf::io::DataInput, 1089
 - decaf::io::DataInputStream, 1098
- readInt
 - activemq::commands::ActiveMQ-BytesMessage, 174
 - activemq::commands::ActiveMQ-StreamMessage, 367
 - cms::BytesMessage, 725
 - cms::StreamMessage, 2613
 - decaf::io::DataInput, 1090
 - decaf::io::DataInputStream, 1099
- readLine
 - decaf::io::DataInput, 1091
 - decaf::io::DataInputStream, 1099
- readLock
 - decaf::util::concurrent::locks::Read-WriteLock, 2248
- readLong
 - activemq::commands::ActiveMQ-BytesMessage, 175
 - activemq::commands::ActiveMQ-StreamMessage, 367
 - cms::BytesMessage, 726
 - cms::StreamMessage, 2613
 - decaf::io::DataInput, 1091
 - decaf::io::DataInputStream, 1100
- readOnly
 - decaf::internal::nio::CharArrayBuffer, 785
- readShort
 - activemq::commands::ActiveMQ-BytesMessage, 175
 - activemq::commands::ActiveMQ-StreamMessage, 368
 - cms::BytesMessage, 726
 - cms::StreamMessage, 2614
 - decaf::io::DataInput, 1091
 - decaf::io::DataInputStream, 1100
- readString
 - activemq::commands::ActiveMQ-BytesMessage, 175
 - activemq::commands::ActiveMQ-StreamMessage, 368
 - cms::BytesMessage, 727
 - cms::StreamMessage, 2614
 - decaf::io::DataInput, 1092
 - decaf::io::DataInputStream, 1101
- readString16
 - activemq::util::MarshallingSupport, 1798
- readString32
 - activemq::util::MarshallingSupport, 1799
- readUTF
 - activemq::commands::ActiveMQ-BytesMessage, 176
 - cms::BytesMessage, 728
 - decaf::io::DataInput, 1093
 - decaf::io::DataInputStream, 1102
- readUnsignedByte
 - decaf::io::DataInput, 1092
 - decaf::io::DataInputStream, 1101
- readUnsignedShort
 - activemq::commands::ActiveMQ-BytesMessage, 176
 - activemq::commands::ActiveMQ-StreamMessage, 369
 - cms::BytesMessage, 727
 - cms::StreamMessage, 2615
 - decaf::io::DataInput, 1093
 - decaf::io::DataInputStream, 1101
- readonly
 - decaf::io::FileDescriptor, 1332
- ready
 - decaf::io::InputStreamReader, 1476
 - decaf::io::Reader, 2242
- rebalanceConnection
 - activemq::commands::Connection-Control, 943
- receive
 - activemq::cmsutil::CachedConsumer, 736, 737
 - activemq::cmsutil::CmsTemplate, 843, 844
 - activemq::core::ActiveMQConsumer, 241, 242
 - cms::MessageConsumer, 1879
- receiveNoWait
 - activemq::cmsutil::CachedConsumer, 737
 - activemq::core::ActiveMQConsumer, 242
 - cms::MessageConsumer, 1879
- receiveSelected
 - activemq::cmsutil::CmsTemplate, 845
- recievedByDFBridge

- activemq::commands::Message, 1838
- reconnect
 - activemq::transport::failover::FailoverTransport, 1313
 - activemq::transport::IOTransport, 1553
 - activemq::transport::mock::MockTransport, 1955
 - activemq::transport::Transport, 2795
 - activemq::transport::TransportFilter, 2806
- reconnectTo
 - activemq::commands::ConnectionControl, 943
- recover
 - activemq::cmsutil::PooledSession, 2105
 - activemq::core::ActiveMQSession, 350
 - activemq::core::ActiveMQTransactionContext, 429
 - cms::Session, 2375
 - cms::XAResource, 2931
- redeliveryCounter
 - activemq::commands::Message, 1838
 - activemq::commands::MessageDispatch, 1886
- redispatch
 - activemq::core::ActiveMQSession, 350
- registerFactory
 - activemq::transport::TransportRegistry, 2814
 - activemq::wireformat::WireFormatRegistry, 2906
- rejectedExecution
 - decaf::util::concurrent::RejectedExecutionHandler, 2267
 - decaf::util::concurrent::ThreadPoolExecutor::AbortPolicy, 106
 - decaf::util::concurrent::ThreadPoolExecutor::CallerRunsPolicy, 748
 - decaf::util::concurrent::ThreadPoolExecutor::CallerRunsPolicy::DiscardOldestPolicy, 1225
 - decaf::util::concurrent::ThreadPoolExecutor::CallerRunsPolicy::DiscardPolicy, 1226
- relativize
 - decaf::net::URI, 2839
- release
 - decaf::lang::ArrayPointer, 468
 - decaf::lang::Pointer, 2089
 - decaf::util::concurrent::atomic::AtomicRefCounter, 483
 - decaf::util::concurrent::Semaphore, 2337
- releaseAll
 - activemq::cmsutil::ResourceLifecycleManager, 2296
- remaining
 - decaf::nio::Buffer, 588
- remainingCapacity
 - decaf::util::concurrent::BlockingQueue, 544
 - decaf::util::concurrent::LinkedBlockingQueue, 1630
 - decaf::util::concurrent::SynchronousQueue, 2663
- remove
 - activemq::util::ActiveMQProperties, 319
 - cms::CMSProperties, 833
 - decaf::internal::util::TimerTaskHeap, 2755
 - decaf::util::AbstractCollection, 116
 - decaf::util::AbstractQueue, 142
 - decaf::util::ArrayList, 455
 - decaf::util::Collection, 861
 - decaf::util::concurrent::ConcurrentMap, 899
 - decaf::util::concurrent::ConcurrentStlMap, 916, 917
 - decaf::util::concurrent::CopyOnWriteArrayList, 1043
 - decaf::util::concurrent::CopyOnWriteArrayList::ArrayListIterator, 461
 - decaf::util::concurrent::CopyOnWriteArraySet, 1058
 - decaf::util::concurrent::LinkedBlockingQueue, 1631
 - decaf::util::concurrent::SynchronousQueue, 2664
 - decaf::util::concurrent::ThreadPoolExecutor, 2728
 - decaf::util::Iterator, 1560

- decaf::util::LinkedList, 1654, 1655
- decaf::util::Map, 1777
- decaf::util::PriorityQueue, 2169, 2170
- decaf::util::Properties, 2207
- decaf::util::Queue, 2226
- decaf::util::StlList, 2550
- decaf::util::StlMap, 2562
- decaf::util::StlSet, 2580
- removeAll
 - activemq::core::FifoMessageDispatch-Channel, 1327
 - activemq::core::MessageDispatch-Channel, 1889
 - activemq::core::SimplePriority-MessageDispatchChannel, 2449
 - decaf::util::AbstractCollection, 117
 - decaf::util::AbstractSet, 153
 - decaf::util::Collection, 862
 - decaf::util::concurrent::CopyOn-WriteArrayList, 1044
 - decaf::util::concurrent::CopyOn-WriteArraySet, 1059
 - decaf::util::concurrent::Synchronous-Queue, 2664
- removeAt
 - decaf::util::AbstractList, 132
 - decaf::util::AbstractSequentialList, 150
 - decaf::util::ArrayList, 456
 - decaf::util::concurrent::CopyOn-WriteArrayList, 1044
 - decaf::util::List, 1668
 - decaf::util::StlList, 2550
- removeConsumer
 - activemq::core::ActiveMQSession, 351
 - activemq::state::SessionState, 2397
- removeDispatcher
 - activemq::core::ActiveMQConnection, 205
- removeFirst
 - decaf::util::Deque, 1207
 - decaf::util::LinkedList, 1655
- removeFirstOccurrence
 - decaf::util::Deque, 1208
 - decaf::util::LinkedList, 1656
- removeHandler
 - decaf::util::logging::Logger, 1703
- removeLast
 - decaf::util::Deque, 1208
 - decaf::util::LinkedList, 1656
- removeLastOccurrence
 - decaf::util::Deque, 1209
 - decaf::util::LinkedList, 1657
- removeProducer
 - activemq::core::ActiveMQConnection, 205
 - activemq::core::ActiveMQSession, 351
 - activemq::state::SessionState, 2397
- removeProperty
 - activemq::wireformat::stomp::Stomp-Frame, 2591
- removePropertyChangeListener
 - decaf::util::logging::LogManager, 1718
- removeRange
 - decaf::util::AbstractList, 133
- removeServiceListener
 - activemq::util::ServiceSupport, 2361
- removeSession
 - activemq::core::ActiveMQConnection, 206
 - activemq::state::ConnectionState, 984
- removeSynchronization
 - activemq::core::ActiveMQTransaction-Context, 430
- removeTask
 - activemq::threads::CompositeTask-Runner, 895
- removeTempDestination
 - activemq::state::ConnectionState, 984
- removeTransactionState
 - activemq::state::ConnectionState, 984
- removeTransportListener
 - activemq::core::ActiveMQConnection, 206
- removeURI
 - activemq::transport::Composite-Transport, 897
 - activemq::transport::failover::Failover-Transport, 1314
 - activemq::transport::failover::URI-Pool, 2853
- renegotiateWireFormat

- activemq::wireformat::openwire::-
OpenWireFormat, 2054
- replace
 - decaf::util::concurrent::Concurrent-
Map, 900, 901
 - decaf::util::concurrent::Concurrent-
StlMap, 917, 918
- replyTo
 - activemq::commands::Message,
1838
- reportError
 - decaf::util::logging::Handler, 1404
- request
 - activemq::transport::correlator::-
ResponseCorrelator, 2306
 - activemq::transport::failover::Failover-
Transport, 1314
 - activemq::transport::IOTransport,
1553, 1554
 - activemq::transport::logging::-
LoggingTransport, 1711
 - activemq::transport::mock::Mock-
Transport, 1955
 - activemq::transport::Transport, 2795,
2796
 - activemq::transport::TransportFilter,
2807
 - activemq::wireformat::openwire::-
OpenWireFormatNegotiator,
2063
- reserved
 - z_stream_s, 2952
- reset
 - activemq::commands::ActiveMQ-
BytesMessage, 177
 - activemq::commands::ActiveMQ-
StreamMessage, 369
 - activemq::state::ConnectionState,
984
 - cms::BytesMessage, 728
 - decaf::internal::util::TimerTaskHeap,
2755
 - decaf::io::BufferedInputStream, 593
 - decaf::io::ByteArrayInputStream, 685
 - decaf::io::ByteArrayOutputStream,
689
 - decaf::io::FilterInputStream, 1339
 - decaf::io::InputStream, 1471
 - decaf::io::PushbackInputStream,
2218
 - decaf::io::Reader, 2243
 - decaf::lang::ArrayPointer, 469
 - decaf::lang::Pointer, 2090
 - decaf::nio::Buffer, 588
 - decaf::util::logging::LogManager,
1718
 - decaf::util::StringTokenizer, 2630
 - decaf::util::zip::Adler32, 440
 - decaf::util::zip::Checksum, 811
 - decaf::util::zip::CRC32, 1066
 - decaf::util::zip::Deflater, 1185
 - decaf::util::zip::Inflater, 1453
 - decaf::util::zip::InflaterInputStream,
1462
- resize
 - decaf::internal::util::ByteArray-
Adapter, 645
- resolve
 - decaf::net::URI, 2839
- resolveDestinationName
 - activemq::cmsutil::CmsDestination-
Accessor, 825
 - activemq::cmsutil::Destination-
Resolver, 1223
 - activemq::cmsutil::DynamicDestination-
Resolver, 1273
- restore
 - activemq::state::ConnectionState-
Tracker, 989
- restoreTransport
 - activemq::transport::failover::Failover-
Transport, 1315
- result
 - activemq::commands::Integer-
Response, 1519
- resume
 - activemq::commands::Connection-
Control, 943
- retainAll
 - decaf::util::AbstractCollection, 118
 - decaf::util::Collection, 863
 - decaf::util::concurrent::CopyOn-
WriteArrayList, 1045
 - decaf::util::concurrent::CopyOn-
WriteArraySet, 1060
 - decaf::util::concurrent::Synchronous-
Queue, 2664
- retroactive
 - activemq::commands::Consumer-
Info, 1017

- returnInstance
 - decaf::util::logging::LogWriter, 1725
- returnSession
 - activemq::cmsutil::SessionPool, 2395
- reverse
 - decaf::lang::Integer, 1510
 - decaf::lang::Long, 1736
 - decaf::util::StlQueue, 2571
- reverseBytes
 - decaf::lang::Integer, 1510
 - decaf::lang::Long, 1736
 - decaf::lang::Short, 2406
- rewind
 - decaf::nio::Buffer, 588
- rollback
 - activemq::cmsutil::PooledSession, 2106
 - activemq::core::ActiveMQConsumer, 242
 - activemq::core::ActiveMQSession, 351
 - activemq::core::ActiveMQTransaction-Context, 430
 - activemq::core::ActiveMQXASession, 438
 - cms::Session, 2376
 - cms::XAResource, 2932
- rotateLeft
 - decaf::lang::Integer, 1510
 - decaf::lang::Long, 1737
- rotateRight
 - decaf::lang::Integer, 1511
 - decaf::lang::Long, 1737
- round
 - decaf::lang::Math, 1811, 1812
- run
 - activemq::threads::CompositeTask-Runner, 895
 - activemq::threads::DedicatedTask-Runner, 1145
 - activemq::threads::SchedulerTimer-Task, 2320
 - activemq::transport::inactivity::Read-Checker, 2237
 - activemq::transport::inactivity::Write-Checker, 2908
 - activemq::transport::IOTransport, 1554
 - activemq::transport::mock::Internal-CommandListener, 1528
 - decaf::lang::Runnable, 2312
 - decaf::lang::Thread, 2710
- sane
 - inflate_state, 1447
- scheduledPeriodically
 - activemq::threads::Scheduler, 2319
- schedule
 - decaf::util::Timer, 2742–2746
- scheduleAtFixedRate
 - decaf::util::Timer, 2747–2750
- scheduledExecutionTime
 - decaf::util::TimerTask, 2752
- seek
 - gz_state, 1401
- selector
 - activemq::cmsutil::CmsTemplate::ReceiveExecutor, 2250
 - activemq::commands::Consumer-Info, 1017
 - activemq::commands::Subscription-Info, 2635
- semaphore
 - decaf::util::concurrent::Condition-Handle, 929
- send
 - activemq::cmsutil::CachedProducer, 741–743
 - activemq::cmsutil::CmsTemplate, 846, 847
 - activemq::core::ActiveMQProducer, 312–314
 - activemq::core::ActiveMQSession, 352
 - cms::MessageProducer, 1928–1930
- sendPullRequest
 - activemq::core::ActiveMQConnection, 206
- sendUrgentData
 - decaf::internal::net::ssl::openssl::OpenSSLSocket, 2024
 - decaf::net::Socket, 2465
 - decaf::net::SocketImpl, 2485
- serviceName
 - activemq::commands::Discovery-Event, 1229
- sessionId

- activemq::commands::ConsumerId, 1004
- activemq::commands::ProducerId, 2186
- activemq::commands::SessionInfo, 2389
- sessionInfo
 - activemq::core::ActiveMQSession, 355
- set
 - decaf::util::AbstractList, 133
 - decaf::util::AbstractSequentialList, 151
 - decaf::util::ArrayList, 456
 - decaf::util::concurrent::atomic::AtomicBoolean, 475
 - decaf::util::concurrent::atomic::AtomicInteger, 480
 - decaf::util::concurrent::atomic::AtomicReference, 486
 - decaf::util::concurrent::CopyOnWriteArrayList, 1046
 - decaf::util::concurrent::CopyOnWriteArrayList::ArrayListIterator, 462
 - decaf::util::LinkedList, 1657
 - decaf::util::List, 1669
 - decaf::util::ListIterator, 1674
 - decaf::util::StlList, 2551
- setAbsolute
 - decaf::internal::net::URIType, 2865
- setAckHandler
 - activemq::commands::Message, 1833
- setAckMode
 - activemq::commands::SessionInfo, 2389
- setAckType
 - activemq::commands::MessageAck, 1871
- setAdditionalPredicate
 - activemq::commands::ConsumerInfo, 1014
- setAddress
 - decaf::net::DatagramPacket, 1083
- setAdvisory
 - activemq::commands::ActiveMQ-Destination, 254
- setAlwaysSyncSend
 - activemq::core::ActiveMQConnection, 207
 - activemq::core::ActiveMQConnectionFactory, 222
- setArrival
 - activemq::commands::Message, 1834
- setAuthority
 - decaf::internal::net::URIType, 2865
- setBackOffMultiplier
 - activemq::core::policies::Default-RedeliveryPolicy, 1153
 - activemq::core::RedeliveryPolicy, 2254
 - activemq::transport::failover::Failover-Transport, 1315
- setBackup
 - activemq::transport::failover::Failover-Transport, 1315
- setBackupPoolSize
 - activemq::transport::failover::Backup-TransportPool, 491
 - activemq::transport::failover::Failover-Transport, 1315
- setBody
 - activemq::wireformat::stomp::Stomp-Frame, 2591
- setBodyBytes
 - activemq::commands::ActiveMQ-BytesMessage, 177
 - cms::BytesMessage, 728
- setBool
 - activemq::util::PrimitiveList, 2121
 - activemq::util::PrimitiveMap, 2132
 - activemq::util::PrimitiveValueNode, 2156
- setBoolean
 - activemq::commands::ActiveMQ-MapMessage, 279
 - cms::MapMessage, 1787
- setBooleanProperty
 - activemq::commands::ActiveMQ-MessageTemplate, 299
 - activemq::wireformat::openwire::utils::MessageProperty-Interceptor, 1937
 - cms::Message, 1857
- setBranchQualifier
 - activemq::commands::XATransaction-Id, 2941

- setBrokerId
 - activemq::commands::BrokerInfo, 576
- setBrokerInTime
 - activemq::commands::Message, 1834
- setBrokerMasterConnector
 - activemq::commands::Connection-Info, 972
- setBrokerName
 - activemq::commands::BrokerInfo, 576
 - activemq::commands::Discovery-Event, 1229
- setBrokerOutTime
 - activemq::commands::Message, 1834
- setBrokerPath
 - activemq::commands::Connection-Info, 972
 - activemq::commands::Consumer-Info, 1014
 - activemq::commands::Destination-Info, 1216
 - activemq::commands::Message, 1834
 - activemq::commands::ProducerInfo, 2193
- setBrokerSequenceld
 - activemq::commands::MessageId, 1911
- setBrokerURI
 - activemq::core::ActiveMQConnection-Factory, 222, 223
- setBrokerURL
 - activemq::commands::BrokerInfo, 576
 - activemq::core::ActiveMQConnection, 207
- setBrokerUploadUrl
 - activemq::commands::BrokerInfo, 576
- setBrowser
 - activemq::commands::Consumer-Info, 1014
- setByte
 - activemq::commands::ActiveMQ-MapMessage, 279
 - activemq::util::PrimitiveList, 2121
 - activemq::util::PrimitiveMap, 2133
 - activemq::util::PrimitiveValueNode, 2156
 - cms::MapMessage, 1787
- setByteArray
 - activemq::util::PrimitiveList, 2122
 - activemq::util::PrimitiveMap, 2133
 - activemq::util::PrimitiveValueNode, 2156
 - decaf::io::BlockingByteArrayInput-Stream, 537
 - decaf::io::ByteArrayInputStream, 685, 686
- setByteProperty
 - activemq::commands::ActiveMQ-MessageTemplate, 299
 - activemq::wireformat::openwire-::utils::MessageProperty-Interceptor, 1937
 - cms::Message, 1857
- setBytes
 - activemq::commands::ActiveMQ-MapMessage, 280
 - cms::MapMessage, 1788
- setCMSCorrelationID
 - activemq::commands::ActiveMQ-MessageTemplate, 299
 - cms::Message, 1858
- setCMSDeliveryMode
 - activemq::commands::ActiveMQ-MessageTemplate, 299
 - cms::Message, 1859
- setCMSDestination
 - activemq::commands::ActiveMQ-MessageTemplate, 299
 - cms::Message, 1859
- setCMSExpiration
 - activemq::commands::ActiveMQ-MessageTemplate, 300
 - cms::Message, 1860
- setCMSMessageID
 - activemq::commands::ActiveMQ-MessageTemplate, 300
 - cms::Message, 1860
- setCMSPriority
 - activemq::commands::ActiveMQ-MessageTemplate, 300
 - cms::Message, 1860
- setCMSRedelivered
 - activemq::commands::ActiveMQ-MessageTemplate, 300

- cms::Message, 1861
- setCMSReplyTo
 - activemq::commands::ActiveMQ-MessageTemplate, 300
 - cms::Message, 1861
- setCMSTimestamp
 - activemq::commands::ActiveMQ-MessageTemplate, 300
 - cms::Message, 1862
- setCMSType
 - activemq::commands::ActiveMQ-MessageTemplate, 300
 - cms::Message, 1863
- setCacheEnabled
 - activemq::commands::WireFormat-Info, 2896
 - activemq::wireformat::openwire::OpenWireFormat, 2054
- setCacheSize
 - activemq::commands::WireFormat-Info, 2896
 - activemq::wireformat::openwire::OpenWireFormat, 2054
- setCause
 - activemq::commands::BrokerError, 561
- setChar
 - activemq::commands::ActiveMQ-MapMessage, 280
 - activemq::util::PrimitiveList, 2122
 - activemq::util::PrimitiveMap, 2133
 - activemq::util::PrimitiveValueNode, 2157
 - cms::MapMessage, 1788
- setCipherSuites
 - decaf::net::ssl::SSLParameters, 2503
- setClientID
 - activemq::core::ActiveMQConnection, 207
 - cms::Connection, 937
- setClientId
 - activemq::commands::Connection-Info, 972
 - activemq::commands::JournalTopic-Ack, 1571
 - activemq::commands::Remove-SubscriptionInfo, 2278
 - activemq::commands::Subscription-Info, 2634
 - activemq::core::ActiveMQConnection-Factory, 223
- setClientMaster
 - activemq::commands::Connection-Info, 972
- setClose
 - activemq::commands::Connection-Control, 942
 - activemq::commands::Consumer-Control, 995
- setCloseTimeout
 - activemq::core::ActiveMQConnection, 208
 - activemq::core::ActiveMQConnection-Factory, 223
- setClosed
 - activemq::transport::failover::Backup-Transport, 488
- setCluster
 - activemq::commands::Message, 1834
- setCollisionAvoidancePercent
 - activemq::core::policies::Default-RedeliveryPolicy, 1153
 - activemq::core::RedeliveryPolicy, 2254
- setCommand
 - activemq::commands::Control-Command, 1024
 - activemq::wireformat::stomp::Stomp-Frame, 2592
- setCommandId
 - activemq::commands::BaseCommand, 497
 - activemq::commands::Command, 870
 - activemq::commands::Partial-Command, 2078
- setComponents
 - activemq::util::CompositeData, 891
- setCompressed
 - activemq::commands::Message, 1834
- setCompressionLevel
 - activemq::core::ActiveMQConnection, 208
 - activemq::core::ActiveMQConnection-Factory, 223
- setConnectedBrokers

- activemq::commands::Connection-
Control, 942
- setConnection
 - activemq::commands::ActiveMQ-
TempDestination, 381
 - activemq::commands::Message,
1834
- setConnectionFactory
 - activemq::cmsutil::CmsAccessor,
823
- setConnectionId
 - activemq::commands::BrokerInfo,
576
 - activemq::commands::Connection-
Error, 950
 - activemq::commands::Connection-
Info, 972
 - activemq::commands::ConsumerId,
1004
 - activemq::commands::Destination-
Info, 1216
 - activemq::commands::LocalTransaction-
Id, 1677
 - activemq::commands::ProducerId,
2185
 - activemq::commands::Remove-
SubscriptionInfo, 2278
 - activemq::commands::SessionId,
2381
 - activemq::commands::Transaction-
Info, 2777
- setConnectionInterruptProcessing-
Complete
 - activemq::state::ConnectionState,
984
 - activemq::transport::failover::Failover-
Transport, 1315
- setConsumerId
 - activemq::commands::Consumer-
Control, 995
 - activemq::commands::Consumer-
Info, 1015
 - activemq::commands::MessageAck,
1871
 - activemq::commands::Message-
Dispatch, 1884
 - activemq::commands::Message-
DispatchNotification, 1898
 - activemq::commands::MessagePull,
1942
- setContent
 - activemq::commands::Message,
1834
- setCorePoolSize
 - decaf::util::concurrent::ThreadPool-
Executor, 2729
- setCorrelationId
 - activemq::commands::Message,
1834
 - activemq::commands::MessagePull,
1942
 - activemq::commands::Response,
2300
- setDaemon
 - decaf::lang::Thread, 2711
- setData
 - activemq::commands::DataArray-
Response, 1071
 - activemq::commands::DataResponse,
1114
 - activemq::commands::Partial-
Command, 2078
 - decaf::net::DatagramPacket, 1083,
1084
- setDataStructure
 - activemq::commands::Message,
1834
- setDefault
 - decaf::net::ssl::SSLContext, 2498
- setDefaultClientId
 - activemq::core::ActiveMQConnection,
208
- setDefaultDestination
 - activemq::cmsutil::CmsTemplate,
847
- setDefaultDestinationName
 - activemq::cmsutil::CmsTemplate,
847
- setDeletedByBroker
 - activemq::commands::ActiveMQ-
BlobMessage, 160
- setDeliveryMode
 - activemq::cmsutil::CachedProducer,
744
 - activemq::cmsutil::CmsTemplate,
848
 - activemq::core::ActiveMQProducer,
314
 - cms::MessageProducer, 1930
- setDeliveryPersistent

- activemq::cmsutil::CmsTemplate, 848
- setDeliverySequenceId
 - activemq::commands::MessageDispatchNotification, 1898
- setDestination
 - activemq::commands::ConsumerControl, 995
 - activemq::commands::ConsumerInfo, 1015
 - activemq::commands::DestinationInfo, 1216
 - activemq::commands::JournalQueueAck, 1563
 - activemq::commands::JournalTopicAck, 1571
 - activemq::commands::Message, 1835
 - activemq::commands::MessageAck, 1871
 - activemq::commands::MessageDispatch, 1884
 - activemq::commands::MessageDispatchNotification, 1898
 - activemq::commands::MessagePull, 1942
 - activemq::commands::ProducerInfo, 2193
 - activemq::commands::SubscriptionInfo, 2634
- setDestinationResolver
 - activemq::cmsutil::CmsDestinationAccessor, 825
- setDictionary
 - decaf::util::zip::Deflater, 1185, 1186
 - decaf::util::zip::Inflater, 1453, 1454
- setDisableMessageID
 - activemq::cmsutil::CachedProducer, 744
 - activemq::core::ActiveMQProducer, 314
 - cms::MessageProducer, 1931
- setDisableMessageTimeStamp
 - activemq::cmsutil::CachedProducer, 744
 - activemq::core::ActiveMQProducer, 315
 - cms::MessageProducer, 1931
- setDispatchAsync
 - activemq::commands::ConsumerInfo, 1015
 - activemq::commands::ProducerInfo, 2193
 - activemq::core::ActiveMQConnection, 208
 - activemq::core::ActiveMQConnectionFactory, 224
- setDouble
 - activemq::commands::ActiveMQMapMessage, 281
 - activemq::util::PrimitiveList, 2123
 - activemq::util::PrimitiveMap, 2133
 - activemq::util::PrimitiveValueNode, 2157
 - cms::MapMessage, 1789
- setDoubleProperty
 - activemq::commands::ActiveMQMessageTemplate, 300
 - activemq::wireformat::openwire::utils::MessagePropertyInterceptor, 1938
 - cms::Message, 1864
- setDroppable
 - activemq::commands::Message, 1835
- setDuplexConnection
 - activemq::commands::BrokerInfo, 576
- setDurableTopicPrefetch
 - activemq::core::policies::DefaultPrefetchPolicy, 1148
 - activemq::core::PrefetchPolicy, 2113
- setEnabled
 - activemq::transport::failover::BackupTransportPool, 491
- setEnabledCipherSuites
 - decaf::internal::net::ssl::openssl::OpenSSLParameters, 2003
 - decaf::internal::net::ssl::openssl::OpenSSLServerSocket, 2008
 - decaf::internal::net::ssl::openssl::OpenSSLSocket, 2025
 - decaf::net::ssl::SSLServerSocket, 2509
 - decaf::net::ssl::SSLSocket, 2519
- setEnabledProtocols
 - decaf::internal::net::ssl::openssl::OpenSSLParameters, 2003

- decaf::internal::net::ssl::openssl::-
OpenSSLServerSocket, 2008
- decaf::internal::net::ssl::openssl::-
OpenSSLSocket, 2025
- decaf::net::ssl::SSLServerSocket,
2509
- decaf::net::ssl::SSLSocket, 2519
- setErrorCode
cms::XAException, 2924
- setErrorManager
decaf::util::logging::Handler, 1404
- setException
activemq::commands::Connection-
Error, 950
- activemq::commands::Exception-
Response, 1290
- setExceptionClass
activemq::commands::BrokerError,
562
- setExceptionListener
activemq::core::ActiveMQConnection,
209
- activemq::core::ActiveMQConnection-
Factory, 224
- cms::Connection, 938
- setExclusive
activemq::commands::ActiveMQ-
Destination, 254
- activemq::commands::Consumer-
Info, 1015
- setExclusiveOwnerThread
decaf::util::concurrent::locks::-
AbstractOwnableSynchronizer,
136
- setExit
activemq::commands::Connection-
Control, 942
- setExpiration
activemq::commands::Message,
1835
- setExplicitQosEnabled
activemq::cmsutil::CmsTemplate,
848
- setFailOnClose
activemq::transport::mock::Mock-
Transport, 1956
- setFailOnKeepAliveSends
activemq::transport::mock::Mock-
Transport, 1956
- setFailOnReceiveMessage
activemq::transport::mock::Mock-
Transport, 1956
- setFailOnSendMessage
activemq::transport::mock::Mock-
Transport, 1956
- setFailOnStart
activemq::transport::mock::Mock-
Transport, 1956
- setFailOnStop
activemq::transport::mock::Mock-
Transport, 1956
- setFailoverReconnect
activemq::commands::Connection-
Info, 972
- setFailureError
activemq::core::ActiveMQConsumer,
242
- setFaultTolerant
activemq::commands::Connection-
Control, 942
- activemq::commands::Connection-
Info, 972
- setFaultTolerantConfiguration
activemq::commands::BrokerInfo,
576
- setFilter
decaf::util::logging::Handler, 1405
- decaf::util::logging::Logger, 1703
- setFirstMessageId
activemq::commands::MessageAck,
1871
- setFirstNakNumber
activemq::commands::Replay-
Command, 2286
- setFloat
activemq::commands::ActiveMQ-
MapMessage, 281
- activemq::util::PrimitiveList, 2123
- activemq::util::PrimitiveMap, 2134
- activemq::util::PrimitiveValueNode,
2157
- cms::MapMessage, 1789
- setFloatProperty
activemq::commands::ActiveMQ-
MessageTemplate, 300
- activemq::wireformat::openwire-
::utils::MessageProperty-
Interceptor, 1938
- cms::Message, 1864
- setFlush

- activemq::commands::Consumer-
Control, 995
- setFormatId
 - activemq::commands::XATransaction-
Id, 2941
- setFormatter
 - decaf::util::logging::Handler, 1405
- setFragment
 - activemq::util::CompositeData, 891
 - decaf::internal::net::URIType, 2865
- setGlobalTransactionId
 - activemq::commands::XATransaction-
Id, 2941
- setGroupId
 - activemq::commands::Message,
1835
- setGroupSequence
 - activemq::commands::Message,
1835
- setHost
 - activemq::util::CompositeData, 891
 - decaf::internal::net::URIType, 2865
- setInitialDelayTime
 - activemq::transport::inactivity::-
InactivityMonitor, 1428
- setInitialReconnectDelay
 - activemq::transport::failover::Failover-
Transport, 1316
- setInitialRedeliveryDelay
 - activemq::core::policies::Default-
RedeliveryPolicy, 1153
 - activemq::core::RedeliveryPolicy,
2255
- setInitialized
 - activemq::transport::failover::Failover-
Transport, 1316
- setInput
 - decaf::util::zip::Deflater, 1186, 1187
 - decaf::util::zip::Inflater, 1455
- setInputStream
 - activemq::transport::IOTransport,
1554
- setInt
 - activemq::commands::ActiveMQ-
MapMessage, 281
 - activemq::util::PrimitiveList, 2123
 - activemq::util::PrimitiveMap, 2134
 - activemq::util::PrimitiveValueNode,
2157
 - cms::MapMessage, 1789
- setIntProperty
 - activemq::commands::ActiveMQ-
MessageTemplate, 300
 - activemq::wireformat::openwire-
::utils::MessageProperty-
Interceptor, 1938
 - cms::Message, 1865
- setKeepAlive
 - decaf::net::Socket, 2465
- setKeepAliveResponseRequired
 - activemq::transport::inactivity::-
InactivityMonitor, 1428
- setKeepAliveTime
 - decaf::util::concurrent::ThreadPool-
Executor, 2729
- setLastDeliveredSequenceId
 - activemq::commands::RemoveInfo,
2270
 - activemq::core::ActiveMQConsumer,
243
 - activemq::core::ActiveMQSession,
352
- setLastMessageId
 - activemq::commands::MessageAck,
1871
- setLastNakNumber
 - activemq::commands::Replay-
Command, 2286
- setLength
 - decaf::net::DatagramPacket, 1084
- setLevel
 - decaf::util::logging::Handler, 1405
 - decaf::util::logging::Logger, 1704
 - decaf::util::logging::LogRecord, 1722
 - decaf::util::zip::Deflater, 1188
- setLimit
 - activemq::util::MemoryUsage, 1820
- setList
 - activemq::util::PrimitiveValueNode,
2157
- setLoggerName
 - decaf::util::logging::LogRecord, 1722
- setLong
 - activemq::commands::ActiveMQ-
MapMessage, 282
 - activemq::util::PrimitiveList, 2124
 - activemq::util::PrimitiveMap, 2134
 - activemq::util::PrimitiveValueNode,
2158
 - cms::MapMessage, 1790

- setLongProperty
 - activemq::commands::ActiveMQ-MessageTemplate, 300
 - activemq::wireformat::openwire::utils::MessageProperty-Interceptor, 1938
 - cms::Message, 1865
- setMagic
 - activemq::commands::WireFormat-Info, 2897
- setManageable
 - activemq::commands::Connection-Info, 972
- setManaged
 - decaf::internal::util::GenericResource, 1398
- setMap
 - activemq::util::PrimitiveValueNode, 2158
- setMark
 - cms::CMSException, 828
 - decaf::lang::Exception, 1285
 - decaf::lang::Throwable, 2736
- setMarshaledForm
 - activemq::commands::BaseData-Structure, 530
 - activemq::wireformat::MarshalAware, 1795
- setMarshaledProperties
 - activemq::commands::Message, 1835
 - activemq::commands::WireFormat-Info, 2897
- setMasterBroker
 - activemq::commands::BrokerInfo, 576
- setMaxCacheSize
 - activemq::state::ConnectionState-Tracker, 989
 - activemq::transport::failover::Failover-Transport, 1316
- setMaxInactivityDuration
 - activemq::commands::WireFormat-Info, 2897
 - activemq::wireformat::openwire::OpenWireFormat, 2054
- setMaxInactivityDurationInitialDelay
 - activemq::commands::WireFormat-Info, 2897
- setMaxInactivityDurationInitialDelay
 - activemq::wireformat::openwire::OpenWireFormat, 2055
- setMaxReconnectAttempts
 - activemq::transport::failover::Failover-Transport, 1316
- setMaxReconnectDelay
 - activemq::transport::failover::Failover-Transport, 1316
- setMaximumPendingMessageLimit
 - activemq::commands::Consumer-Info, 1015
- setMaximumPoolSize
 - decaf::util::concurrent::ThreadPool-Executor, 2730
- setMaximumRedeliveries
 - activemq::core::policies::Default-RedeliveryPolicy, 1154
 - activemq::core::RedeliveryPolicy, 2255
- setMessage
 - activemq::commands::BrokerError, 562
 - activemq::commands::JournalTrace, 1579
 - activemq::commands::Message-Dispatch, 1885
 - decaf::lang::Exception, 1285
 - decaf::util::logging::LogRecord, 1722
- setMessageAck
 - activemq::commands::Journal-QueueAck, 1563
- setMessageCount
 - activemq::commands::MessageAck, 1871
- setMessageId
 - activemq::commands::JournalTopic-Ack, 1571
 - activemq::commands::Message, 1835
 - activemq::commands::Message-DispatchNotification, 1898
 - activemq::commands::MessagePull, 1943
- setMessageIdEnabled
 - activemq::cmsutil::CmsTemplate, 849
- setMessageListener
 - activemq::cmsutil::CachedConsumer, 737

- activemq::core::ActiveMQConsumer, 243
- cms::MessageConsumer, 1880
- setMessagePrioritySupported
 - activemq::core::ActiveMQConnection, 209
 - activemq::core::ActiveMQConnection-Factory, 224
- setMessageSequenced
 - activemq::commands::JournalTopic-Ack, 1571
- setMessageTimestampEnabled
 - activemq::cmsutil::CmsTemplate, 849
- setMimeType
 - activemq::commands::ActiveMQ-BlobMessage, 160
- setName
 - activemq::commands::ActiveMQ-BlobMessage, 160
 - activemq::transport::mock::Mock-Transport, 1956
 - decaf::lang::Thread, 2711
- setNeedClientAuth
 - decaf::internal::net::ssl::openssl::OpenSSLParameters, 2003
 - decaf::internal::net::ssl::openssl::OpenSSLServerSocket, 2008
 - decaf::internal::net::ssl::openssl::OpenSSLSocket, 2025
 - decaf::net::ssl::SSLParameters, 2504
 - decaf::net::ssl::SSLServerSocket, 2510
 - decaf::net::ssl::SSLSocket, 2520
- setNetworkBrokerId
 - activemq::commands::Network-BridgeFilter, 1973
- setNetworkConnection
 - activemq::commands::BrokerInfo, 576
- setNetworkConsumerPath
 - activemq::commands::Consumer-Info, 1015
- setNetworkProperties
 - activemq::commands::BrokerInfo, 576
- setNetworkSubscription
 - activemq::commands::Consumer-Info, 1015
- setNetworkTTL
 - activemq::commands::Network-BridgeFilter, 1973
- setNoLocal
 - activemq::cmsutil::CmsTemplate, 849
 - activemq::commands::Consumer-Info, 1015
- setNoRangeAcks
 - activemq::commands::Consumer-Info, 1015
- setNumReceivedMessageBeforeFail
 - activemq::transport::mock::Mock-Transport, 1956
- setNumReceivedMessages
 - activemq::transport::mock::Mock-Transport, 1956
- setNumSentKeepAlives
 - activemq::transport::mock::Mock-Transport, 1956
- setNumSentKeepAlivesBeforeFail
 - activemq::transport::mock::Mock-Transport, 1956
- setNumSentMessageBeforeFail
 - activemq::transport::mock::Mock-Transport, 1956
- setNumSentMessages
 - activemq::transport::mock::Mock-Transport, 1956
- setOOBInline
 - decaf::internal::net::ssl::openssl::OpenSSLSocket, 2026
 - decaf::net::Socket, 2465
- setObjectId
 - activemq::commands::RemoveInfo, 2270
- setOffset
 - decaf::net::DatagramPacket, 1085
- setOpaque
 - decaf::internal::net::URIType, 2866
- setOperationType
 - activemq::commands::Destination-Info, 1217
- setOptimizedAcknowledge
 - activemq::commands::Consumer-Info, 1015
- setOption
 - decaf::internal::net::tcp::TcpSocket, 2686
 - decaf::net::SocketImpl, 2486

- setOrdered
 - activemq::commands::ActiveMQ-Destination, 255
- setOrderedTarget
 - activemq::commands::ActiveMQ-Destination, 255
- setOriginalDestination
 - activemq::commands::Message, 1835
- setOriginalTransactionId
 - activemq::commands::Message, 1835
- setOutputStream
 - decaf::util::logging::StreamHandler, 2606
- setOutgoingListener
 - activemq::transport::mock::MockTransport, 1957
- setOutputStream
 - activemq::transport::IOTransport, 1555
- setParameters
 - activemq::util::CompositeData, 892
- setParent
 - decaf::util::logging::Logger, 1704
- setPassword
 - activemq::commands::Connection-Info, 972
 - activemq::core::ActiveMQConnection, 209
 - activemq::core::ActiveMQConnection-Factory, 224
- setPath
 - activemq::util::CompositeData, 892
 - decaf::internal::net::URIType, 2866
- setPeerBrokerInfos
 - activemq::commands::BrokerInfo, 577
- setPersistent
 - activemq::commands::Message, 1835
- setPhysicalName
 - activemq::commands::ActiveMQ-Destination, 255
- setPort
 - decaf::internal::net::URIType, 2866
 - decaf::net::DatagramPacket, 1085
- setPreferredWireFormatInfo
 - activemq::wireformat::openwire::OpenWireFormat, 2055
- setPrefetch
 - activemq::commands::Consumer-Control, 995
- setPrefetchPolicy
 - activemq::core::ActiveMQConnection, 209
 - activemq::core::ActiveMQConnection-Factory, 225
- setPrefetchSize
 - activemq::commands::Consumer-Info, 1015
- setPrepared
 - activemq::state::TransactionState, 2786
- setPreparedResult
 - activemq::state::TransactionState, 2786
- setPriority
 - activemq::cmsutil::CachedProducer, 745
 - activemq::cmsutil::CmsTemplate, 849
 - activemq::commands::Consumer-Info, 1015
 - activemq::commands::Message, 1835
 - activemq::core::ActiveMQProducer, 315
 - cms::MessageProducer, 1932
 - decaf::lang::Thread, 2711
- setProducerId
 - activemq::commands::Message, 1835
 - activemq::commands::MessageId, 1911
 - activemq::commands::ProducerAck, 2174
 - activemq::commands::ProducerInfo, 2193
- setProducerSequenceId
 - activemq::commands::MessageId, 1911
- setProducerSessionKey
 - activemq::commands::ProducerId, 2185
- setProducerWindowSize
 - activemq::core::ActiveMQConnection, 210
 - activemq::core::ActiveMQConnection-Factory, 225

- setProperties
 - activemq::commands::WireFormat-Info, 2898
 - activemq::util::ActiveMQProperties, 320
 - decaf::util::logging::LogManager, 1718
- setProperty
 - activemq::util::ActiveMQProperties, 320
 - activemq::wireformat::stomp::Stomp-Frame, 2592
 - cms::CMSProperties, 834
 - decaf::lang::System, 2675
 - decaf::util::Properties, 2207
- setProtocols
 - decaf::net::ssl::SSLParameters, 2504
- setPubSubDomain
 - activemq::cmsutil::CmsDestination-Accessor, 825
 - activemq::cmsutil::CmsTemplate, 849
- setQuery
 - decaf::internal::net::URIType, 2866
- setQueueBrowserPrefetch
 - activemq::core::policies::Default-PrefetchPolicy, 1148
 - activemq::core::PrefetchPolicy, 2113
- setQueuePrefetch
 - activemq::core::policies::Default-PrefetchPolicy, 1149
 - activemq::core::PrefetchPolicy, 2113
- setRandomize
 - activemq::transport::failover::Failover-Transport, 1316
 - activemq::transport::failover::URI-Pool, 2853
- setReadCheckTime
 - activemq::transport::inactivity::InactivityMonitor, 1428
- setReadOnly
 - decaf::internal::nio::ByteBuffer, 679
 - decaf::internal::nio::CharArrayBuffer, 783
 - decaf::internal::nio::DoubleArray-Buffer, 1258
 - decaf::internal::nio::FloatArrayBuffer, 1367
 - decaf::internal::nio::IntArrayBuffer, 1487
 - decaf::internal::nio::LongArrayBuffer, 1752
 - decaf::internal::nio::ShortArray-Buffer, 2418
- setReadOnlyBody
 - activemq::commands::Message, 1835
- setReadOnlyProperties
 - activemq::commands::Message, 1835
- setRebalanceConnection
 - activemq::commands::Connection-Control, 942
- setReceiveBufferSize
 - decaf::net::ServerSocket, 2350
 - decaf::net::Socket, 2466
- setReceiveTimeout
 - activemq::cmsutil::CmsTemplate, 849
- setRecievedByDFBridge
 - activemq::commands::Message, 1836
- setReconnectDelay
 - activemq::transport::failover::Failover-Transport, 1316
- setReconnectSupported
 - activemq::transport::failover::Failover-Transport, 1316
- setReconnectTo
 - activemq::commands::Connection-Control, 942
- setRedeliveryCounter
 - activemq::commands::Message, 1836
 - activemq::commands::Message-Dispatch, 1885
- setRedeliveryDelay
 - activemq::core::policies::Default-RedeliveryPolicy, 1154
 - activemq::core::RedeliveryPolicy, 2255
- setRedeliveryPolicy
 - activemq::core::ActiveMQConnection, 210
 - activemq::core::ActiveMQConnection-Factory, 225
 - activemq::core::ActiveMQConsumer, 243

- setRejectedExecutionHandler
 - decaf::util::concurrent::ThreadPool-Executor, 2730
- setRemoteBlobUrl
 - activemq::commands::ActiveMQ-BlobMessage, 161
- setReplyTo
 - activemq::commands::Message, 1836
- setResponse
 - activemq::transport::correlator::FutureResponse, 1394
- setResponseBuilder
 - activemq::transport::mock::Internal-CommandListener, 1528
 - activemq::transport::mock::Mock-Transport, 1957
- setResponseRequired
 - activemq::commands::BaseCommand, 498
 - activemq::commands::Command, 870
- setRestoreConsumers
 - activemq::state::ConnectionState-Tracker, 989
- setRestoreProducers
 - activemq::state::ConnectionState-Tracker, 989
- setRestoreSessions
 - activemq::state::ConnectionState-Tracker, 989
- setRestoreTransaction
 - activemq::state::ConnectionState-Tracker, 989
- setResult
 - activemq::commands::Integer-Response, 1518
- setResume
 - activemq::commands::Connection-Control, 942
- setRetroactive
 - activemq::commands::Consumer-Info, 1015
- setReuseAddress
 - decaf::net::ServerSocket, 2350
 - decaf::net::Socket, 2466
- setSSLParameters
 - decaf::net::ssl::SSLSocket, 2520
- setScheduledTime
 - decaf::util::TimerTask, 2753
- setScheme
 - activemq::util::CompositeData, 892
 - decaf::internal::net::URIType, 2866
- setSchemeSpecificPart
 - decaf::internal::net::URIType, 2867
- setSeed
 - decaf::security::SecureRandom, 2324, 2325
 - decaf::util::Random, 2234
- setSelector
 - activemq::commands::Consumer-Info, 1015
 - activemq::commands::Subscription-Info, 2634
- setSendBufferSize
 - decaf::net::Socket, 2466
- setSendTimeout
 - activemq::core::ActiveMQConnection, 210
 - activemq::core::ActiveMQConnection-Factory, 225
 - activemq::core::ActiveMQProducer, 315
- setServerAuthority
 - decaf::internal::net::URIType, 2867
- setServiceName
 - activemq::commands::Discovery-Event, 1229
- setSessionAcknowledgeMode
 - activemq::cmsutil::CmsAccessor, 823
- setSessionId
 - activemq::commands::ConsumerId, 1004
 - activemq::commands::ProducerId, 2185
 - activemq::commands::SessionInfo, 2389
- setShort
 - activemq::commands::ActiveMQ-MapMessage, 282
 - activemq::util::PrimitiveList, 2124
 - activemq::util::PrimitiveMap, 2134
 - activemq::util::PrimitiveValueNode, 2158
 - cms::MapMessage, 1790
- setShortProperty
 - activemq::commands::ActiveMQ-MessageTemplate, 300

- activemq::wireformat::openwire-
::utils::MessageProperty-
Interceptor, 1939
- cms::Message, 1866
- setSize
 - activemq::commands::ProducerAck,
2174
- setSizePrefixDisabled
 - activemq::commands::WireFormat-
Info, 2898
 - activemq::wireformat::openwire::-
OpenWireFormat, 2055
- setSlaveBroker
 - activemq::commands::BrokerInfo,
577
- setSoLinger
 - decaf::net::Socket, 2467
- setSoTimeout
 - decaf::net::ServerSocket, 2351
 - decaf::net::Socket, 2467
- setSocketAddress
 - decaf::net::DatagramPacket, 1085
- setSocketImplFactory
 - decaf::net::ServerSocket, 2351
 - decaf::net::Socket, 2467
- setSource
 - decaf::internal::net::URIType, 2867
- setSourceFile
 - decaf::util::logging::LogRecord, 1722
- setSourceFunction
 - decaf::util::logging::LogRecord, 1723
- setSourceLine
 - decaf::util::logging::LogRecord, 1723
- setStackTrace
 - decaf::lang::Exception, 1286
- setStackTraceElements
 - activemq::commands::BrokerError,
562
- setStackTraceEnabled
 - activemq::commands::WireFormat-
Info, 2898
 - activemq::wireformat::openwire::-
OpenWireFormat, 2055
- setStart
 - activemq::commands::Consumer-
Control, 995
- setStartupMaxReconnectAttempts
 - activemq::transport::failover::Failover-
Transport, 1316
- setStop
 - activemq::commands::Consumer-
Control, 995
- setStrategy
 - decaf::util::zip::Deflater, 1188
- setString
 - activemq::commands::ActiveMQ-
MapMessage, 283
 - activemq::util::PrimitiveList, 2124
 - activemq::util::PrimitiveMap, 2135
 - activemq::util::PrimitiveValueNode,
2158
 - cms::MapMessage, 1791
- setStringProperty
 - activemq::commands::ActiveMQ-
MessageTemplate, 301
 - activemq::wireformat::openwire-
::utils::MessageProperty-
Interceptor, 1939
 - cms::Message, 1866
- setSubscriptionName
 - activemq::commands::Remove-
SubscriptionInfo, 2278
 - activemq::commands::Subscription-
Info, 2634
- setSubscribedDestination
 - activemq::commands::Subscription-
Info, 2634
- setSubscriptionName
 - activemq::commands::Consumer-
Info, 1015
- setSubscriptionName
 - activemq::commands::JournalTopic-
Ack, 1571
- setSuspend
 - activemq::commands::Connection-
Control, 942
- setSynchronizationRegistered
 - activemq::core::ActiveMQConsumer,
243
- setTargetConsumerId
 - activemq::commands::Message,
1836
- setTcpNoDelay
 - decaf::net::Socket, 2468
- setTcpNoDelayEnabled
 - activemq::commands::WireFormat-
Info, 2898
 - activemq::wireformat::openwire::-
OpenWireFormat, 2056
- setText

- activemq::commands::ActiveMQ-
 TextMessage, 409
- cms::TextMessage, 2702, 2703
- setTextView
 - activemq::commands::MessageId,
 1911
- setThreadFactory
 - decaf::util::concurrent::ThreadPool-
 Executor, 2730
- setThrown
 - decaf::util::logging::LogRecord, 1723
- setTightEncodingEnabled
 - activemq::commands::WireFormat-
 Info, 2898
 - activemq::wireformat::openwire::-
 OpenWireFormat, 2056
- setTime
 - decaf::util::Date, 1142
- setTimeToLive
 - activemq::cmsutil::CachedProducer,
 745
 - activemq::cmsutil::CmsTemplate,
 850
 - activemq::core::ActiveMQProducer,
 315
 - cms::MessageProducer, 1932
- setTimeout
 - activemq::commands::Destination-
 Info, 1217
 - activemq::commands::MessagePull,
 1943
 - activemq::transport::failover::Failover-
 Transport, 1316
- setTimestamp
 - activemq::commands::Message,
 1836
 - decaf::util::logging::LogRecord, 1723
- setTopicPrefetch
 - activemq::core::policies::Default-
 PrefetchPolicy, 1149
 - activemq::core::PrefetchPolicy, 2114
- setTrackMessages
 - activemq::state::ConnectionState-
 Tracker, 989
 - activemq::transport::failover::Failover-
 Transport, 1316
- setTrackTransactionProducers
 - activemq::state::ConnectionState-
 Tracker, 989
- activemq::transport::failover::Failover-
 Transport, 1316
- setTrackTransactions
 - activemq::state::ConnectionState-
 Tracker, 990
- setTrafficClass
 - decaf::net::Socket, 2468
- setTransactionId
 - activemq::commands::JournalTopic-
 Ack, 1571
 - activemq::commands::Journal-
 Transaction, 1586
 - activemq::commands::Message,
 1836
 - activemq::commands::MessageAck,
 1871
 - activemq::commands::Transaction-
 Info, 2777
- setTransactionState
 - activemq::state::ProducerState, 2199
- setTransactionTimeout
 - activemq::core::ActiveMQTransaction-
 Context, 431
 - cms::XAResource, 2932
- setTransport
 - activemq::transport::failover::Backup-
 Transport, 488
 - activemq::transport::mock::Internal-
 CommandListener, 1528
- setTransportInterruptionProcessing-
 Complete
 - activemq::core::ActiveMQConnection,
 211
- setTransportListener
 - activemq::transport::failover::Failover-
 Transport, 1316
 - activemq::transport::IOTransport,
 1555
 - activemq::transport::mock::Mock-
 Transport, 1957
 - activemq::transport::Transport, 2796
 - activemq::transport::TransportFilter,
 2808
- setTreadId
 - decaf::util::logging::LogRecord, 1724
- setType
 - activemq::commands::Journal-
 Transaction, 1586
 - activemq::commands::Message,
 1836

- activemq::commands::Transaction-Info, 2777
- setUncaughtExceptionHandler
 - decaf::lang::Thread, 2711
- setUpdateURLsSupported
 - activemq::transport::failover::Failover-Transport, 1317
- setUri
 - activemq::transport::failover::Backup-Transport, 488
- setUsage
 - activemq::util::MemoryUsage, 1821
- setUseAsyncSend
 - activemq::core::ActiveMQConnection, 211
 - activemq::core::ActiveMQConnection-Factory, 226
- setUseClientMode
 - decaf::internal::net::ssl::openssl::-OpenSSLParameters, 2003
 - decaf::internal::net::ssl::openssl::-OpenSSLSocket, 2026
 - decaf::net::ssl::SSLSocket, 2520
- setUseCollisionAvoidance
 - activemq::core::policies::Default-RedeliveryPolicy, 1154
 - activemq::core::RedeliveryPolicy, 2255
- setUseCompression
 - activemq::core::ActiveMQConnection, 211
 - activemq::core::ActiveMQConnection-Factory, 226
- setUseExponentialBackOff
 - activemq::core::policies::Default-RedeliveryPolicy, 1154
 - activemq::core::RedeliveryPolicy, 2256
 - activemq::transport::failover::Failover-Transport, 1317
- setUseParentHandlers
 - decaf::util::logging::Logger, 1704
- setUserID
 - activemq::commands::Message, 1836
- setUserInfo
 - decaf::internal::net::URIType, 2867
- setUserName
 - activemq::commands::Connection-Info, 972
- setUsername
 - activemq::core::ActiveMQConnection, 211
 - activemq::core::ActiveMQConnection-Factory, 226
- setValid
 - decaf::internal::net::URIType, 2868
- setValue
 - activemq::commands::BrokerId, 567
 - activemq::commands::ConnectionId, 963
 - activemq::commands::ConsumerId, 1004
 - activemq::commands::LocalTransaction-Id, 1677
 - activemq::commands::MessageId, 1911
 - activemq::commands::ProducerId, 2185
 - activemq::commands::SessionId, 2381
 - activemq::util::PrimitiveValueNode, 2159
 - decaf::util::Map::Entry, 1274
- setVersion
 - activemq::commands::WireFormat-Info, 2899
 - activemq::wireformat::openwire::-OpenWireFormat, 2056
 - activemq::wireformat::stomp::Stomp-WireFormat, 2600
 - activemq::wireformat::WireFormat, 2887
- setWantClientAuth
 - decaf::internal::net::ssl::openssl::-OpenSSLParameters, 2003
 - decaf::internal::net::ssl::openssl::-OpenSSLServerSocket, 2009
 - decaf::internal::net::ssl::openssl::-OpenSSLSocket, 2027
 - decaf::net::ssl::SSLParameters, 2504
 - decaf::net::ssl::SSLServerSocket, 2510
 - decaf::net::ssl::SSLSocket, 2521
- setWasPrepared
 - activemq::commands::Journal-Transaction, 1586
- setWindowSize

- activemq::commands::ProducerInfo, 2193
- setWireFormat
 - activemq::transport::failover::FailoverTransport, 1317
 - activemq::transport::IOTransport, 1555
 - activemq::transport::mock::MockTransport, 1957
 - activemq::transport::Transport, 2797
 - activemq::transport::TransportFilter, 2808
- setWriteCheckTime
 - activemq::transport::inactivity::InactivityMonitor, 1428
- setenv
 - decaf::lang::System, 2674
- setupSocketImpl
 - decaf::net::ServerSocket, 2351
- severe
 - decaf::util::logging::Logger, 1704
- shortValue
 - activemq::util::PrimitiveValueNode::PrimitiveValue, 2143
 - decaf::lang::Byte, 621
 - decaf::lang::Character, 772
 - decaf::lang::Double, 1245
 - decaf::lang::Float, 1354
 - decaf::lang::Integer, 1511
 - decaf::lang::Long, 1737
 - decaf::lang::Number, 1994
 - decaf::lang::Short, 2406
- shutdown
 - activemq::state::ConnectionState, 984
 - activemq::state::SessionState, 2397
 - activemq::state::TransactionState, 2786
 - activemq::threads::CompositeTaskRunner, 895
 - activemq::threads::DedicatedTaskRunner, 1145
 - activemq::threads::Scheduler, 2319
 - activemq::threads::TaskRunner, 2677, 2678
 - decaf::util::concurrent::ExecutorService, 1304
 - decaf::util::concurrent::ThreadPoolExecutor, 2731
- shutdownInput
 - decaf::internal::net::ssl::openssl::OpenSSLSocket, 2027
 - decaf::internal::net::tcp::TcpSocket, 2687
 - decaf::net::Socket, 2469
 - decaf::net::SocketImpl, 2486
- shutdownLibrary
 - activemq::library::ActiveMQCPP, 246
- shutdownNetworking
 - decaf::internal::net::Network, 1971
- shutdownNow
 - decaf::util::concurrent::ExecutorService, 1304
 - decaf::util::concurrent::ThreadPoolExecutor, 2731
- shutdownOutput
 - decaf::internal::net::ssl::openssl::OpenSSLSocket, 2027
 - decaf::internal::net::tcp::TcpSocket, 2687
 - decaf::net::Socket, 2469
 - decaf::net::SocketImpl, 2486
- shutdownRuntime
 - decaf::lang::Runtime, 2314
- signal
 - decaf::util::concurrent::locks::Condition, 927
- signalAll
 - decaf::util::concurrent::locks::Condition, 927
- signalInterruptProcessingComplete
 - activemq::core::ActiveMQConnection, 211
- signum
 - decaf::lang::Integer, 1512
 - decaf::lang::Long, 1738
 - decaf::lang::Math, 1812, 1813
- size
 - activemq::commands::ProducerAck, 2174
 - activemq::core::FifoMessageDispatchChannel, 1327
 - activemq::core::MessageDispatchChannel, 1890
 - activemq::core::SimplePriorityMessageDispatchChannel, 2449
 - activemq::util::ActiveMQProperties, 320

- activemq::wireformat::openwire-
 ::utils::HexTable, 1408
- cms::CMSProperties, 834
- decaf::internal::util::TimerTaskHeap,
 2755
- decaf::io::ByteArrayOutputStream,
 689
- decaf::io::DataOutputStream, 1111
- decaf::util::ArrayList, 457
- decaf::util::Collection, 864
- decaf::util::concurrent::Concurrent-
 StlMap, 918
- decaf::util::concurrent::CopyOn-
 WriteArrayList, 1046
- decaf::util::concurrent::CopyOn-
 WriteArraySet, 1061
- decaf::util::concurrent::Linked-
 BlockingQueue, 1632
- decaf::util::concurrent::Synchronous-
 Queue, 2664
- decaf::util::LinkedList, 1658
- decaf::util::Map, 1778
- decaf::util::PriorityQueue, 2170
- decaf::util::Properties, 2207
- decaf::util::StlList, 2551
- decaf::util::StlMap, 2563
- decaf::util::StlQueue, 2571
- decaf::util::StlSet, 2581
- gz_state, 1401
- skip
 - decaf::internal::net::ssl::openssl:-
 OpenSSLSocketInputStream,
 2042
 - decaf::internal::net::tcp::TcpSocket-
 InputStream, 2690
 - decaf::io::BlockingByteArrayInput-
 Stream, 537
 - decaf::io::BufferedInputStream, 594
 - decaf::io::ByteArrayInputStream, 686
 - decaf::io::FilterInputStream, 1339
 - decaf::io::InputStream, 1472
 - decaf::io::PushbackInputStream,
 2219
 - decaf::io::Reader, 2243
 - decaf::util::zip::CheckedInputStream,
 807
 - decaf::util::zip::InflaterInputStream,
 1463
 - gz_state, 1401
- decaf::io::DataInput, 1093
- decaf::io::DataInputStream, 1102
- slaveBroker
 - activemq::commands::BrokerInfo,
 578
- sleep
 - decaf::lang::Thread, 2712
 - decaf::util::concurrent::TimeUnit,
 2759
- slice
 - decaf::internal::nio::ByteBuffer,
 679
 - decaf::internal::nio::CharArrayBuffer,
 784
 - decaf::internal::nio::DoubleArray-
 Buffer, 1258
 - decaf::internal::nio::FloatArrayBuffer,
 1367
 - decaf::internal::nio::IntArrayBuffer,
 1487
 - decaf::internal::nio::LongArrayBuffer,
 1752
 - decaf::internal::nio::ShortArray-
 Buffer, 2418
 - decaf::nio::ByteBuffer, 716
 - decaf::nio::CharBuffer, 801
 - decaf::nio::DoubleBuffer, 1270
 - decaf::nio::FloatBuffer, 1378
 - decaf::nio::IntBuffer, 1498
 - decaf::nio::LongBuffer, 1763
 - decaf::nio::ShortBuffer, 2429
- sqrt
 - decaf::lang::Math, 1814
- src/main/activemq/cmsutil/Cached-
 Consumer.h, 2957
- src/main/activemq/cmsutil/Cached-
 Producer.h, 2957
- src/main/activemq/cmsutil/CmsAccessor.-
 h, 2958
- src/main/activemq/cmsutil/CmsDestination-
 Accessor.h, 2958
- src/main/activemq/cmsutil/CmsTemplate.-
 h, 2959
- src/main/activemq/cmsutil/Destination-
 Resolver.h, 2959
- src/main/activemq/cmsutil/Dynamic-
 DestinationResolver.h, 2960
- src/main/activemq/cmsutil/Message-
 Creator.h, 2960

- src/main/activemq/cmsutil/PooledSession.h, 2961
- src/main/activemq/cmsutil/Producer-Callback.h, 2961
- src/main/activemq/cmsutil/Resource-LifecycleManager.h, 2962
- src/main/activemq/cmsutil/Session-Callback.h, 2962
- src/main/activemq/cmsutil/SessionPool.h, 2963
- src/main/activemq/commands/ActiveMQ-BlobMessage.h, 2963
- src/main/activemq/commands/ActiveMQ-BytesMessage.h, 2964
- src/main/activemq/commands/ActiveMQ-Destination.h, 2964
- src/main/activemq/commands/ActiveMQ-MapMessage.h, 2965
- src/main/activemq/commands/ActiveMQ-Message.h, 2965
- src/main/activemq/commands/ActiveMQ-MessageTemplate.h, 2966
- src/main/activemq/commands/ActiveMQ-ObjectMessage.h, 2966
- src/main/activemq/commands/ActiveMQ-Queue.h, 2967
- src/main/activemq/commands/ActiveMQ-StreamMessage.h, 2967
- src/main/activemq/commands/ActiveMQ-TempDestination.h, 2968
- src/main/activemq/commands/ActiveMQ-TempQueue.h, 2968
- src/main/activemq/commands/ActiveMQ-TempTopic.h, 2969
- src/main/activemq/commands/ActiveMQ-TextMessage.h, 2969
- src/main/activemq/commands/ActiveMQ-Topic.h, 2970
- src/main/activemq/commands/Base-Command.h, 2970
- src/main/activemq/commands/BaseData-Structure.h, 2971
- src/main/activemq/commands/Boolean-Expression.h, 2971
- src/main/activemq/commands/Broker-Error.h, 2971
- src/main/activemq/commands/BrokerId.h, 2972
- src/main/activemq/commands/Broker-Info.h, 2972
- src/main/activemq/commands/Command.h, 2973
- src/main/activemq/commands/Connection-Control.h, 2973
- src/main/activemq/commands/Connection-Error.h, 2974
- src/main/activemq/commands/Connection-Id.h, 2974
- src/main/activemq/commands/Connection-Info.h, 2974
- src/main/activemq/commands/Consumer-Control.h, 2975
- src/main/activemq/commands/Consumer-Id.h, 2975
- src/main/activemq/commands/Consumer-Info.h, 2976
- src/main/activemq/commands/Control-Command.h, 2976
- src/main/activemq/commands/DataArray-Response.h, 2977
- src/main/activemq/commands/Data-Response.h, 2977
- src/main/activemq/commands/Data-Structure.h, 2978
- src/main/activemq/commands/Destination-Info.h, 2978
- src/main/activemq/commands/Discovery-Event.h, 2978
- src/main/activemq/commands/Exception-Response.h, 2979
- src/main/activemq/commands/Flush-Command.h, 2979
- src/main/activemq/commands/Integer-Response.h, 2980
- src/main/activemq/commands/Journal-QueueAck.h, 2980
- src/main/activemq/commands/Journal-TopicAck.h, 2981
- src/main/activemq/commands/Journal-Trace.h, 2981
- src/main/activemq/commands/Journal-Transaction.h, 2982
- src/main/activemq/commands/KeepAlive-Info.h, 2982
- src/main/activemq/commands/LastPartial-Command.h, 2982
- src/main/activemq/commands/Local-TransactionId.h, 2983
- src/main/activemq/commands/Message.h, 2983

- src/main/activemq/commands/Message-Ack.h, 2984
- src/main/activemq/commands/Message-Dispatch.h, 2985
- src/main/activemq/commands/Message-DispatchNotification.h, 2985
- src/main/activemq/commands/Message-Id.h, 2986
- src/main/activemq/commands/Message-Pull.h, 2986
- src/main/activemq/commands/Network-BridgeFilter.h, 2987
- src/main/activemq/commands/Partial-Command.h, 2987
- src/main/activemq/commands/Producer-Ack.h, 2988
- src/main/activemq/commands/Producer-Id.h, 2988
- src/main/activemq/commands/Producer-Info.h, 2988
- src/main/activemq/commands/Remove-Info.h, 2989
- src/main/activemq/commands/Remove-SubscriptionInfo.h, 2989
- src/main/activemq/commands/Replay-Command.h, 2990
- src/main/activemq/commands/Response.-h, 2990
- src/main/activemq/commands/SessionId.-h, 2991
- src/main/activemq/commands/Session-Info.h, 2991
- src/main/activemq/commands/Shutdown-Info.h, 2992
- src/main/activemq/commands/Subscription-Info.h, 2992
- src/main/activemq/commands/Transaction-Id.h, 2992
- src/main/activemq/commands/Transaction-Info.h, 2993
- src/main/activemq/commands/Wire-FormatInfo.h, 2993
- src/main/activemq/commands/XATransaction-Id.h, 2994
- src/main/activemq/core/ActiveMQAck-Handler.h, 2994
- src/main/activemq/core/ActiveMQConnection.-h, 2995
- src/main/activemq/core/ActiveMQConnection-Factory.h, 2995
- src/main/activemq/core/ActiveMQConnection-MetaData.h, 2996
- src/main/activemq/core/ActiveMQConstants.-h, 2996
- src/main/activemq/core/ActiveMQConsumer.-h, 2997
- src/main/activemq/core/ActiveMQProducer.-h, 2997
- src/main/activemq/core/ActiveMQQueue-Browser.h, 2998
- src/main/activemq/core/ActiveMQSession.-h, 2998
- src/main/activemq/core/ActiveMQSession-Executor.h, 2999
- src/main/activemq/core/ActiveMQTransaction-Context.h, 3000
- src/main/activemq/core/ActiveMQXA-Connection.h, 3000
- src/main/activemq/core/ActiveMQXA-ConnectionFactory.h, 3001
- src/main/activemq/core/ActiveMQXA-Session.h, 3001
- src/main/activemq/core/DispatchData.h, 3001
- src/main/activemq/core/Dispatcher.h, 3002
- src/main/activemq/core/FifoMessage-DispatchChannel.h, 3002
- src/main/activemq/core/MessageDispatch-Channel.h, 3003
- src/main/activemq/core/PrefetchPolicy.h, 3004
- src/main/activemq/core/RedeliveryPolicy.-h, 3005
- src/main/activemq/core/SimplePriority-MessageDispatchChannel.h, 3005
- src/main/activemq/core/Synchronization.-h, 3005
- src/main/activemq/core/policies/Default-PrefetchPolicy.h, 3003
- src/main/activemq/core/policies/Default-RedeliveryPolicy.h, 3004
- src/main/activemq/exceptions/ActiveMQ-Exception.h, 3006
- src/main/activemq/exceptions/Broker-Exception.h, 3006
- src/main/activemq/exceptions/Connection-FailedException.h, 3007
- src/main/activemq/exceptions/Exception-

- Defines.h, 3007
- src/main/activemq/io/LoggingInput-Stream.h, 3012
- src/main/activemq/io/LoggingOutput-Stream.h, 3012
- src/main/activemq/library/ActiveMQCPP.h, 3012
- src/main/activemq/state/Command-Visitor.h, 3013
- src/main/activemq/state/CommandVisitor-Adapter.h, 3013
- src/main/activemq/state/Connection-State.h, 3014
- src/main/activemq/state/ConnectionState-Tracker.h, 3015
- src/main/activemq/state/ConsumerState.h, 3015
- src/main/activemq/state/ProducerState.h, 3016
- src/main/activemq/state/SessionState.h, 3016
- src/main/activemq/state/Tracked.h, 3017
- src/main/activemq/state/Transaction-State.h, 3017
- src/main/activemq/threads/Composite-Task.h, 3018
- src/main/activemq/threads/Composite-TaskRunner.h, 3018
- src/main/activemq/threads/Dedicated-TaskRunner.h, 3019
- src/main/activemq/threads/Scheduler.h, 3019
- src/main/activemq/threads/Scheduler-TimerTask.h, 3020
- src/main/activemq/threads/Task.h, 3020
- src/main/activemq/threads/TaskRunner.h, 3020
- src/main/activemq/transport/Abstract-TransportFactory.h, 3021
- src/main/activemq/transport/Composite-Transport.h, 3021
- src/main/activemq/transport/Default-TransportListener.h, 3023
- src/main/activemq/transport/IOTransport.h, 3029
- src/main/activemq/transport/Transport.h, 3034
- src/main/activemq/transport/Transport-Factory.h, 3035
- src/main/activemq/transport/Transport-Filter.h, 3035
- src/main/activemq/transport/Transport-Listener.h, 3036
- src/main/activemq/transport/Transport-Registry.h, 3036
- src/main/activemq/transport/correlator/-FutureResponse.h, 3022
- src/main/activemq/transport/correlator/-ResponseCorrelator.h, 3022
- src/main/activemq/transport/failover/-BackupTransport.h, 3024
- src/main/activemq/transport/failover/-BackupTransportPool.h, 3024
- src/main/activemq/transport/failover/-CloseTransportsTask.h, 3025
- src/main/activemq/transport/failover/-FailoverTransport.h, 3025
- src/main/activemq/transport/failover/-FailoverTransportFactory.h, 3026
- src/main/activemq/transport/failover/-FailoverTransportListener.h, 3026
- src/main/activemq/transport/failover/URI-Pool.h, 3027
- src/main/activemq/transport/inactivity/-InactivityMonitor.h, 3027
- src/main/activemq/transport/inactivity/-ReadChecker.h, 3028
- src/main/activemq/transport/inactivity/-WriteChecker.h, 3028
- src/main/activemq/transport/logging/-LoggingTransport.h, 3029
- src/main/activemq/transport/mock/-InternalCommandListener.h, 3030
- src/main/activemq/transport/mock/Mock-Transport.h, 3030
- src/main/activemq/transport/mock/Mock-TransportFactory.h, 3031
- src/main/activemq/transport/mock/-ResponseBuilder.h, 3032
- src/main/activemq/transport/tcp/Ssl-Transport.h, 3032
- src/main/activemq/transport/tcp/Ssl-TransportFactory.h, 3033
- src/main/activemq/transport/tcp/Tcp-Transport.h, 3033
- src/main/activemq/transport/tcp/Tcp-TransportFactory.h, 3034

- src/main/activemq/util/ActiveMQProperties.- h, 3037
- src/main/activemq/util/CMSException-Support.h, 3037
- src/main/activemq/util/CompositeData.h, 3039
- src/main/activemq/util/Config.h, 3039
- src/main/activemq/util/IdGenerator.h, 3040
- src/main/activemq/util/LongSequence-Generator.h, 3041
- src/main/activemq/util/MarshallingSupport.- h, 3041
- src/main/activemq/util/MemoryUsage.h, 3041
- src/main/activemq/util/PrimitiveList.h, 3042
- src/main/activemq/util/PrimitiveMap.h, 3042
- src/main/activemq/util/PrimitiveValue-Converter.h, 3043
- src/main/activemq/util/PrimitiveValue-Node.h, 3043
- src/main/activemq/util/Service.h, 3044
- src/main/activemq/util/ServiceListener.h, 3044
- src/main/activemq/util/ServiceStopper.h, 3045
- src/main/activemq/util/ServiceSupport.h, 3045
- src/main/activemq/util/URISupport.h, 3046
- src/main/activemq/util/Usage.h, 3046
- src/main/activemq/wireformat/Marshal-Aware.h, 3046
- src/main/activemq/wireformat/Wire-Format.h, 3094
- src/main/activemq/wireformat/Wire-FormatFactory.h, 3094
- src/main/activemq/wireformat/Wire-FormatNegotiator.h, 3095
- src/main/activemq/wireformat/Wire-FormatRegistry.h, 3095
- src/main/activemq/wireformat/openwire/-OpenWireFormat.h, 3088
- src/main/activemq/wireformat/openwire/-OpenWireFormatFactory.h, 3088
- src/main/activemq/wireformat/openwire/-OpenWireFormatNegotiator.h, 3089
- src/main/activemq/wireformat/openwire/-OpenWireResponseBuilder.h, 3089
- src/main/activemq/wireformat/openwire/marshal/-BaseDataStreamMarshaller.h, 3047
- src/main/activemq/wireformat/openwire/marshal/-DataStreamMarshaller.h, 3047
- src/main/activemq/wireformat/openwire/marshal/-PrimitiveTypesMarshaller.h, 3087
- src/main/activemq/wireformat/openwire/marshal/generated/-ActiveMQBlobMessageMarshaller.- h, 3048
- src/main/activemq/wireformat/openwire/marshal/generated/-ActiveMQBytesMessage-Marshaller.h, 3049
- src/main/activemq/wireformat/openwire/marshal/generated/-ActiveMQDestinationMarshaller.- h, 3049
- src/main/activemq/wireformat/openwire/marshal/generated/-ActiveMQMapMessageMarshaller.- h, 3050
- src/main/activemq/wireformat/openwire/marshal/generated/-ActiveMQMessageMarshaller.h, 3051
- src/main/activemq/wireformat/openwire/marshal/generated/-ActiveMQObjectMessage-Marshaller.h, 3051
- src/main/activemq/wireformat/openwire/marshal/generated/-ActiveMQQueueMarshaller.h, 3052
- src/main/activemq/wireformat/openwire/marshal/generated/-ActiveMQStreamMessage-Marshaller.h, 3052
- src/main/activemq/wireformat/openwire/marshal/generated/-ActiveMQTempDestination-Marshaller.h, 3053
- src/main/activemq/wireformat/openwire/marshal/generated/-ActiveMQTempQueueMarshaller.- h, 3054
- src/main/activemq/wireformat/openwire/marshal/generated/-ActiveMQTempTopicMarshaller.- h, 3054
- src/main/activemq/wireformat/openwire/marshal/generated/-ActiveMQTextMessageMarshaller.- h, 3055
- src/main/activemq/wireformat/openwire/marshal/generated/-ActiveMQTopicMarshaller.h,

3056
src/main/activemq/wireformat/openwire/marshall/generated/-
BaseCommandMarshaller.h, 3056
src/main/activemq/wireformat/openwire/marshall/generated/-
BrokerIdMarshaller.h, 3057
src/main/activemq/wireformat/openwire/marshall/generated/-
BrokerInfoMarshaller.h, 3058
src/main/activemq/wireformat/openwire/marshall/generated/-
JournalTraceMarshaller.h, 3069
src/main/activemq/wireformat/openwire/marshall/generated/-
ConnectionControlMarshaller.h, 3058
src/main/activemq/wireformat/openwire/marshall/generated/-
JournalTransactionMarshaller.h, 3070
src/main/activemq/wireformat/openwire/marshall/generated/-
ConnectionErrorMarshaller.h, 3059
src/main/activemq/wireformat/openwire/marshall/generated/-
KeepAliveInfoMarshaller.h, 3070
src/main/activemq/wireformat/openwire/marshall/generated/-
ConnectionIdMarshaller.h, 3060
src/main/activemq/wireformat/openwire/marshall/generated/-
ConnectionInfoMarshaller.h, 3071
src/main/activemq/wireformat/openwire/marshall/generated/-
ConsumerControlMarshaller.h, 3071
src/main/activemq/wireformat/openwire/marshall/generated/-
ConsumerIdMarshaller.h, 3061
src/main/activemq/wireformat/openwire/marshall/generated/-
ConsumerInfoMarshaller.h, 3062
src/main/activemq/wireformat/openwire/marshall/generated/-
ControlCommandMarshaller.h, 3063
src/main/activemq/wireformat/openwire/marshall/generated/-
DataArrayResponseMarshaller.h, 3063
src/main/activemq/wireformat/openwire/marshall/generated/-
DataResponseMarshaller.h, 3064
src/main/activemq/wireformat/openwire/marshall/generated/-
DestinationInfoMarshaller.h, 3065
src/main/activemq/wireformat/openwire/marshall/generated/-
DiscoveryEventMarshaller.h, 3065
src/main/activemq/wireformat/openwire/marshall/generated/-
ExceptionResponseMarshaller.h, 3066
src/main/activemq/wireformat/openwire/marshall/generated/-
FlushCommandMarshaller.h, 3066
src/main/activemq/wireformat/openwire/marshall/generated/-
IntegerResponseMarshaller.h, 3067
src/main/activemq/wireformat/openwire/marshall/generated/-
JournalQueueAckMarshaller.h, 3068
src/main/activemq/wireformat/openwire/marshall/generated/-
JournalTopicAckMarshaller.h, 3068
src/main/activemq/wireformat/openwire/marshall/generated/-
JournalTraceMarshaller.h, 3069
src/main/activemq/wireformat/openwire/marshall/generated/-
JournalTransactionMarshaller.h, 3070
src/main/activemq/wireformat/openwire/marshall/generated/-
KeepAliveInfoMarshaller.h, 3070
src/main/activemq/wireformat/openwire/marshall/generated/-
PartialCommandMarshaller.h, 3071
src/main/activemq/wireformat/openwire/marshall/generated/-
TransactionIdMarshaller.h, 3071
src/main/activemq/wireformat/openwire/marshall/generated/-
MessageFactory.h, 3072
src/main/activemq/wireformat/openwire/marshall/generated/-
MessageAckMarshaller.h, 3073
src/main/activemq/wireformat/openwire/marshall/generated/-
MessageDispatchMarshaller.h, 3073
src/main/activemq/wireformat/openwire/marshall/generated/-
MessageDispatchNotificationMarshaller.h, 3074
src/main/activemq/wireformat/openwire/marshall/generated/-
MessageIdMarshaller.h, 3074
src/main/activemq/wireformat/openwire/marshall/generated/-
MessageMarshaller.h, 3075
src/main/activemq/wireformat/openwire/marshall/generated/-
MessagePullMarshaller.h, 3076
src/main/activemq/wireformat/openwire/marshall/generated/-
NetworkBridgeFilterMarshaller.h, 3076
src/main/activemq/wireformat/openwire/marshall/generated/-
PartialCommandMarshaller.h, 3077
src/main/activemq/wireformat/openwire/marshall/generated/-
ProducerAckMarshaller.h, 3078
src/main/activemq/wireformat/openwire/marshall/generated/-
ProducerIdMarshaller.h, 3078
src/main/activemq/wireformat/openwire/marshall/generated/-
ProducerInfoMarshaller.h, 3079
src/main/activemq/wireformat/openwire/marshall/generated/-

RemoveInfoMarshaller.h, 3079
 src/main/activemq/wireformat/openwire/marshaller/generated/-
 RemoveSubscriptionInfo-
 Marshaller.h, 3080
 src/main/activemq/wireformat/openwire/marshaller/generated/-
 ReplayCommandMarshaller.h, 3081
 src/main/activemq/wireformat/openwire/marshaller/generated/-
 ResponseMarshaller.h, 3081
 src/main/activemq/wireformat/openwire/marshaller/generated/-
 SessionIdMarshaller.h, 3082
 src/main/activemq/wireformat/openwire/marshaller/generated/-
 SessionInfoMarshaller.h, 3083
 src/main/activemq/wireformat/openwire/marshaller/generated/-
 ShutdownInfoMarshaller.h, 3083
 src/main/activemq/wireformat/openwire/marshaller/generated/-
 SubscriptionInfoMarshaller.h, 3084
 src/main/activemq/wireformat/openwire/marshaller/generated/-
 TransactionIdMarshaller.h, 3084
 src/main/activemq/wireformat/openwire/marshaller/generated/-
 TransactionInfoMarshaller.h, 3085
 src/main/activemq/wireformat/openwire/marshaller/generated/-
 WireFormatInfoMarshaller.h, 3086
 src/main/activemq/wireformat/openwire/marshaller/generated/-
 XATransactionIdMarshaller.h, 3086
 src/main/activemq/wireformat/openwire/utills/-
 BooleanStream.h, 3090
 src/main/activemq/wireformat/openwire/utills/-
 HexTable.h, 3090
 src/main/activemq/wireformat/openwire/utills/-
 MessagePropertyInterceptor.h, 3091
 src/main/activemq/wireformat/stomp/-
 StompCommandConstants.h, 3091
 src/main/activemq/wireformat/stomp/-
 StompFrame.h, 3092
 src/main/activemq/wireformat/stomp/-
 StompHelper.h, 3092
 src/main/activemq/wireformat/stomp/-
 StompWireFormat.h, 3093
 src/main/activemq/wireformat/stomp/-
 StompWireFormatFactory.h, 3093
 src/main/cms/BytesMessage.h, 3096
 src/main/cms/CMSException.h, 3097
 src/main/cms/CMSProperties.h, 3098
 src/main/cms/CMSSecurityException.h, 3098
 src/main/cms/Closeable.h, 3096
 src/main/cms/Config.h, 3039
 src/main/cms/Connection.h, 3098
 src/main/cms/ConnectionFactory.h, 3099
 src/main/cms/ConnectionMetaData.h, 3099
 src/main/cms/DeliveryMode.h, 3100
 src/main/cms/Destination.h, 3100
 src/main/cms/ExceptionListener.h, 3100
 src/main/cms/IllegalStateException.h, 3101
 src/main/cms/InvalidClientIdException.h, 3102
 src/main/cms/InvalidDestinationException.h, 3102
 src/main/cms/InvalidSelectorException.h, 3103
 src/main/cms/MapMessage.h, 3103
 src/main/cms/Message.h, 2984
 src/main/cms/MessageConsumer.h, 3103
 src/main/cms/MessageEOFException.h, 3104
 src/main/cms/MessageEnumeration.h, 3104
 src/main/cms/MessageFormatException.h, 3105
 src/main/cms/MessageListener.h, 3105
 src/main/cms/MessageNotReadableException.h, 3105
 src/main/cms/MessageNotWritableException.h, 3106
 src/main/cms/MessageProducer.h, 3106
 src/main/cms/ObjectMessage.h, 3107
 src/main/cms/Queue.h, 3107
 src/main/cms/QueueBrowser.h, 3108
 src/main/cms/Session.h, 3109
 src/main/cms/Startable.h, 3109
 src/main/cms/Stoppable.h, 3109
 src/main/cms/StreamMessage.h, 3110
 src/main/cms/TemporaryQueue.h, 3110
 src/main/cms/TemporaryTopic.h, 3111
 src/main/cms/TextMessage.h, 3111
 src/main/cms/Topic.h, 3112
 src/main/cms/TransactionInProgressException.h, 3112

- src/main/cms/TransactionRolledBackException.h, 3112
- src/main/cms/UnsupportedOperationException.h, 3113
- src/main/cms/XAConnection.h, 3114
- src/main/cms/XAConnectionFactory.h, 3114
- src/main/cms/XAException.h, 3115
- src/main/cms/XAResource.h, 3115
- src/main/cms/XASession.h, 3115
- src/main/cms/Xid.h, 3116
- src/main/decaf/internal/AprPool.h, 3116
- src/main/decaf/internal/DecafRuntime.h, 3117
- src/main/decaf/internal/io/StandardErrorOutputStream.h, 3117
- src/main/decaf/internal/io/StandardInputStream.h, 3118
- src/main/decaf/internal/io/StandardOutputStream.h, 3118
- src/main/decaf/internal/net/DefaultServerSocketFactory.h, 3119
- src/main/decaf/internal/net/DefaultSocketFactory.h, 3119
- src/main/decaf/internal/net/Network.h, 3120
- src/main/decaf/internal/net/SocketFileDescriptor.h, 3120
- src/main/decaf/internal/net/URIEncoderDecoder.h, 3128
- src/main/decaf/internal/net/URIHelper.h, 3129
- src/main/decaf/internal/net/URIType.h, 3129
- src/main/decaf/internal/net/ssl/DefaultSSLContext.h, 3121
- src/main/decaf/internal/net/ssl/DefaultSSLServerSocketFactory.h, 3121
- src/main/decaf/internal/net/ssl/DefaultSSLSocketFactory.h, 3122
- src/main/decaf/internal/net/ssl/openssl/OpenSSLContextSpi.h, 3122
- src/main/decaf/internal/net/ssl/openssl/OpenSSLParameters.h, 3123
- src/main/decaf/internal/net/ssl/openssl/OpenSSLServerSocket.h, 3123
- src/main/decaf/internal/net/ssl/openssl/OpenSSLServerSocketFactory.h, 3124
- src/main/decaf/internal/net/ssl/openssl/OpenSSLSocket.h, 3124
- src/main/decaf/internal/net/ssl/openssl/OpenSSLSocketException.h, 3125
- src/main/decaf/internal/net/ssl/openssl/OpenSSLSocketFactory.h, 3125
- src/main/decaf/internal/net/ssl/openssl/OpenSSLSocketInputStream.h, 3126
- src/main/decaf/internal/net/ssl/openssl/OpenSSLSocketOutputStream.h, 3126
- src/main/decaf/internal/net/tcp/TcpSocket.h, 3127
- src/main/decaf/internal/net/tcp/TcpSocketInputStream.h, 3127
- src/main/decaf/internal/net/tcp/TcpSocketOutputStream.h, 3128
- src/main/decaf/internal/nio/BufferFactory.h, 3129
- src/main/decaf/internal/nio/ByteBuffer.h, 3130
- src/main/decaf/internal/nio/CharArrayBuffer.h, 3131
- src/main/decaf/internal/nio/DoubleArrayBuffer.h, 3131
- src/main/decaf/internal/nio/FloatArrayBuffer.h, 3132
- src/main/decaf/internal/nio/IntArrayBuffer.h, 3132
- src/main/decaf/internal/nio/LongArrayBuffer.h, 3133
- src/main/decaf/internal/nio/ShortArrayBuffer.h, 3133
- src/main/decaf/internal/security/unix/SecureRandomImpl.h, 3134
- src/main/decaf/internal/security/windows/SecureRandomImpl.h, 3134
- src/main/decaf/internal/util/ByteArrayAdapter.h, 3135
- src/main/decaf/internal/util/GenericResource.h, 3140
- src/main/decaf/internal/util/HexStringParser.h, 3141
- src/main/decaf/internal/util/Resource.h, 3141
- src/main/decaf/internal/util/ResourceLifecycleManager.h, 2962
- src/main/decaf/internal/util/TimerTask-

- Heap.h, 3141
- src/main/decaf/internal/util/concurrent/-
ConditionImpl.h, 3135
- src/main/decaf/internal/util/concurrent/-
MutexImpl.h, 3136
- src/main/decaf/internal/util/concurrent/-
SynchronizableImpl.h, 3136
- src/main/decaf/internal/util/concurrent/-
TransferQueue.h, 3137
- src/main/decaf/internal/util/concurrent/-
TransferStack.h, 3138
- src/main/decaf/internal/util/concurrent/-
Transferer.h, 3137
- src/main/decaf/internal/util/concurrent/unix/-
ConditionHandle.h, 3138
- src/main/decaf/internal/util/concurrent/unix/-
MutexHandle.h, 3139
- src/main/decaf/internal/util/concurrent/windows/-
ConditionHandle.h, 3139
- src/main/decaf/internal/util/concurrent/windows/-
MutexHandle.h, 3140
- src/main/decaf/internal/util/zip/crc32.h,
3142
- src/main/decaf/internal/util/zip/deflate.h,
3142
- src/main/decaf/internal/util/zip/gzguts.h,
3146
- src/main/decaf/internal/util/zip/inffast.h,
3148
- src/main/decaf/internal/util/zip/inffixed.h,
3148
- src/main/decaf/internal/util/zip/inflate.h,
3148
- src/main/decaf/internal/util/zip/inftrees.h,
3149
- src/main/decaf/internal/util/zip/trees.h,
3150
- src/main/decaf/internal/util/zip/zconf.h,
3152
- src/main/decaf/internal/util/zip/zlib.h, 3153
- src/main/decaf/internal/util/zip/zutil.h,
3161
- src/main/decaf/io/BlockingByteArrayInput-
Stream.h, 3164
- src/main/decaf/io/BufferedInputStream.h,
3164
- src/main/decaf/io/BufferedOutputStream.-
h, 3164
- src/main/decaf/io/ByteArrayInputStream.-
h, 3165
- src/main/decaf/io/ByteArrayOutput-
Stream.h, 3165
- src/main/decaf/io/Closeable.h, 3097
- src/main/decaf/io/DataInput.h, 3166
- src/main/decaf/io/DataInputStream.h,
3166
- src/main/decaf/io/DataOutput.h, 3167
- src/main/decaf/io/DataOutputStream.h,
3167
- src/main/decaf/io/EOFException.h, 3168
- src/main/decaf/io/FileDescriptor.h, 3168
- src/main/decaf/io/FilterInputStream.h,
3169
- src/main/decaf/io/FilterOutputStream.h,
3169
- src/main/decaf/io/Flushable.h, 3170
- src/main/decaf/io/IOException.h, 3171
- src/main/decaf/io/InputStream.h, 3170
- src/main/decaf/io/InputStreamReader.h,
3171
- src/main/decaf/io/InterruptedIOException.-
h, 3171
- src/main/decaf/io/OutputStream.h, 3172
- src/main/decaf/io/OutputStreamWriter.h,
3172
- src/main/decaf/io/PushbackInputStream.-
h, 3173
- src/main/decaf/io/Reader.h, 3173
- src/main/decaf/io/UTFDataFormat-
Exception.h, 3174
- src/main/decaf/io/UnsupportedEncoding-
Exception.h, 3174
- src/main/decaf/io/Writer.h, 3174
- src/main/decaf/lang/Appendable.h, 3175
- src/main/decaf/lang/ArrayPointer.h, 3175
- src/main/decaf/lang/Boolean.h, 3176
- src/main/decaf/lang/Byte.h, 3177
- src/main/decaf/lang/CharSequence.h,
3177
- src/main/decaf/lang/Character.h, 3177
- src/main/decaf/lang/Comparable.h, 3178
- src/main/decaf/lang/Double.h, 3178
- src/main/decaf/lang/Exception.h, 3179
- src/main/decaf/lang/Float.h, 3183
- src/main/decaf/lang/Integer.h, 3184
- src/main/decaf/lang/Iterable.h, 3184
- src/main/decaf/lang/Long.h, 3185
- src/main/decaf/lang/Math.h, 3185
- src/main/decaf/lang/Number.h, 3186
- src/main/decaf/lang/Pointer.h, 3186

- src/main/decaf/lang/Readable.h, 3187
- src/main/decaf/lang/Runnable.h, 3187
- src/main/decaf/lang/Runtime.h, 3188
- src/main/decaf/lang/Short.h, 3188
- src/main/decaf/lang/String.h, 3189
- src/main/decaf/lang/System.h, 3189
- src/main/decaf/lang/Thread.h, 3190
- src/main/decaf/lang/ThreadGroup.h, 3190
- src/main/decaf/lang/Throwable.h, 3191
- src/main/decaf/lang/exceptions/Class-CastException.h, 3179
- src/main/decaf/lang/exceptions/Exception-Defines.h, 3009
- src/main/decaf/lang/exceptions/Illegal-ArgumentException.h, 3180
- src/main/decaf/lang/exceptions/Illegal-MonitorStateException.h, 3180
- src/main/decaf/lang/exceptions/Illegal-StateException.h, 3101
- src/main/decaf/lang/exceptions/Illegal-ThreadStateException.h, 3180
- src/main/decaf/lang/exceptions/IndexOut-OfBoundsException.h, 3181
- src/main/decaf/lang/exceptions/Interrupted-Exception.h, 3181
- src/main/decaf/lang/exceptions/Invalid-StateException.h, 3182
- src/main/decaf/lang/exceptions/Null-PointerException.h, 3182
- src/main/decaf/lang/exceptions/Number-FormatException.h, 3183
- src/main/decaf/lang/exceptions/Runtime-Exception.h, 3183
- src/main/decaf/lang/exceptions/Unsupported-OperationException.h, 3113
- src/main/decaf/net/BindException.h, 3191
- src/main/decaf/net/ConnectException.h, 3192
- src/main/decaf/net/DatagramPacket.h, 3192
- src/main/decaf/net/HttpRetryException.h, 3192
- src/main/decaf/net/Inet4Address.h, 3193
- src/main/decaf/net/Inet6Address.h, 3193
- src/main/decaf/net/InetAddress.h, 3194
- src/main/decaf/net/InetSocketAddress.h, 3194
- src/main/decaf/net/MalformedURLException-Exception.h, 3194
- src/main/decaf/net/NoRouteToHost-Exception.h, 3195
- src/main/decaf/net/PortUnreachable-Exception.h, 3195
- src/main/decaf/net/ProtocolException.h, 3196
- src/main/decaf/net/ServerSocket.h, 3196
- src/main/decaf/net/ServerSocketFactory.-h, 3196
- src/main/decaf/net/Socket.h, 3197
- src/main/decaf/net/SocketAddress.h, 3197
- src/main/decaf/net/SocketError.h, 3198
- src/main/decaf/net/SocketException.h, 3198
- src/main/decaf/net/SocketFactory.h, 3199
- src/main/decaf/net/SocketImpl.h, 3199
- src/main/decaf/net/SocketImplFactory.h, 3200
- src/main/decaf/net/SocketOptions.h, 3200
- src/main/decaf/net/SocketTimeout-Exception.h, 3200
- src/main/decaf/net/URI.h, 3205
- src/main/decaf/net/URISyntaxException.-h, 3205
- src/main/decaf/net/URL.h, 3206
- src/main/decaf/net/URLDecoder.h, 3206
- src/main/decaf/net/URLEncoder.h, 3206
- src/main/decaf/net/UnknownHostException.-h, 3204
- src/main/decaf/net/UnknownService-Exception.h, 3204
- src/main/decaf/net/ssl/SSLContext.h, 3201
- src/main/decaf/net/ssl/SSLContextSpi.h, 3201
- src/main/decaf/net/ssl/SSLParameters.h, 3202
- src/main/decaf/net/ssl/SSLServerSocket.-h, 3202
- src/main/decaf/net/ssl/SSLServerSocket-Factory.h, 3203
- src/main/decaf/net/ssl/SSLSocket.h, 3203
- src/main/decaf/net/ssl/SSLSocketFactory.-h, 3203
- src/main/decaf/nio/Buffer.h, 3207
- src/main/decaf/nio/BufferOverflowException.-h, 3207
- src/main/decaf/nio/BufferUnderflow-Exception.h, 3208

- src/main/decaf/nio/ByteBuffer.h, 3208
src/main/decaf/nio/CharBuffer.h, 3208
src/main/decaf/nio/DoubleBuffer.h, 3209
src/main/decaf/nio/FloatBuffer.h, 3210
src/main/decaf/nio/IntBuffer.h, 3210
src/main/decaf/nio/InvalidMarkException.h, 3211
src/main/decaf/nio/LongBuffer.h, 3211
src/main/decaf/nio/ReadOnlyBufferException.h, 3211
src/main/decaf/nio/ShortBuffer.h, 3212
src/main/decaf/security/GeneralSecurityException.h, 3216
src/main/decaf/security/InvalidKeyException.h, 3216
src/main/decaf/security/Key.h, 3217
src/main/decaf/security/KeyException.h, 3217
src/main/decaf/security/KeyManagementException.h, 3218
src/main/decaf/security/NoSuchAlgorithmExceptionException.h, 3218
src/main/decaf/security/NoSuchProviderException.h, 3218
src/main/decaf/security/Principal.h, 3219
src/main/decaf/security/PublicKey.h, 3219
src/main/decaf/security/SecureRandom.h, 3220
src/main/decaf/security/SecureRandomSpi.h, 3220
src/main/decaf/security/SignatureException.h, 3221
src/main/decaf/security/auth/x500/X500Principal.h, 3212
src/main/decaf/security/cert/Certificate.h, 3213
src/main/decaf/security/cert/CertificateEncodingException.h, 3213
src/main/decaf/security/cert/CertificateException.h, 3214
src/main/decaf/security/cert/CertificateExpiredException.h, 3214
src/main/decaf/security/cert/CertificateNotYetValidException.h, 3215
src/main/decaf/security/cert/CertificateParsingException.h, 3215
src/main/decaf/security/cert/X509Certificate.h, 3216
src/main/decaf/util/AbstractCollection.h, 3221
src/main/decaf/util/AbstractList.h, 3222
src/main/decaf/util/AbstractMap.h, 3222
src/main/decaf/util/AbstractQueue.h, 3223
src/main/decaf/util/AbstractSequentialList.h, 3223
src/main/decaf/util/AbstractSet.h, 3224
src/main/decaf/util/ArrayList.h, 3224
src/main/decaf/util/Arrays.h, 3225
src/main/decaf/util/Collection.h, 3225
src/main/decaf/util/Comparator.h, 3226
src/main/decaf/util/ConcurrentModificationException.h, 3247
src/main/decaf/util/Config.h, 3040
src/main/decaf/util/Date.h, 3248
src/main/decaf/util/Deque.h, 3248
src/main/decaf/util/Iterator.h, 3248
src/main/decaf/util/LinkedList.h, 3249
src/main/decaf/util/List.h, 3250
src/main/decaf/util/ListIterator.h, 3250
src/main/decaf/util/Map.h, 3260
src/main/decaf/util/NoSuchElementException.h, 3260
src/main/decaf/util/PriorityQueue.h, 3261
src/main/decaf/util/Properties.h, 3261
src/main/decaf/util/Queue.h, 3108
src/main/decaf/util/Random.h, 3262
src/main/decaf/util/Set.h, 3262
src/main/decaf/util/StlList.h, 3263
src/main/decaf/util/StlMap.h, 3263
src/main/decaf/util/StlQueue.h, 3264
src/main/decaf/util/StlSet.h, 3264
src/main/decaf/util/StringTokenizer.h, 3265
src/main/decaf/util/Timer.h, 3265
src/main/decaf/util/TimerTask.h, 3266
src/main/decaf/util/UUID.h, 3266
src/main/decaf/util/comparators/Less.h, 3226
src/main/decaf/util/concurrent/AbstractExecutorService.h, 3227
src/main/decaf/util/concurrent/BlockingQueue.h, 3229
src/main/decaf/util/concurrent/BrokenBarrierException.h, 3230
src/main/decaf/util/concurrent/Callable.h, 3230
src/main/decaf/util/concurrent/CancellationException.h, 3230
src/main/decaf/util/concurrent/Concurrent.h, 3231

- src/main/decaf/util/concurrent/Concurrent-Map.h, 3232
- src/main/decaf/util/concurrent/Concurrent-StlMap.h, 3232
- src/main/decaf/util/concurrent/CopyOn-WriteArrayList.h, 3233
- src/main/decaf/util/concurrent/CopyOn-WriteArraySet.h, 3234
- src/main/decaf/util/concurrent/Count-DownLatch.h, 3234
- src/main/decaf/util/concurrent/Delayed.h, 3235
- src/main/decaf/util/concurrent/Execution-Exception.h, 3235
- src/main/decaf/util/concurrent/Executor.h, 3236
- src/main/decaf/util/concurrent/Executor-Service.h, 3237
- src/main/decaf/util/concurrent/Executors.-h, 3236
- src/main/decaf/util/concurrent/Future.h, 3237
- src/main/decaf/util/concurrent/Linked-BlockingQueue.h, 3238
- src/main/decaf/util/concurrent/Lock.h, 3238
- src/main/decaf/util/concurrent/Mutex.h, 3242
- src/main/decaf/util/concurrent/Rejected-ExecutionException.h, 3242
- src/main/decaf/util/concurrent/Rejected-ExecutionHandler.h, 3243
- src/main/decaf/util/concurrent/Semaphore.-h, 3243
- src/main/decaf/util/concurrent/Synchronizable.h, 3244
- src/main/decaf/util/concurrent/Synchronous-Queue.h, 3244
- src/main/decaf/util/concurrent/Thread-Factory.h, 3245
- src/main/decaf/util/concurrent/Thread-PoolExecutor.h, 3245
- src/main/decaf/util/concurrent/TimeUnit.h, 3247
- src/main/decaf/util/concurrent/Timeout-Exception.h, 3246
- src/main/decaf/util/concurrent/atomic/-AtomicBoolean.h, 3227
- src/main/decaf/util/concurrent/atomic/-AtomicInteger.h, 3228
- src/main/decaf/util/concurrent/atomic/-AtomicRefCounter.h, 3228
- src/main/decaf/util/concurrent/atomic/-AtomicReference.h, 3229
- src/main/decaf/util/concurrent/locks/-AbstractOwnableSynchronizer.-h, 3239
- src/main/decaf/util/concurrent/locks/-Condition.h, 3240
- src/main/decaf/util/concurrent/locks/Lock.-h, 3239
- src/main/decaf/util/concurrent/locks/Lock-Support.h, 3240
- src/main/decaf/util/concurrent/locks/Read-WriteLock.h, 3241
- src/main/decaf/util/concurrent/locks/-ReentrantLock.h, 3241
- src/main/decaf/util/logging/Console-Handler.h, 3251
- src/main/decaf/util/logging/EventManager.-h, 3251
- src/main/decaf/util/logging/Filter.h, 3251
- src/main/decaf/util/logging/Formatter.h, 3252
- src/main/decaf/util/logging/Handler.h, 3252
- src/main/decaf/util/logging/Level.h, 3253
- src/main/decaf/util/logging/LogManager.h, 3256
- src/main/decaf/util/logging/LogRecord.h, 3256
- src/main/decaf/util/logging/LogWriter.h, 3257
- src/main/decaf/util/logging/Logger.h, 3253
- src/main/decaf/util/logging/LoggerCommon.-h, 3254
- src/main/decaf/util/logging/LoggerDefines.-h, 3254
- src/main/decaf/util/logging/LoggerHierarchy.-h, 3255
- src/main/decaf/util/logging/MarkBlock-Logger.h, 3257
- src/main/decaf/util/logging/Properties-ChangeListener.h, 3258
- src/main/decaf/util/logging/SimpleFormatter.-h, 3258
- src/main/decaf/util/logging/SimpleLogger.-h, 3259
- src/main/decaf/util/logging/Stream-Handler.h, 3259

- src/main/decaf/util/logging/XMLFormatter.h, 3260
- src/main/decaf/util/zip/Adler32.h, 3267
- src/main/decaf/util/zip/CRC32.h, 3269
- src/main/decaf/util/zip/CheckedInputStream.h, 3267
- src/main/decaf/util/zip/CheckedOutputStream.h, 3268
- src/main/decaf/util/zip/Checksum.h, 3268
- src/main/decaf/util/zip/DataFormatException.h, 3269
- src/main/decaf/util/zip/Deflater.h, 3270
- src/main/decaf/util/zip/DeflaterOutputStream.h, 3270
- src/main/decaf/util/zip/Inflater.h, 3271
- src/main/decaf/util/zip/InflaterInputStream.h, 3271
- src/main/decaf/util/zip/ZipException.h, 3272
- stackTrace
 - decaf::lang::Exception, 1286
- start
 - activemq::cmsutil::CachedConsumer, 738
 - activemq::cmsutil::PooledSession, 2106
 - activemq::commands::ConsumerControl, 996
 - activemq::core::ActiveMQConnection, 211
 - activemq::core::ActiveMQConsumer, 244
 - activemq::core::ActiveMQSession, 352
 - activemq::core::ActiveMQSessionExecutor, 358
 - activemq::core::ActiveMQTransactionContext, 431
 - activemq::core::FifoMessageDispatchChannel, 1328
 - activemq::core::MessageDispatchChannel, 1890
 - activemq::core::SimplePriorityMessageDispatchChannel, 2449
 - activemq::transport::correlator::ResponseCorrelator, 2307
 - activemq::transport::failover::FailoverTransport, 1317
 - activemq::transport::IOTransport, 1555
- activemq::transport::mock::MockTransport, 1957
- activemq::transport::Transport, 2797
- activemq::transport::TransportFilter, 2808
- activemq::util::Service, 2356
- activemq::util::ServiceSupport, 2361
- activemq::wireformat::openwire::OpenWireFormatNegotiator, 2064
- cms::Startable, 2534
- cms::XAResource, 2933
- decaf::lang::Thread, 2712
- gz_state, 1401
- startHandshake
 - decaf::internal::net::ssl::openssl::OpenSSLSocket, 2028
 - decaf::net::ssl::SSLSocket, 2521
- started
 - activemq::util::ServiceListener, 2357
- stat_desc
 - tree_desc_s, 2815
- state
 - z_stream_s, 2952
- static_dtree
 - trees.h, 3152
- static_len
 - internal_state, 1526
- static_ltree
 - trees.h, 3152
- static_tree_desc
 - deflate.h, 3145
- staticCast
 - decaf::lang::Pointer, 2090
- status
 - internal_state, 1526
- std, 103
- std::less< decaf::lang::ArrayPointer< T > >, 1614
 - operator(), 1615
- std::less< decaf::lang::Pointer< T > >, 1615
 - operator(), 1615
- stop
 - activemq::cmsutil::CachedConsumer, 738
 - activemq::cmsutil::PooledSession, 2106

- activemq::commands::Consumer-
Control, 996
- activemq::core::ActiveMQConnection,
212
- activemq::core::ActiveMQConsumer,
244
- activemq::core::ActiveMQSession,
352
- activemq::core::ActiveMQSession-
Executor, 358
- activemq::core::FifoMessageDispatch-
Channel, 1328
- activemq::core::MessageDispatch-
Channel, 1890
- activemq::core::SimplePriority-
MessageDispatchChannel,
2449
- activemq::transport::failover::Failover-
Transport, 1317
- activemq::transport::IOTransport,
1556
- activemq::transport::mock::Mock-
Transport, 1958
- activemq::transport::Transport, 2797
- activemq::transport::TransportFilter,
2809
- activemq::util::Service, 2356
- activemq::util::ServiceStopper, 2358
- activemq::util::ServiceSupport, 2361
- cms::Stoppable, 2602
- stopped
 - activemq::util::ServiceListener, 2357
- store
 - decaf::util::Properties, 2207, 2208
- strategy
 - gz_state, 1401
 - internal_state, 1526
- stringValue
 - activemq::util::PrimitiveValueNode::-
PrimitiveValue, 2143
- strm
 - gz_state, 1401
 - internal_state, 1526
- strstart
 - internal_state, 1526
- subSequence
 - decaf::internal::nio::CharArrayBuffer,
784
 - decaf::lang::CharSequence, 804
 - decaf::lang::String, 2624
- decaf::nio::CharBuffer, 801
- subscriptionName
 - activemq::commands::Remove-
SubscriptionInfo, 2279
 - activemq::commands::Subscription-
Info, 2635
- subscribedDestination
 - activemq::commands::Subscription-
Info, 2635
- subscriptionName
 - activemq::commands::Consumer-
Info, 1017
- subscriptionName
 - activemq::commands::JournalTopic-
Ack, 1572
- supportsUrgentData
 - decaf::net::SocketImpl, 2487
- suspend
 - activemq::commands::Connection-
Control, 943
- swap
 - decaf::lang::ArrayPointer, 469
 - decaf::lang::Pointer, 2090
 - decaf::util::concurrent::atomic::-
AtomicRefCounter, 483
- sync
 - decaf::io::FileDescriptor, 1332
- syncRequest
 - activemq::core::ActiveMQConnection,
212
 - activemq::core::ActiveMQSession,
353
- synchronized
 - Concurrent.h, 3231
- take
 - decaf::util::concurrent::Blocking-
Queue, 544
 - decaf::util::concurrent::Linked-
BlockingQueue, 1632
 - decaf::util::concurrent::Synchronous-
Queue, 2664
- takeSession
 - activemq::cmsutil::SessionPool,
2395
- targetConsumerId
 - activemq::commands::Message,
1838
- terminated

- decaf::util::concurrent::ThreadPool-
Executor, 2731
- text
 - activemq::commands::ActiveMQ-
TextMessage, 410
 - gz_header_s, 1399
- throwFirstException
 - activemq::util::ServiceStopper, 2358
- throwing
 - decaf::util::logging::Logger, 1705
- tightMarshal1
 - activemq::wireformat::openwire-
::marshal::BaseDataStream-
Marshaller, 517
 - activemq::wireformat::openwire-
::marshal::DataStreamMarshaller,
1127
 - activemq::wireformat::openwire-
::marshal::generated::Active-
MQBlobMessageMarshaller,
164
 - activemq::wireformat::openwire-
::marshal::generated::Active-
MQBytesMessageMarshaller,
186
 - activemq::wireformat::openwire-
::marshal::generated::Active-
MQDestinationMarshaller, 260
 - activemq::wireformat::openwire-
::marshal::generated::Active-
MQMapMessageMarshaller,
286
 - activemq::wireformat::openwire-
::marshal::generated::Active-
MQMessageMarshaller, 293
 - activemq::wireformat::openwire-
::marshal::generated::Active-
MQObjectMessageMarshaller,
306
 - activemq::wireformat::openwire-
::marshal::generated::Active-
MQQueueMarshaller, 331
 - activemq::wireformat::openwire-
::marshal::generated::Active-
MQStreamMessageMarshaller,
377
 - activemq::wireformat::openwire-
::marshal::generated::Active-
MQTempDestinationMarshaller,
384
 - activemq::wireformat::openwire-
::marshal::generated::Active-
MQTempQueueMarshaller, 394
 - activemq::wireformat::openwire-
::marshal::generated::Active-
MQTempTopicMarshaller, 403
 - activemq::wireformat::openwire-
::marshal::generated::ActiveM-
QTextMessageMarshaller, 413
 - activemq::wireformat::openwire-
::marshal::generated::Active-
MQTopicMarshaller, 421
 - activemq::wireformat::openwire-
::marshal::generated::Base-
CommandMarshaller, 502
 - activemq::wireformat::openwire-
::marshal::generated::Broker-
IdMarshaller, 570
 - activemq::wireformat::openwire-
::marshal::generated::Broker-
InfoMarshaller, 581
 - activemq::wireformat::openwire-
::marshal::generated::Connection-
ControlMarshaller, 946
 - activemq::wireformat::openwire-
::marshal::generated::Connection-
ErrorMarshaller, 954
 - activemq::wireformat::openwire-
::marshal::generated::Connection-
IdMarshaller, 966
 - activemq::wireformat::openwire-
::marshal::generated::Connection-
InfoMarshaller, 976
 - activemq::wireformat::openwire-
::marshal::generated::Consumer-
ControlMarshaller, 999
 - activemq::wireformat::openwire-
::marshal::generated::Consumer-
IdMarshaller, 1007
 - activemq::wireformat::openwire-
::marshal::generated::Consumer-
InfoMarshaller, 1020
 - activemq::wireformat::openwire-
::marshal::generated::Control-
CommandMarshaller, 1028
 - activemq::wireformat::openwire-
::marshal::generated::Data-
ArrayResponseMarshaller,
1074
 - activemq::wireformat::openwire-

- `::marshal::generated::Data-ResponseMarshaller`, 1118
- `activemq::wireformat::openwire-
::marshal::generated::Destination-InfoMarshaller`, 1220
- `activemq::wireformat::openwire-
::marshal::generated::Discovery-EventMarshaller`, 1232
- `activemq::wireformat::openwire-
::marshal::generated::Exception-ResponseMarshaller`, 1293
- `activemq::wireformat::openwire-
::marshal::generated::Flush-CommandMarshaller`, 1386
- `activemq::wireformat::openwire-
::marshal::generated::Integer-ResponseMarshaller`, 1521
- `activemq::wireformat::openwire-
::marshal::generated::Journal-QueueAckMarshaller`, 1566
- `activemq::wireformat::openwire-
::marshal::generated::Journal-TopicAckMarshaller`, 1575
- `activemq::wireformat::openwire-
::marshal::generated::Journal-TraceMarshaller`, 1582
- `activemq::wireformat::openwire-
::marshal::generated::Journal-TransactionMarshaller`, 1589
- `activemq::wireformat::openwire-
::marshal::generated::Keep-AliveInfoMarshaller`, 1597
- `activemq::wireformat::openwire-
::marshal::generated::Last-PartialCommandMarshaller`, 1611
- `activemq::wireformat::openwire-
::marshal::generated::Local-TransactionIdMarshaller`, 1681
- `activemq::wireformat::openwire-
::marshal::generated::Message-AckMarshaller`, 1875
- `activemq::wireformat::openwire-
::marshal::generated::Message-DispatchMarshaller`, 1893
- `activemq::wireformat::openwire-
::marshal::generated::Message-DispatchNotificationMarshaller`, 1902
- `activemq::wireformat::openwire-
::marshal::generated::Message-IdMarshaller`, 1915
- `activemq::wireformat::openwire-
::marshal::generated::Message-Marshaller`, 1920
- `activemq::wireformat::openwire-
::marshal::generated::Message-PullMarshaller`, 1946
- `activemq::wireformat::openwire-
::marshal::generated::Network-BridgeFilterMarshaller`, 1977
- `activemq::wireformat::openwire-
::marshal::generated::Partial-CommandMarshaller`, 2082
- `activemq::wireformat::openwire-
::marshal::generated::Producer-AckMarshaller`, 2177
- `activemq::wireformat::openwire-
::marshal::generated::Producer-IdMarshaller`, 2188
- `activemq::wireformat::openwire-
::marshal::generated::Producer-InfoMarshaller`, 2197
- `activemq::wireformat::openwire-
::marshal::generated::Remove-InfoMarshaller`, 2274
- `activemq::wireformat::openwire-
::marshal::generated::Remove-SubscriptionInfoMarshaller`, 2282
- `activemq::wireformat::openwire-
::marshal::generated::Replay-CommandMarshaller`, 2289
- `activemq::wireformat::openwire-
::marshal::generated::Response-Marshaller`, 2310
- `activemq::wireformat::openwire-
::marshal::generated::Session-IdMarshaller`, 2385
- `activemq::wireformat::openwire-
::marshal::generated::Session-InfoMarshaller`, 2392
- `activemq::wireformat::openwire-
::marshal::generated::Shutdown-InfoMarshaller`, 2437
- `activemq::wireformat::openwire-
::marshal::generated::Subscription-InfoMarshaller`, 2637
- `activemq::wireformat::openwire-
::marshal::generated::Transaction-`

- IdMarshaller, 2772
- activemq::wireformat::openwire-
::marshal::generated::Transaction-
InfoMarshaller, 2780
- activemq::wireformat::openwire-
::marshal::generated::Wire-
FormatInfoMarshaller, 2902
- activemq::wireformat::openwire-
::marshal::generated::XA-
TransactionIdMarshaller, 2945
- tightMarshal2
- activemq::wireformat::openwire-
::marshal::BaseDataStream-
Marshaller, 518
- activemq::wireformat::openwire-
::marshal::DataStreamMarshaller,
1129
- activemq::wireformat::openwire-
::marshal::generated::Active-
MQBlobMessageMarshaller,
164
- activemq::wireformat::openwire-
::marshal::generated::Active-
MQBytesMessageMarshaller,
186
- activemq::wireformat::openwire-
::marshal::generated::Active-
MQDestinationMarshaller, 260
- activemq::wireformat::openwire-
::marshal::generated::Active-
MQMapMessageMarshaller,
287
- activemq::wireformat::openwire-
::marshal::generated::Active-
MQMessageMarshaller, 293
- activemq::wireformat::openwire-
::marshal::generated::Active-
MQObjectMessageMarshaller,
306
- activemq::wireformat::openwire-
::marshal::generated::Active-
MQQueueMarshaller, 332
- activemq::wireformat::openwire-
::marshal::generated::Active-
MQStreamMessageMarshaller,
377
- activemq::wireformat::openwire-
::marshal::generated::Active-
MQTempDestinationMarshaller,
385
- activemq::wireformat::openwire-
::marshal::generated::Active-
MQTempQueueMarshaller, 394
- activemq::wireformat::openwire-
::marshal::generated::Active-
MQTempTopicMarshaller, 404
- activemq::wireformat::openwire-
::marshal::generated::ActiveM-
QTextMessageMarshaller, 413
- activemq::wireformat::openwire-
::marshal::generated::Active-
MQTopicMarshaller, 422
- activemq::wireformat::openwire-
::marshal::generated::Base-
CommandMarshaller, 504
- activemq::wireformat::openwire-
::marshal::generated::Broker-
IdMarshaller, 570
- activemq::wireformat::openwire-
::marshal::generated::Broker-
InfoMarshaller, 581
- activemq::wireformat::openwire-
::marshal::generated::Connection-
ControlMarshaller, 946
- activemq::wireformat::openwire-
::marshal::generated::Connection-
ErrorMarshaller, 954
- activemq::wireformat::openwire-
::marshal::generated::Connection-
IdMarshaller, 966
- activemq::wireformat::openwire-
::marshal::generated::Connection-
InfoMarshaller, 977
- activemq::wireformat::openwire-
::marshal::generated::Consumer-
ControlMarshaller, 999
- activemq::wireformat::openwire-
::marshal::generated::Consumer-
IdMarshaller, 1007
- activemq::wireformat::openwire-
::marshal::generated::Consumer-
InfoMarshaller, 1020
- activemq::wireformat::openwire-
::marshal::generated::Control-
CommandMarshaller, 1028
- activemq::wireformat::openwire-
::marshal::generated::Data-
ArrayResponseMarshaller,
1074
- activemq::wireformat::openwire-

- `::marshal::generated::Data-ResponseMarshaller`, 1118
- `activemq::wireformat::openwire-
::marshal::generated::Destination-InfoMarshaller`, 1221
- `activemq::wireformat::openwire-
::marshal::generated::Discovery-EventMarshaller`, 1232
- `activemq::wireformat::openwire-
::marshal::generated::Exception-ResponseMarshaller`, 1293
- `activemq::wireformat::openwire-
::marshal::generated::Flush-CommandMarshaller`, 1386
- `activemq::wireformat::openwire-
::marshal::generated::Integer-ResponseMarshaller`, 1522
- `activemq::wireformat::openwire-
::marshal::generated::Journal-QueueAckMarshaller`, 1567
- `activemq::wireformat::openwire-
::marshal::generated::Journal-TopicAckMarshaller`, 1575
- `activemq::wireformat::openwire-
::marshal::generated::Journal-TraceMarshaller`, 1582
- `activemq::wireformat::openwire-
::marshal::generated::Journal-TransactionMarshaller`, 1590
- `activemq::wireformat::openwire-
::marshal::generated::Keep-AliveInfoMarshaller`, 1597
- `activemq::wireformat::openwire-
::marshal::generated::Last-PartialCommandMarshaller`, 1611
- `activemq::wireformat::openwire-
::marshal::generated::Local-TransactionIdMarshaller`, 1681
- `activemq::wireformat::openwire-
::marshal::generated::Message-AckMarshaller`, 1875
- `activemq::wireformat::openwire-
::marshal::generated::Message-DispatchMarshaller`, 1893
- `activemq::wireformat::openwire-
::marshal::generated::Message-DispatchNotificationMarshaller`, 1902
- `activemq::wireformat::openwire-
::marshal::generated::Message-IdMarshaller`, 1915
- `activemq::wireformat::openwire-
::marshal::generated::Message-Marshaller`, 1920
- `activemq::wireformat::openwire-
::marshal::generated::Message-PullMarshaller`, 1947
- `activemq::wireformat::openwire-
::marshal::generated::Network-BridgeFilterMarshaller`, 1977
- `activemq::wireformat::openwire-
::marshal::generated::Partial-CommandMarshaller`, 2082
- `activemq::wireformat::openwire-
::marshal::generated::Producer-AckMarshaller`, 2178
- `activemq::wireformat::openwire-
::marshal::generated::Producer-IdMarshaller`, 2189
- `activemq::wireformat::openwire-
::marshal::generated::Producer-InfoMarshaller`, 2198
- `activemq::wireformat::openwire-
::marshal::generated::Remove-InfoMarshaller`, 2274
- `activemq::wireformat::openwire-
::marshal::generated::Remove-SubscriptionInfoMarshaller`, 2282
- `activemq::wireformat::openwire-
::marshal::generated::Replay-CommandMarshaller`, 2290
- `activemq::wireformat::openwire-
::marshal::generated::Response-Marshaller`, 2311
- `activemq::wireformat::openwire-
::marshal::generated::Session-IdMarshaller`, 2385
- `activemq::wireformat::openwire-
::marshal::generated::Session-InfoMarshaller`, 2392
- `activemq::wireformat::openwire-
::marshal::generated::Shutdown-InfoMarshaller`, 2437
- `activemq::wireformat::openwire-
::marshal::generated::Subscription-InfoMarshaller`, 2638
- `activemq::wireformat::openwire-
::marshal::generated::Transaction-`

- IdMarshaller, 2773
- activemq::wireformat::openwire-
::marshal::generated::Transaction-
InfoMarshaller, 2781
- activemq::wireformat::openwire-
::marshal::generated::Wire-
FormatInfoMarshaller, 2903
- activemq::wireformat::openwire-
::marshal::generated::XA-
TransactionIdMarshaller, 2945
- tightMarshalBrokerError1
 - activemq::wireformat::openwire-
::marshal::BaseDataStream-
Marshaller, 518
- tightMarshalBrokerError2
 - activemq::wireformat::openwire-
::marshal::BaseDataStream-
Marshaller, 518
- tightMarshalCachedObject1
 - activemq::wireformat::openwire-
::marshal::BaseDataStream-
Marshaller, 519
- tightMarshalCachedObject2
 - activemq::wireformat::openwire-
::marshal::BaseDataStream-
Marshaller, 519
- tightMarshalLong1
 - activemq::wireformat::openwire-
::marshal::BaseDataStream-
Marshaller, 520
- tightMarshalLong2
 - activemq::wireformat::openwire-
::marshal::BaseDataStream-
Marshaller, 520
- tightMarshalNestedObject1
 - activemq::wireformat::openwire-
::marshal::BaseDataStream-
Marshaller, 521
 - activemq::wireformat::openwire::-
OpenWireFormat, 2057
- tightMarshalNestedObject2
 - activemq::wireformat::openwire-
::marshal::BaseDataStream-
Marshaller, 521
 - activemq::wireformat::openwire::-
OpenWireFormat, 2057
- tightMarshalObjectArray1
 - activemq::wireformat::openwire-
::marshal::BaseDataStream-
Marshaller, 522
- tightMarshalObjectArray2
 - activemq::wireformat::openwire-
::marshal::BaseDataStream-
Marshaller, 522
- tightMarshalString1
 - activemq::wireformat::openwire-
::marshal::BaseDataStream-
Marshaller, 523
- tightMarshalString2
 - activemq::wireformat::openwire-
::marshal::BaseDataStream-
Marshaller, 523
- tightUnmarshal
 - activemq::wireformat::openwire-
::marshal::BaseDataStream-
Marshaller, 524
 - activemq::wireformat::openwire-
::marshal::DataStreamMarshaller,
1131
 - activemq::wireformat::openwire-
::marshal::generated::Active-
MQBlobMessageMarshaller,
165
 - activemq::wireformat::openwire-
::marshal::generated::Active-
MQBytesMessageMarshaller,
187
 - activemq::wireformat::openwire-
::marshal::generated::Active-
MQDestinationMarshaller, 261
 - activemq::wireformat::openwire-
::marshal::generated::Active-
MQMapMessageMarshaller,
287
 - activemq::wireformat::openwire-
::marshal::generated::Active-
MQMessageMarshaller, 294
 - activemq::wireformat::openwire-
::marshal::generated::Active-
MQObjectMessageMarshaller,
307
 - activemq::wireformat::openwire-
::marshal::generated::Active-
MQQueueMarshaller, 332
 - activemq::wireformat::openwire-
::marshal::generated::Active-
MQStreamMessageMarshaller,
378
 - activemq::wireformat::openwire-
::marshal::generated::Active-

- MQTempDestinationMarshaller, 385
- activemq::wireformat::openwire-
::marshal::generated::Active-
MQTempQueueMarshaller, 395
- activemq::wireformat::openwire-
::marshal::generated::Active-
MQTempTopicMarshaller, 404
- activemq::wireformat::openwire-
::marshal::generated::ActiveM-
QTextMessageMarshaller, 413
- activemq::wireformat::openwire-
::marshal::generated::Active-
MQTopicMarshaller, 422
- activemq::wireformat::openwire-
::marshal::generated::Base-
CommandMarshaller, 505
- activemq::wireformat::openwire-
::marshal::generated::Broker-
IdMarshaller, 571
- activemq::wireformat::openwire-
::marshal::generated::Broker-
InfoMarshaller, 582
- activemq::wireformat::openwire-
::marshal::generated::Connection-
ControlMarshaller, 947
- activemq::wireformat::openwire-
::marshal::generated::Connection-
ErrorMarshaller, 954
- activemq::wireformat::openwire-
::marshal::generated::Connection-
IdMarshaller, 967
- activemq::wireformat::openwire-
::marshal::generated::Connection-
InfoMarshaller, 977
- activemq::wireformat::openwire-
::marshal::generated::Consumer-
ControlMarshaller, 1000
- activemq::wireformat::openwire-
::marshal::generated::Consumer-
IdMarshaller, 1008
- activemq::wireformat::openwire-
::marshal::generated::Consumer-
InfoMarshaller, 1021
- activemq::wireformat::openwire-
::marshal::generated::Control-
CommandMarshaller, 1029
- activemq::wireformat::openwire-
::marshal::generated::Data-
ArrayResponseMarshaller, 1075
- activemq::wireformat::openwire-
::marshal::generated::Data-
ResponseMarshaller, 1118
- activemq::wireformat::openwire-
::marshal::generated::Destination-
InfoMarshaller, 1221
- activemq::wireformat::openwire-
::marshal::generated::Discovery-
EventMarshaller, 1233
- activemq::wireformat::openwire-
::marshal::generated::Exception-
ResponseMarshaller, 1294
- activemq::wireformat::openwire-
::marshal::generated::Flush-
CommandMarshaller, 1387
- activemq::wireformat::openwire-
::marshal::generated::Integer-
ResponseMarshaller, 1522
- activemq::wireformat::openwire-
::marshal::generated::Journal-
QueueAckMarshaller, 1567
- activemq::wireformat::openwire-
::marshal::generated::Journal-
TopicAckMarshaller, 1576
- activemq::wireformat::openwire-
::marshal::generated::Journal-
TraceMarshaller, 1583
- activemq::wireformat::openwire-
::marshal::generated::Journal-
TransactionMarshaller, 1590
- activemq::wireformat::openwire-
::marshal::generated::Keep-
AliveInfoMarshaller, 1597
- activemq::wireformat::openwire-
::marshal::generated::Last-
PartialCommandMarshaller, 1611
- activemq::wireformat::openwire-
::marshal::generated::Local-
TransactionIdMarshaller, 1682
- activemq::wireformat::openwire-
::marshal::generated::Message-
AckMarshaller, 1876
- activemq::wireformat::openwire-
::marshal::generated::Message-
DispatchMarshaller, 1894
- activemq::wireformat::openwire-
::marshal::generated::Message-
DispatchNotificationMarshaller,

- 1903
- activemq::wireformat::openwire-
::marshal::generated::Message-
IdMarshaller, 1916
- activemq::wireformat::openwire-
::marshal::generated::Message-
Marshaller, 1921
- activemq::wireformat::openwire-
::marshal::generated::Message-
PullMarshaller, 1947
- activemq::wireformat::openwire-
::marshal::generated::Network-
BridgeFilterMarshaller, 1978
- activemq::wireformat::openwire-
::marshal::generated::Partial-
CommandMarshaller, 2083
- activemq::wireformat::openwire-
::marshal::generated::Producer-
AckMarshaller, 2178
- activemq::wireformat::openwire-
::marshal::generated::Producer-
IdMarshaller, 2189
- activemq::wireformat::openwire-
::marshal::generated::Producer-
InfoMarshaller, 2198
- activemq::wireformat::openwire-
::marshal::generated::Remove-
InfoMarshaller, 2274
- activemq::wireformat::openwire-
::marshal::generated::Remove-
SubscriptionInfoMarshaller,
2283
- activemq::wireformat::openwire-
::marshal::generated::Replay-
CommandMarshaller, 2290
- activemq::wireformat::openwire-
::marshal::generated::Response-
Marshaller, 2311
- activemq::wireformat::openwire-
::marshal::generated::Session-
IdMarshaller, 2385
- activemq::wireformat::openwire-
::marshal::generated::Session-
InfoMarshaller, 2393
- activemq::wireformat::openwire-
::marshal::generated::Shutdown-
InfoMarshaller, 2437
- activemq::wireformat::openwire-
::marshal::generated::Subscription-
InfoMarshaller, 2638
- activemq::wireformat::openwire-
::marshal::generated::Transaction-
IdMarshaller, 2773
- activemq::wireformat::openwire-
::marshal::generated::Transaction-
InfoMarshaller, 2781
- activemq::wireformat::openwire-
::marshal::generated::Wire-
FormatInfoMarshaller, 2903
- activemq::wireformat::openwire-
::marshal::generated::XA-
TransactionIdMarshaller, 2946
- tightUnmarshalBrokerError
- activemq::wireformat::openwire-
::marshal::BaseDataStream-
Marshaller, 524
- tightUnmarshalByteArray
- activemq::wireformat::openwire-
::marshal::BaseDataStream-
Marshaller, 524
- tightUnmarshalCachedObject
- activemq::wireformat::openwire-
::marshal::BaseDataStream-
Marshaller, 525
- tightUnmarshalConstByteArray
- activemq::wireformat::openwire-
::marshal::BaseDataStream-
Marshaller, 525
- tightUnmarshalLong
- activemq::wireformat::openwire-
::marshal::BaseDataStream-
Marshaller, 526
- tightUnmarshalNestedObject
- activemq::wireformat::openwire-
::marshal::BaseDataStream-
Marshaller, 526
- activemq::wireformat::openwire::-
OpenWireFormat, 2057
- tightUnmarshalString
- activemq::wireformat::openwire-
::marshal::BaseDataStream-
Marshaller, 527
- time
- gz_header_s, 1399
- timedJoin
- decaf::util::concurrent::TimeUnit,
2759
- timedWait
- decaf::util::concurrent::TimeUnit,
2760

- timeout
 - activemq::commands::Destination-Info, 1218
 - activemq::commands::MessagePull, 1944
- timestamp
 - activemq::commands::Message, 1838
 - decaf::util::UUID, 2883
- toArray
 - activemq::util::ActiveMQProperties, 320
 - cms::CMSProperties, 834
 - decaf::util::AbstractCollection, 119
 - decaf::util::ArrayList, 457
 - decaf::util::Collection, 865
 - decaf::util::concurrent::CopyOn-WriteArrayList, 1047
 - decaf::util::concurrent::CopyOn-WriteArraySet, 1062
 - decaf::util::concurrent::Linked-BlockingQueue, 1632
 - decaf::util::concurrent::Synchronous-Queue, 2665
 - decaf::util::LinkedList, 1658
 - decaf::util::Properties, 2209
 - decaf::util::StlQueue, 2571
 - decaf::util::StringTokenizer, 2630
- toBinaryString
 - decaf::lang::Integer, 1512
 - decaf::lang::Long, 1738
- toByteArray
 - decaf::io::ByteArrayOutputStream, 689
- toDays
 - decaf::util::concurrent::TimeUnit, 2760
- toDegrees
 - decaf::lang::Math, 1817
- toDestinationOption
 - activemq::core::ActiveMQConstants, 233
- toHexFromBytes
 - activemq::wireformat::openwire-::marshal::BaseDataStream-Marshaller, 527
- toHexString
 - decaf::lang::Double, 1245
 - decaf::lang::Float, 1354
 - decaf::lang::Integer, 1512
 - decaf::lang::Long, 1738
- toHours
 - decaf::util::concurrent::TimeUnit, 2761
- toMicros
 - decaf::util::concurrent::TimeUnit, 2761
- toMillis
 - decaf::util::concurrent::TimeUnit, 2761
- toMinutes
 - decaf::util::concurrent::TimeUnit, 2762
- toNanos
 - decaf::util::concurrent::TimeUnit, 2762
- toOctalString
 - decaf::lang::Integer, 1513
 - decaf::lang::Long, 1739
- toRadians
 - decaf::lang::Math, 1817
- toSeconds
 - decaf::util::concurrent::TimeUnit, 2763
- toStream
 - activemq::wireformat::stomp::Stomp-Frame, 2592
- toString
 - activemq::commands::ActiveMQ-BlobMessage, 161
 - activemq::commands::ActiveMQ-BytesMessage, 177
 - activemq::commands::ActiveMQ-Destination, 255
 - activemq::commands::ActiveMQ-MapMessage, 283
 - activemq::commands::ActiveMQ-Message, 290
 - activemq::commands::ActiveMQ-ObjectMessage, 303
 - activemq::commands::ActiveMQ-Queue, 325
 - activemq::commands::ActiveMQ-StreamMessage, 369
 - activemq::commands::ActiveMQ-TempDestination, 382
 - activemq::commands::ActiveMQ-TempQueue, 391
 - activemq::commands::ActiveMQ-TempTopic, 400

- activemq::commands::ActiveMQ-
 TextMessage, 409
- activemq::commands::ActiveMQ-
 Topic, 418
- activemq::commands::BaseCommand,
 498
- activemq::commands::BaseData-
 Structure, 531
- activemq::commands::Boolean-
 Expression, 552
- activemq::commands::BrokerId, 567
- activemq::commands::BrokerInfo,
 577
- activemq::commands::Command,
 870
- activemq::commands::Connection-
 Control, 942
- activemq::commands::Connection-
 Error, 950
- activemq::commands::ConnectionId,
 963
- activemq::commands::Connection-
 Info, 972
- activemq::commands::Consumer-
 Control, 995
- activemq::commands::ConsumerId,
 1004
- activemq::commands::Consumer-
 Info, 1016
- activemq::commands::Control-
 Command, 1025
- activemq::commands::DataArray-
 Response, 1071
- activemq::commands::DataResponse,
 1114
- activemq::commands::DataStructure,
 1138
- activemq::commands::Destination-
 Info, 1217
- activemq::commands::Discovery-
 Event, 1229
- activemq::commands::Exception-
 Response, 1290
- activemq::commands::FlushCommand,
 1382
- activemq::commands::Integer-
 Response, 1518
- activemq::commands::Journal-
 QueueAck, 1563
- activemq::commands::JournalTopic-
 Ack, 1571
- activemq::commands::JournalTrace,
 1579
- activemq::commands::Journal-
 Transaction, 1586
- activemq::commands::KeepAliveInfo,
 1593
- activemq::commands::LastPartial-
 Command, 1607
- activemq::commands::LocalTransaction-
 Id, 1678
- activemq::commands::Message,
 1836
- activemq::commands::MessageAck,
 1871
- activemq::commands::Message-
 Dispatch, 1885
- activemq::commands::Message-
 DispatchNotification, 1898
- activemq::commands::MessageId,
 1911
- activemq::commands::MessagePull,
 1943
- activemq::commands::Network-
 BridgeFilter, 1974
- activemq::commands::Partial-
 Command, 2078
- activemq::commands::ProducerAck,
 2174
- activemq::commands::ProducerId,
 2185
- activemq::commands::ProducerInfo,
 2194
- activemq::commands::RemoveInfo,
 2270
- activemq::commands::Remove-
 SubscriptionInfo, 2278
- activemq::commands::Replay-
 Command, 2286
- activemq::commands::Response,
 2300
- activemq::commands::SessionId,
 2381
- activemq::commands::SessionInfo,
 2389
- activemq::commands::ShutdownInfo,
 2433
- activemq::commands::Subscription-
 Info, 2634
- activemq::commands::TransactionId,

- 2769
- activemq::commands::Transaction-Info, 2777
- activemq::commands::WireFormat-Info, 2899
- activemq::commands::XATransaction-Id, 2942
- activemq::core::ActiveMQConstants, 233
- activemq::state::ConnectionState, 984
- activemq::state::ConsumerState, 1022
- activemq::state::ProducerState, 2199
- activemq::state::SessionState, 2397
- activemq::state::TransactionState, 2786
- activemq::util::ActiveMQProperties, 320
- activemq::util::PrimitiveList, 2125
- activemq::util::PrimitiveMap, 2135
- activemq::util::PrimitiveValueNode, 2159
- activemq::wireformat::openwire-::marshal::BaseDataStream-Mmarshaller, 527, 528
- cms::CMSProperties, 834
- decaf::io::ByteArrayOutputStream, 689
- decaf::io::FilterOutputStream, 1344
- decaf::io::InputStream, 1472
- decaf::io::OutputStream, 2070
- decaf::lang::Boolean, 549
- decaf::lang::Byte, 621, 622
- decaf::lang::Character, 772
- decaf::lang::CharSequence, 805
- decaf::lang::Double, 1246
- decaf::lang::Float, 1355
- decaf::lang::Integer, 1513, 1514
- decaf::lang::Long, 1739, 1740
- decaf::lang::Short, 2406
- decaf::lang::String, 2625
- decaf::lang::Thread, 2713
- decaf::net::InetAddress, 1444
- decaf::net::ServerSocket, 2351
- decaf::net::Socket, 2469
- decaf::net::SocketImpl, 2487
- decaf::net::URI, 2840
- decaf::nio::ByteBuffer, 716
- decaf::nio::CharBuffer, 802
- decaf::nio::DoubleBuffer, 1270
- decaf::nio::FloatBuffer, 1378
- decaf::nio::IntBuffer, 1498
- decaf::nio::LongBuffer, 1763
- decaf::nio::ShortBuffer, 2430
- decaf::security::cert::Certificate, 753
- decaf::util::ArrayList, 458
- decaf::util::concurrent::atomic:-AtomicBoolean, 475
- decaf::util::concurrent::atomic:-AtomicInteger, 481
- decaf::util::concurrent::atomic:-AtomicReference, 486
- decaf::util::concurrent::CopyOn-WriteArrayList, 1047
- decaf::util::concurrent::Linked-BlockingQueue, 1633
- decaf::util::concurrent::locks:-ReentrantLock, 2261
- decaf::util::concurrent::Mutex, 1963
- decaf::util::concurrent::Semaphore, 2338
- decaf::util::concurrent::TimeUnit, 2763
- decaf::util::Date, 1142
- decaf::util::logging::Level, 1619
- decaf::util::Properties, 2209
- decaf::util::UUID, 2883
- toURI
 - activemq::util::CompositeData, 892
- toURIOption
 - activemq::core::ActiveMQConstants, 233
- toURL
 - decaf::net::URI, 2840
- total
 - inflate_state, 1447
- total_in
 - z_stream_s, 2952
- total_out
 - z_stream_s, 2953
- track
 - activemq::state::ConnectionState-Tracker, 990
- trackBack
 - activemq::state::ConnectionState-Tracker, 990
- transaction
 - activemq::core::ActiveMQSession, 355

- transactionId
 - activemq::commands::JournalTopic-Ack, 1572
 - activemq::commands::Journal-Transaction, 1586
 - activemq::commands::Message, 1839
 - activemq::commands::MessageAck, 1872
 - activemq::commands::Transaction-Info, 2778
- transfer
 - decaf::internal::util::concurrent::TransferQueue, 2788
 - decaf::internal::util::concurrent::TransferStack, 2789, 2790
- transportInterrupted
 - activemq::core::ActiveMQConnection, 212
 - activemq::state::ConnectionState-Tracker, 990
 - activemq::transport::DefaultTransport-Listener, 1179
 - activemq::transport::failover::Failover-TransportListener, 1322
 - activemq::transport::TransportFilter, 2809
 - activemq::transport::Transport-Listener, 2811
- transportResumed
 - activemq::core::ActiveMQConnection, 212
 - activemq::transport::DefaultTransport-Listener, 1179
 - activemq::transport::failover::Failover-TransportListener, 1322
 - activemq::transport::TransportFilter, 2809
 - activemq::transport::Transport-Listener, 2812
- tree_desc
 - deflate.h, 3145
- tree_desc_s, 2815
 - dyn_tree, 2815
 - max_code, 2815
 - stat_desc, 2815
- trees.h
 - _dist_code, 3151
 - _length_code, 3151
 - base_dist, 3152
 - base_length, 3152
 - static_dtree, 3152
 - static_ltree, 3152
- trimToSize
 - decaf::util::ArrayList, 458
- tryAcquire
 - decaf::util::concurrent::Semaphore, 2338–2340
- tryLock
 - activemq::core::FifoMessageDispatch-Channel, 1328
 - activemq::core::SimplePriority-MessageDispatchChannel, 2449
 - decaf::internal::util::concurrent::SynchronizableImpl, 2652
 - decaf::io::InputStream, 1473
 - decaf::io::OutputStream, 2070
 - decaf::util::AbstractCollection, 119
 - decaf::util::concurrent::Concurrent-StlMap, 919
 - decaf::util::concurrent::CopyOn-WriteArrayList, 1047
 - decaf::util::concurrent::locks::Lock, 1687, 1688
 - decaf::util::concurrent::locks::ReentrantLock, 2261, 2262
 - decaf::util::concurrent::Mutex, 1963
 - decaf::util::concurrent::Synchronizable, 2643
 - decaf::util::StlMap, 2563
 - decaf::util::StlQueue, 2571
- trylock
 - decaf::internal::util::concurrent::MutexImpl, 1967
- type
 - activemq::commands::Journal-Transaction, 1587
 - activemq::commands::Message, 1839
 - activemq::commands::Transaction-Info, 2778
- ulnt
 - zconf.h, 3153
- ulntf
 - zconf.h, 3153
- uLong
 - zconf.h, 3153
- uLongf
 - zconf.h, 3153

- zconf.h, 3153
- uch
 - zutil.h, 3163
- uchf
 - zutil.h, 3163
- ulg
 - zutil.h, 3163
- uncaughtException
 - decaf::lang::Thread::UncaughtExceptionHandler, 2816
- unlock
 - activemq::core::FifoMessageDispatchChannel, 1328
 - activemq::core::SimplePriorityMessageDispatchChannel, 2450
 - decaf::internal::util::concurrent::MutexImpl, 1968
 - decaf::internal::util::concurrent::SynchronizableImpl, 2652
 - decaf::io::InputStream, 1473
 - decaf::io::OutputStream, 2071
 - decaf::util::AbstractCollection, 120
 - decaf::util::concurrent::ConcurrentStlMap, 919
 - decaf::util::concurrent::CopyOnWriteArrayList, 1048
 - decaf::util::concurrent::Lock, 1683
 - decaf::util::concurrent::locks::Lock, 1689
 - decaf::util::concurrent::locks::ReentrantLock, 2263
 - decaf::util::concurrent::Mutex, 1963
 - decaf::util::concurrent::Synchronizable, 2645
 - decaf::util::StlMap, 2563
 - decaf::util::StlQueue, 2572
- unmarshal
 - activemq::wireformat::openwire::marshal::PrimitiveTypesMarshaller, 2139, 2140
 - activemq::wireformat::openwire::OpenWireFormat, 2058
 - activemq::wireformat::openwire::utils::BooleanStream, 555
 - activemq::wireformat::stomp::StompWireFormat, 2600
 - activemq::wireformat::WireFormat, 2887
- unmarshalList
 - activemq::wireformat::openwire::marshal::PrimitiveTypesMarshaller, 2140
- unmarshalMap
 - activemq::wireformat::openwire::marshal::PrimitiveTypesMarshaller, 2141
- unmarshalPrimitive
 - activemq::wireformat::openwire::marshal::PrimitiveTypesMarshaller, 2141
- unmarshalPrimitiveList
 - activemq::wireformat::openwire::marshal::PrimitiveTypesMarshaller, 2141
- unmarshalPrimitiveMap
 - activemq::wireformat::openwire::marshal::PrimitiveTypesMarshaller, 2142
- unpark
 - decaf::util::concurrent::locks::LockSupport, 1692
- unread
 - decaf::io::PushbackInputStream, 2220
- unregisterAllFactories
 - activemq::transport::TransportRegistry, 2814
 - activemq::wireformat::WireFormatRegistry, 2907
- unregisterFactory
 - activemq::transport::TransportRegistry, 2814
 - activemq::wireformat::WireFormatRegistry, 2907
- unsetenv
 - decaf::lang::System, 2675
- unsubscribe
 - activemq::cmsutil::PooledSession, 2106
 - activemq::core::ActiveMQSession, 353
 - cms::Session, 2376
- update
 - decaf::util::zip::Adler32, 440, 441
 - decaf::util::zip::Checksum, 811, 812
 - decaf::util::zip::CRC32, 1066, 1067
- updateURLs
 - activemq::transport::failover::FailoverTransport, 1317

- activemq::transport::IOTransport, 1556
- activemq::transport::mock::MockTransport, 1958
- activemq::transport::Transport, 2798
- activemq::transport::TransportFilter, 2809
- uriParams
 - activemq::core::ActiveMQConstants::StaticInitializer, 2536
- uriParamsMap
 - activemq::core::ActiveMQConstants::StaticInitializer, 2536
- userID
 - activemq::commands::Message, 1839
- userName
 - activemq::commands::ConnectionInfo, 973
- ush
 - zutil.h, 3163
- ushf
 - zutil.h, 3163
- val
 - code, 851
- valid
 - decaf::io::FileDescriptor, 1332
- validate
 - decaf::internal::net::URIEncoderDecoder, 2843
- validateAuthority
 - decaf::internal::net::URIHelper, 2848
- validateFragment
 - decaf::internal::net::URIHelper, 2849
- validatePath
 - decaf::internal::net::URIHelper, 2849
- validateQuery
 - decaf::internal::net::URIHelper, 2849
- validateScheme
 - decaf::internal::net::URIHelper, 2850
- validateSimple
 - decaf::internal::net::URIEncoderDecoder, 2843
- validateSsp
 - decaf::internal::net::URIHelper, 2850
- validateUserinfo
 - decaf::internal::net::URIHelper, 2850
- value
 - activemq::commands::BrokerId, 567
 - activemq::commands::ConnectionId, 963
 - activemq::commands::ConsumerId, 1004
 - activemq::commands::LocalTransactionId, 1678
 - activemq::commands::ProducerId, 2186
 - activemq::commands::SessionId, 2382
- valueOf
 - decaf::lang::Boolean, 550
 - decaf::lang::Byte, 622, 623
 - decaf::lang::Character, 772
 - decaf::lang::Double, 1247
 - decaf::lang::Float, 1355, 1356
 - decaf::lang::Integer, 1514, 1515
 - decaf::lang::Long, 1740, 1741
 - decaf::lang::Short, 2407
 - decaf::lang::String, 2625–2627
 - decaf::util::concurrent::TimeUnit, 2763
- values
 - decaf::util::concurrent::ConcurrentStlMap, 919
 - decaf::util::concurrent::TimeUnit, 2764
 - decaf::util::Map, 1779
 - decaf::util::StlMap, 2563
- variant
 - decaf::util::UUID, 2883
- verify
 - decaf::security::cert::Certificate, 753, 754
- version
 - decaf::util::UUID, 2884
- visit
 - activemq::commands::BrokerError, 562
 - activemq::commands::BrokerInfo, 577
 - activemq::commands::Command, 871
 - activemq::commands::ConnectionControl, 942
 - activemq::commands::ConnectionError, 950
 - activemq::commands::ConnectionInfo, 972

- activemq::commands::Consumer-
Control, 995
- activemq::commands::Consumer-
Info, 1016
- activemq::commands::Control-
Command, 1025
- activemq::commands::Destination-
Info, 1217
- activemq::commands::FlushCommand,
1383
- activemq::commands::KeepAliveInfo,
1593
- activemq::commands::Message,
1837
- activemq::commands::MessageAck,
1872
- activemq::commands::Message-
Dispatch, 1885
- activemq::commands::Message-
DispatchNotification, 1898
- activemq::commands::MessagePull,
1943
- activemq::commands::ProducerAck,
2174
- activemq::commands::ProducerInfo,
2194
- activemq::commands::RemoveInfo,
2270
- activemq::commands::Remove-
SubscriptionInfo, 2279
- activemq::commands::Replay-
Command, 2286
- activemq::commands::Response,
2301
- activemq::commands::SessionInfo,
2389
- activemq::commands::ShutdownInfo,
2433
- activemq::commands::Transaction-
Info, 2777
- activemq::commands::WireFormat-
Info, 2899
- voidp
zconf.h, 3153
- voidpc
zconf.h, 3153
- voidpf
zconf.h, 3153
- w_bits
- internal_state, 1526
- w_mask
internal_state, 1527
- w_size
internal_state, 1527
- wait
 - activemq::core::FifoMessageDispatch-
Channel, 1328, 1329
 - activemq::core::SimplePriority-
MessageDispatchChannel,
2450, 2451
 - decaf::internal::util::concurrent::-
ConditionImpl, 930
 - decaf::internal::util::concurrent::-
SynchronizableImpl, 2652, 2653
 - decaf::io::InputStream, 1473, 1474
 - decaf::io::OutputStream, 2071
 - decaf::util::AbstractCollection, 120,
121
 - decaf::util::concurrent::Concurrent-
StlMap, 920
 - decaf::util::concurrent::CopyOn-
WriteArrayList, 1048, 1049
 - decaf::util::concurrent::Mutex, 1964
 - decaf::util::concurrent::Synchronizable,
2646–2648
 - decaf::util::StlMap, 2564
 - decaf::util::StlQueue, 2572, 2573
- waitForSpace
 - activemq::util::MemoryUsage, 1821
 - activemq::util::Usage, 2874
- waitForTransportInterruptProcessing-
ToComplete
 - activemq::core::ActiveMQConnection,
213
- wakeup
 - activemq::core::ActiveMQSession,
353
 - activemq::core::ActiveMQSession-
Executor, 359
 - activemq::threads::CompositeTask-
Runner, 896
 - activemq::threads::DedicatedTask-
Runner, 1145
 - activemq::threads::TaskRunner,
2678
- want
gz_state, 1401
- warn

- decaf::util::logging::SimpleLogger, 2443
- warning
 - decaf::util::logging::Logger, 1705
- was
 - inflate_state, 1447
- wasPrepared
 - activemq::commands::JournalTransaction, 1587
- wbits
 - inflate_state, 1447
- what
 - cms::CMSException, 829
 - decaf::lang::Exception, 1286
- whave
 - inflate_state, 1447
- window
 - inflate_state, 1447
 - internal_state, 1527
- window_size
 - internal_state, 1527
- windowSize
 - activemq::commands::ProducerInfo, 2194
- wnext
 - inflate_state, 1447
- work
 - inflate_state, 1447
- wrap
 - decaf::nio::ByteBuffer, 716, 717
 - decaf::nio::CharBuffer, 802
 - decaf::nio::DoubleBuffer, 1270, 1271
 - decaf::nio::FloatBuffer, 1378, 1379
 - decaf::nio::IntBuffer, 1498, 1499
 - decaf::nio::LongBuffer, 1764
 - decaf::nio::ShortBuffer, 2430, 2431
 - inflate_state, 1447
 - internal_state, 1527
- write
 - decaf::internal::net::ssl::openssl::OpenSSLSocket, 2028
 - decaf::internal::net::tcp::TcpSocket, 2687
 - decaf::internal::util::ByteArrayAdapter, 645
 - decaf::io::OutputStream, 2072, 2073
 - decaf::io::Writer, 2912, 2913
- writeBoolean
 - activemq::commands::ActiveMQBytesMessage, 178
- activemq::commands::ActiveMQStreamMessage, 369
- activemq::wireformat::openwire::utils::BooleanStream, 555
- cms::BytesMessage, 729
- cms::StreamMessage, 2615
- decaf::io::DataOutput, 1104
- decaf::io::DataOutputStream, 1111
- writeByte
 - activemq::commands::ActiveMQBytesMessage, 178
 - activemq::commands::ActiveMQStreamMessage, 370
 - cms::BytesMessage, 729
 - cms::StreamMessage, 2615
 - decaf::io::DataOutput, 1105
 - decaf::io::DataOutputStream, 1111
- writeBytes
 - activemq::commands::ActiveMQBytesMessage, 179
 - activemq::commands::ActiveMQStreamMessage, 370, 371
 - cms::BytesMessage, 730
 - cms::StreamMessage, 2616
 - decaf::io::DataOutput, 1105
 - decaf::io::DataOutputStream, 1111
- writeChar
 - activemq::commands::ActiveMQBytesMessage, 179
 - activemq::commands::ActiveMQStreamMessage, 371
 - cms::BytesMessage, 731
 - cms::StreamMessage, 2617
 - decaf::io::DataOutput, 1105
 - decaf::io::DataOutputStream, 1111
- writeChars
 - decaf::io::DataOutput, 1106
 - decaf::io::DataOutputStream, 1111
- writeDouble
 - activemq::commands::ActiveMQBytesMessage, 180
 - activemq::commands::ActiveMQStreamMessage, 371
 - cms::BytesMessage, 731
 - cms::StreamMessage, 2617
 - decaf::io::DataOutput, 1106
 - decaf::io::DataOutputStream, 1111
- writeFloat
 - activemq::commands::ActiveMQBytesMessage, 180

- activemq::commands::ActiveMQ-StreamMessage, 372
- cms::BytesMessage, 731
- cms::StreamMessage, 2618
- decaf::io::DataOutput, 1106
- decaf::io::DataOutputStream, 1111
- writeInt
 - activemq::commands::ActiveMQ-BytesMessage, 180
 - activemq::commands::ActiveMQ-StreamMessage, 372
 - cms::BytesMessage, 732
 - cms::StreamMessage, 2618
 - decaf::io::DataOutput, 1107
 - decaf::io::DataOutputStream, 1111
- writeLock
 - decaf::util::concurrent::locks::ReadWriteLock, 2248
- writeLong
 - activemq::commands::ActiveMQ-BytesMessage, 181
 - activemq::commands::ActiveMQ-StreamMessage, 373
 - cms::BytesMessage, 732
 - cms::StreamMessage, 2618
 - decaf::io::DataOutput, 1107
 - decaf::io::DataOutputStream, 1111
- writeShort
 - activemq::commands::ActiveMQ-BytesMessage, 181
 - activemq::commands::ActiveMQ-StreamMessage, 373
 - cms::BytesMessage, 733
 - cms::StreamMessage, 2619
 - decaf::io::DataOutput, 1107
 - decaf::io::DataOutputStream, 1111
- writeString
 - activemq::commands::ActiveMQ-BytesMessage, 182
 - activemq::commands::ActiveMQ-StreamMessage, 373
 - activemq::util::MarshallingSupport, 1799
 - cms::BytesMessage, 733
 - cms::StreamMessage, 2619
- writeString16
 - activemq::util::MarshallingSupport, 1800
- writeString32
 - activemq::util::MarshallingSupport, 1800
- writeTo
 - decaf::io::ByteArrayOutputStream, 690
- writeUTF
 - activemq::commands::ActiveMQ-BytesMessage, 182
 - cms::BytesMessage, 734
 - decaf::io::DataOutput, 1108
 - decaf::io::DataOutputStream, 1112
- writeUnsignedShort
 - activemq::commands::ActiveMQ-BytesMessage, 182
 - activemq::commands::ActiveMQ-StreamMessage, 374
 - cms::BytesMessage, 733
 - cms::StreamMessage, 2620
 - decaf::io::DataOutput, 1108
 - decaf::io::DataOutputStream, 1112
- written
 - decaf::io::DataOutputStream, 1112
- wsize
 - inflate_state, 1448
- xflags
 - gz_header_s, 1399
- yield
 - decaf::lang::Thread, 2713
- z_errmsg
 - zutil.h, 3163
- z_off64_t
 - zconf.h, 3153
- z_stream
 - zlib.h, 3158
- z_stream_s, 2952
 - adler, 2952
 - avail_in, 2952
 - avail_out, 2952
 - data_type, 2952
 - msg, 2952
 - next_in, 2952
 - next_out, 2952
 - opaque, 2952
 - reserved, 2952
 - state, 2952
 - total_in, 2952
 - total_out, 2953

- zalloc, 2953
- zfree, 2953
- z_streamp
 - zlib.h, 3158
- zalloc
 - z_stream_s, 2953
- zconf.h
 - Byte, 3153
 - Bytef, 3153
 - charf, 3153
 - intf, 3153
 - uInt, 3153
 - uIntf, 3153
 - uLong, 3153
 - uLongf, 3153
 - voidp, 3153
 - voidpc, 3153
 - voidpf, 3153
 - z_off64_t, 3153
- zfree
 - z_stream_s, 2953
- zlib.h
 - OF, 3158–3161
 - ZLIB_VERNUM, 3158
 - ZLIB_VERSION, 3158
 - ZLIB_VER_MAJOR, 3158
 - ZLIB_VER_MINOR, 3158
 - ZLIB_VER_REVISION, 3158
 - ZLIB_VER_SUBREVISION, 3158
 - Z_ASCII, 3157
 - Z_BEST_COMPRESSION, 3157
 - Z_BEST_SPEED, 3157
 - Z_BINARY, 3157
 - Z_BLOCK, 3157
 - Z_BUF_ERROR, 3157
 - Z_DATA_ERROR, 3157
 - Z_DEFAULT_COMPRESSION, 3157
 - Z_DEFAULT_STRATEGY, 3157
 - Z_DEFLATED, 3157
 - Z_ERRNO, 3157
 - Z_FILTERED, 3157
 - Z_FINISH, 3157
 - Z_FIXED, 3157
 - Z_FULL_FLUSH, 3157
 - Z_HUFFMAN_ONLY, 3157
 - Z_MEM_ERROR, 3157
 - Z_NEED_DICT, 3157
 - Z_NO_COMPRESSION, 3157
 - Z_NO_FLUSH, 3157
 - Z_NULL, 3157
 - Z_OK, 3157
 - Z_PARTIAL_FLUSH, 3158
 - Z_RLE, 3158
 - Z_STREAM_END, 3158
 - Z_STREAM_ERROR, 3158
 - Z_SYNC_FLUSH, 3158
 - Z_TEXT, 3158
 - Z_TREES, 3158
 - Z_UNKNOWN, 3158
 - Z_VERSION_ERROR, 3158
 - deflateInit, 3156
 - deflateInit2, 3156
 - gz_header, 3158
 - gz_headerp, 3158
 - gzFile, 3158
 - inflateBackInit, 3156
 - inflateInit, 3156
 - inflateInit2, 3157
 - z_stream, 3158
 - z_streamp, 3158
 - zlib_version, 3158
- zlib_version
 - zlib.h, 3158
- zstrerror
 - gzguts.h, 3147
- zutil.h
 - Assert, 3162
 - DEF_MEM_LEVEL, 3162
 - DYN_TREES, 3162
 - ERR_MSG, 3162
 - ERR_RETURN, 3162
 - MAX_MATCH, 3162
 - MIN_MATCH, 3162
 - OF, 3163
 - PRESET_DICT, 3162
 - STATIC_TREES, 3162
 - STORED_BLOCK, 3162
 - TRY_FREE, 3163
 - Trace, 3162
 - Tracec, 3162
 - Tracecv, 3162
 - Tracev, 3162
 - Tracevv, 3162
 - ZALLOC, 3163
 - ZFREE, 3163
 - ZLIB_INTERNAL, 3163
 - uch, 3163
 - uchf, 3163
 - ulg, 3163

ush, 3163
ushf, 3163
z_errmsg, 3163