

# globus gsi proxy core

## 4.4

Generated by Doxygen 1.6.1

Fri Apr 16 17:47:03 2010

# Contents

<b>1</b>	<b>Globus GSI Proxy API</b>	<b>1</b>
<b>2</b>	<b>Module Index</b>	<b>2</b>
2.1	Modules . . . . .	2
<b>3</b>	<b>Module Documentation</b>	<b>2</b>
3.1	Activation . . . . .	2
3.1.1	Detailed Description . . . . .	2
3.1.2	Define Documentation . . . . .	2
3.2	Handle Management . . . . .	3
3.2.1	Detailed Description . . . . .	5
3.2.2	Typedef Documentation . . . . .	5
3.2.3	Function Documentation . . . . .	5
3.3	Handle Attributes . . . . .	13
3.3.1	Detailed Description . . . . .	15
3.3.2	Typedef Documentation . . . . .	15
3.3.3	Function Documentation . . . . .	15
3.4	Proxy Operations . . . . .	19
3.4.1	Detailed Description . . . . .	20
3.4.2	Function Documentation . . . . .	20
3.5	Proxy Constants . . . . .	22
3.5.1	Enumeration Type Documentation . . . . .	23

## 1 Globus GSI Proxy API

The globus\_gsi\_proxy library is motivated by the desire to provide a abstraction layer for the proxy creation and delegation process. For background on this process please refer to the proxy certificate profile draft.

Any program that uses Globus GSI Proxy functions must include "globus\_gsi\_proxy.h".

We envision the API being used in the following manner:

Delegator:	Delegatee:
	set desired cert info extension in the handle by using the handle set functions.
	globus_gsi_proxy_create_req
globus_gsi_proxy_inquire_req	
modify cert info extension by using handle set/get/clear functions.	
globus_gsi_proxy_sign_req	
	globus_gsi_proxy_assemble_cred

## 2 Module Index

### 2.1 Modules

Here is a list of all modules:

<b>Activation</b>	<b><a href="#">2</a></b>
<b>Handle Management</b>	<b><a href="#">3</a></b>
<b>Handle Attributes</b>	<b><a href="#">13</a></b>
<b>Proxy Operations</b>	<b><a href="#">19</a></b>
<b>Proxy Constants</b>	<b><a href="#">22</a></b>

## 3 Module Documentation

### 3.1 Activation

Globus GSI Proxy uses standard Globus module activation and deactivation.

#### Defines

- `#define` [GLOBUS\\_GSI\\_PROXY\\_MODULE](#)

#### 3.1.1 Detailed Description

Globus GSI Proxy uses standard Globus module activation and deactivation. Before any Globus GSI Proxy functions are called, the following function must be called:

```
globus_module_activate(GLOBUS_GSI_PROXY_MODULE)
```

This function returns `GLOBUS_SUCCESS` if Globus GSI Proxy was successfully initialized, and you are therefore allowed to subsequently call Globus GSI Proxy functions. Otherwise, an error code is returned, and Globus GSI Proxy functions should not be subsequently called. This function may be called multiple times.

To deactivate Globus GSI Proxy, the following function must be called:

```
globus_module_deactivate(GLOBUS_GSI_PROXY_MODULE)
```

This function should be called once for each time Globus GSI Proxy was activated.

#### 3.1.2 Define Documentation

##### 3.1.2.1 `#define` `GLOBUS_GSI_PROXY_MODULE`

Module descriptor.

## 3.2 Handle Management

Create/Destroy/Modify a GSI Proxy Handle.

### Typedefs

- typedef struct globus\_l\_gsi\_proxy\_handle\_s \* globus\_gsi\_proxy\_handle\_t

### Initialize and Destroy

- globus\_result\_t globus\_gsi\_proxy\_handle\_init (globus\_gsi\_proxy\_handle\_t \*handle, globus\_gsi\_proxy\_handle\_attr\_t handle\_attr)
- globus\_result\_t globus\_gsi\_proxy\_handle\_destroy (globus\_gsi\_proxy\_handle\_t handle)

### Get/Set Request

- globus\_result\_t globus\_gsi\_proxy\_handle\_get\_req (globus\_gsi\_proxy\_handle\_t handle, X509\_REQ \*\*req)
- globus\_result\_t globus\_gsi\_proxy\_handle\_set\_req (globus\_gsi\_proxy\_handle\_t handle, X509\_REQ \*req)

### Get/Set Private Key

- globus\_result\_t globus\_gsi\_proxy\_handle\_get\_private\_key (globus\_gsi\_proxy\_handle\_t handle, EVP\_PKEY \*\*proxy\_key)
- globus\_result\_t globus\_gsi\_proxy\_handle\_set\_private\_key (globus\_gsi\_proxy\_handle\_t handle, EVP\_PKEY \*proxy\_key)

### Get/Set Proxy Type

- globus\_result\_t globus\_gsi\_proxy\_handle\_get\_type (globus\_gsi\_proxy\_handle\_t handle, globus\_gsi\_cert\_utils\_cert\_type\_t \*type)
- globus\_result\_t globus\_gsi\_proxy\_handle\_set\_type (globus\_gsi\_proxy\_handle\_t handle, globus\_gsi\_cert\_utils\_cert\_type\_t type)

### Get/Set Policy

- globus\_result\_t globus\_gsi\_proxy\_handle\_set\_policy (globus\_gsi\_proxy\_handle\_t handle, unsigned char \*policy\_data, int policy\_length, int policy\_language\_NID)
- globus\_result\_t globus\_gsi\_proxy\_handle\_get\_policy (globus\_gsi\_proxy\_handle\_t handle, unsigned char \*\*policy\_data, int \*policy\_length, int \*policy\_NID)

### Get/Set Path Length

- globus\_result\_t globus\_gsi\_proxy\_handle\_set\_pathlen (globus\_gsi\_proxy\_handle\_t handle, long pathlen)
- globus\_result\_t globus\_gsi\_proxy\_handle\_get\_pathlen (globus\_gsi\_proxy\_handle\_t handle, int \*pathlen)

### Get/Set Time Valid

- globus\_result\_t globus\_gsi\_proxy\_handle\_get\_time\_valid (globus\_gsi\_proxy\_handle\_t handle, int \*time\_valid)

- globus\_result\_t [globus\\_gsi\\_proxy\\_handle\\_set\\_time\\_valid](#) (globus\_gsi\_proxy\_handle\_t handle, int time\_valid)

### Clear Cert Info

- globus\_result\_t [globus\\_gsi\\_proxy\\_handle\\_clear\\_cert\\_info](#) (globus\_gsi\_proxy\_handle\_t handle)

### Get/Set Cert Info

- globus\_result\_t [globus\\_gsi\\_proxy\\_handle\\_get\\_proxy\\_cert\\_info](#) (globus\_gsi\_proxy\_handle\_t handle, PROXYCERTINFO \*\*pci)
- globus\_result\_t [globus\\_gsi\\_proxy\\_handle\\_set\\_proxy\\_cert\\_info](#) (globus\_gsi\_proxy\_handle\_t handle, PROXYCERTINFO \*pci)

### Get Signing Algorithm

- globus\_result\_t [globus\\_gsi\\_proxy\\_handle\\_get\\_signing\\_algorithm](#) (globus\_gsi\_proxy\_handle\_t handle, EVP\_MD \*\*signing\_algorithm)

### Get Key Bits

- globus\_result\_t [globus\\_gsi\\_proxy\\_handle\\_get\\_keybits](#) (globus\_gsi\_proxy\_handle\_t handle, int \*key\_bits)

### Get Init Prime

- globus\_result\_t [globus\\_gsi\\_proxy\\_handle\\_get\\_init\\_prime](#) (globus\_gsi\_proxy\_handle\_t handle, int \*init\_prime)

### Get Clock Skew

- globus\_result\_t [globus\\_gsi\\_proxy\\_handle\\_get\\_clock\\_skew\\_allowable](#) (globus\_gsi\_proxy\_handle\_t handle, int \*skew)

### Get Callback for Creating Keys

- globus\_result\_t [globus\\_gsi\\_proxy\\_handle\\_get\\_key\\_gen\\_callback](#) (globus\_gsi\_proxy\_handle\_t handle, void(\*\*callback)(int, int, void \*))

### Get/Set Proxy Common Name

- globus\_result\_t [globus\\_gsi\\_proxy\\_handle\\_get\\_common\\_name](#) (globus\_gsi\_proxy\_handle\_t handle, char \*\*common\_name)
- globus\_result\_t [globus\\_gsi\\_proxy\\_handle\\_set\\_common\\_name](#) (globus\_gsi\_proxy\_handle\_t handle, char \*common\_name)

### Set/Check Proxy Is Limited

- globus\_result\_t [globus\\_gsi\\_proxy\\_handle\\_set\\_is\\_limited](#) (globus\_gsi\_proxy\_handle\_t handle, globus\_bool\_t is\_limited)

- `globus_result_t globus_gsi_proxy_is_limited (globus_gsi_proxy_handle_t handle, globus_bool_t *is_limited)`

### 3.2.1 Detailed Description

Create/Destroy/Modify a GSI Proxy Handle. Within the Globus GSI Proxy Library, all proxy operations require a handle parameter. Currently, only one proxy operation may be in progress at once per proxy handle.

This section defines operations to create, modify and destroy GSI Proxy handles.

### 3.2.2 Typedef Documentation

#### 3.2.2.1 `typedef struct globus_l_gsi_proxy_handle_s* globus_gsi_proxy_handle_t`

GSI Proxy Handle. An GSI Proxy handle is used to associate state with a group of operations. Handles can have immutable [attributes](#) associated with them. All proxy [operations](#) take a handle pointer as a parameter.

See also:

[globus\\_gsi\\_proxy\\_handle\\_init\(\)](#), [globus\\_gsi\\_proxy\\_handle\\_destroy\(\)](#), [Handle Attributes](#)

### 3.2.3 Function Documentation

#### 3.2.3.1 `globus_result_t globus_gsi_proxy_handle_init (globus_gsi_proxy_handle_t * handle, globus_gsi_proxy_handle_attrs_t handle_attrs)`

Initialize a GSI Proxy handle. Initialize a proxy handle which can be used in subsequent operations. The handle may only be used in one sequence of operations at a time.

Parameters:

*handle* A pointer to the handle to be initialized. If the handle is originally NULL, space is allocated for it. Otherwise, the current values of the handle are overwritten.

*handle\_attrs* Initial attributes to be used to create this handle.

Returns:

GLOBUS\_SUCCESS unless an error occurred, in which case, a globus error object ID is returned

See also:

[globus\\_gsi\\_proxy\\_handle\\_destroy\(\)](#)

#### 3.2.3.2 `globus_result_t globus_gsi_proxy_handle_get_req (globus_gsi_proxy_handle_t handle, X509_REQ ** req)`

Get the certificate request from a GSI Proxy handle.

Parameters:

*handle* The handle from which to get the certificate request

*req* Parameter used to return the request. It is the users responsibility to free the returned request.

**Returns:**

GLOBUS\_SUCCESS unless an error occurred, in which case, a globus error object ID is returned

**See also:**

[globus\\_gsi\\_proxy\\_handle\\_set\\_req\(\)](#)

**3.2.3.3 globus\_result\_t globus\_gsi\_proxy\_handle\_get\_private\_key (globus\_gsi\_proxy\_handle\_t *handle*, EVP\_PKEY \*\* *proxy\_key*)**

Get the private key from a GSI Proxy handle.

**Parameters:**

*handle* The handle from which to get the private key

*proxy\_key* Parameter used to return the key. It is the users responsibility to free the returned key.

**Returns:**

GLOBUS\_SUCCESS unless an error occurred, in which case, a globus error object ID is returned

**See also:**

[globus\\_gsi\\_proxy\\_handle\\_set\\_private\\_key\(\)](#)

**3.2.3.4 globus\_result\_t globus\_gsi\_proxy\_handle\_get\_type (globus\_gsi\_proxy\_handle\_t *handle*, globus\_gsi\_cert\_utils\_cert\_type\_t \* *type*)**

Determine the type of proxy that will be generated when using this handle.

**Parameters:**

*handle* The handle from which to get the type

*type* Parameter used to return the type.

**Returns:**

GLOBUS\_SUCCESS unless an error occurred, in which case, a globus error object ID is returned

**See also:**

[globus\\_gsi\\_proxy\\_handle\\_set\\_type\(\)](#)

**3.2.3.5 globus\_result\_t globus\_gsi\_proxy\_handle\_set\_policy (globus\_gsi\_proxy\_handle\_t *handle*, unsigned char \* *policy\_data*, int *policy\_length*, int *policy\_language\_NID*)**

Set the policy to be used in the GSI Proxy handle. This function sets the policy to be used in the proxy cert info extension.

**Parameters:**

*handle* The handle to be modified.  
*policy\_data* The policy data.  
*policy\_length* The length of the policy data  
*policy\_language\_NID* The NID of the policy language.

**Returns:**

GLOBUS\_SUCCESS if the handle and its associated fields are valid otherwise an error is returned

**See also:**

[globus\\_gsi\\_proxy\\_handle\\_get\\_policy\(\)](#)

**3.2.3.6 globus\_result\_t globus\_gsi\_proxy\_handle\_set\_pathlen (globus\_gsi\_proxy\_handle\_t *handle*, long *pathlen*)**

Set the path length to be used in the GSI Proxy handle. This function sets the path length to be used in the proxy cert info extension.

**Parameters:**

*handle* The handle to be modified.  
*pathlen* The maximum allowable path length

**Returns:**

GLOBUS\_SUCCESS if the handle is valid, otherwise an error is returned

**See also:**

[globus\\_gsi\\_proxy\\_handle\\_get\\_pathlen\(\)](#)

**3.2.3.7 globus\_result\_t globus\_gsi\_proxy\_handle\_get\_time\_valid (globus\_gsi\_proxy\_handle\_t *handle*, int \* *time\_valid*)**

Get the validity time of the proxy.

**Parameters:**

*handle* The proxy handle to get the expiration date of  
*time\_valid* expiration date of the proxy handle

**Returns:**

GLOBUS\_SUCCESS unless an error occurred, in which case, a globus error object ID is returned



### 3.2.3.8 `globus_result_t globus_gsi_proxy_handle_clear_cert_info (globus_gsi_proxy_handle_t handle)`

Clear the proxy cert info extension stored in the GSI Proxy handle. This function clears proxy cert info extension related setting in the GSI Proxy handle.

#### Parameters:

*handle* The handle for which to clear the proxy cert info extension.

#### Returns:

GLOBUS\_SUCCESS if the handle is valid, otherwise an error is returned

### 3.2.3.9 `globus_result_t globus_gsi_proxy_handle_get_proxy_cert_info (globus_gsi_proxy_handle_t handle, PROXYCERTINFO **pci)`

Get the proxy cert info extension stored in the GSI Proxy handle. This function retrieves the proxy cert info extension from the GSI Proxy handle.

#### Parameters:

*handle* The handle from which to get the proxy cert info extension.

*pci* Contains the proxy cert info extension upon successful return. If the handle does not contain a pci extension, this parameter will be NULL upon return.

#### Returns:

GLOBUS\_SUCCESS upon success GLOBUS\_GSI\_PROXY\_ERROR\_WITH\_HANDLE if handle is invalid  
GLOBUS\_GSI\_PROXY\_ERROR\_WITH\_PROXYCERTINFO if the pci pointer is invalid or if the get failed.

### 3.2.3.10 `globus_result_t globus_gsi_proxy_handle_get_signing_algorithm (globus_gsi_proxy_handle_t handle, EVP_MD **signing_algorithm)`

Get the signing algorithm used to sign the proxy cert request.

#### Parameters:

*handle* The proxy handle containing the type of signing algorithm used

*signing\_algorithm* signing algorithm of the proxy handle

#### Returns:

GLOBUS\_SUCCESS unless an error occurred, in which case, a globus error object ID is returned GLOBUS\_SUCCESS

### 3.2.3.11 `globus_result_t globus_gsi_proxy_handle_get_keybits (globus_gsi_proxy_handle_t handle, int *key_bits)`

Get the key bits used for the pub/private key pair of the proxy.

**Parameters:**

*handle* The proxy handle to get the key bits of  
*key\_bits* key bits of the proxy handle

**Returns:**

GLOBUS\_SUCCESS unless an error occurred, in which case, a globus error object ID is returned GLOBUS\_SUCCESS

**3.2.3.12 globus\_result\_t globus\_gsi\_proxy\_handle\_get\_init\_prime (globus\_gsi\_proxy\_handle\_t *handle*, int \* *init\_prime*)**

Get the init prime of the proxy handle.

**Parameters:**

*handle* The handle to get the init prime used in generating the key pair  
*init\_prime* The resulting init prime

**Returns:**

GLOBUS\_SUCCESS unless an error occurred, in which case an error object identifier (in the form of a globus\_result\_t) is returned

**3.2.3.13 globus\_result\_t globus\_gsi\_proxy\_handle\_get\_clock\_skew\_allowable (globus\_gsi\_proxy\_handle\_t *handle*, int \* *skew*)**

Get the clock skew of the proxy handle.

**Parameters:**

*handle* The handle to get the clock skew of  
*skew* The resulting clock skew

**Returns:**

GLOBUS\_SUCCESS unless an error occurred, in which case an error object identifier (in the form of a globus\_result\_t) is returned

**3.2.3.14 globus\_result\_t globus\_gsi\_proxy\_handle\_get\_key\_gen\_callback (globus\_gsi\_proxy\_handle\_t *handle*, void(\*\*)(int, int, void \*) *callback*)**

Get the callback for creating the public/private key pair.

**Parameters:**

*handle* The proxy handle to get the callback from  
*callback* Parameter used for returning the callback

**Returns:**

GLOBUS\_SUCCESS or an error object identifier

### 3.2.3.15 **globus\_result\_t globus\_gsi\_proxy\_handle\_get\_common\_name (globus\_gsi\_proxy\_handle\_t handle, char \*\* common\_name)**

Get the proxy common name stored in the GSI Proxy handle. This function retrieves the proxy common name from the GSI Proxy handle. The common name only impacts draft compliant proxies.

#### **Parameters:**

*handle* The handle from which to get the proxy common name.

*common\_name* Contains the proxy common name upon successful return. If the handle does not contain a common name, this parameter will be NULL upon return.

#### **Returns:**

GLOBUS\_SUCCESS upon success GLOBUS\_GSI\_PROXY\_ERROR\_WITH\_HANDLE if handle is invalid

### 3.2.3.16 **globus\_result\_t globus\_gsi\_proxy\_handle\_set\_is\_limited (globus\_gsi\_proxy\_handle\_t handle, globus\_bool\_t is\_limited)**

Set the limited proxy flag on the proxy handle.

#### **Parameters:**

*handle* the proxy handle

*is\_limited* boolean value to set on the proxy handle

#### **Returns:**

GLOBUS\_SUCCESS unless an error occurred, in which case, a globus error object ID is returned

### 3.2.3.17 **globus\_result\_t globus\_gsi\_proxy\_handle\_destroy (globus\_gsi\_proxy\_handle\_t handle)**

Destroy a GSI Proxy handle.

#### **Parameters:**

*handle* The handle to be destroyed.

#### **Returns:**

GLOBUS\_SUCCESS unless an error occurred, in which case, a globus error object ID is returned

#### **See also:**

[globus\\_gsi\\_proxy\\_handle\\_init\(\)](#)

### 3.2.3.18 `globus_result_t globus_gsi_proxy_handle_set_req (globus_gsi_proxy_handle_t handle, X509_REQ * req)`

Set the certificate request in a GSI Proxy handle.

#### Parameters:

*handle* The handle for which to set the certificate request

*req* Request to be copied to handle.

#### Returns:

GLOBUS\_SUCCESS unless an error occurred, in which case, a globus error object ID is returned

#### See also:

[globus\\_gsi\\_proxy\\_handle\\_get\\_req\(\)](#)

### 3.2.3.19 `globus_result_t globus_gsi_proxy_handle_set_private_key (globus_gsi_proxy_handle_t handle, EVP_PKEY * proxy_key)`

Set the private key in a GSI Proxy handle.

#### Parameters:

*handle* The handle for which to set the private key

*proxy\_key* Parameter used to pass the key

#### Returns:

GLOBUS\_SUCCESS unless an error occurred, in which case, a globus error object ID is returned

#### See also:

[globus\\_gsi\\_proxy\\_handle\\_get\\_private\\_key\(\)](#)

### 3.2.3.20 `globus_result_t globus_gsi_proxy_handle_set_type (globus_gsi_proxy_handle_t handle, globus_gsi_cert_utils_cert_type_t type)`

Set the type of proxy that will be generated when using this handle. Note that this will have no effect when generating a proxy from a proxy. In that case the generated proxy will inherit the type of the parent.

#### Parameters:

*handle* The handle for which to set the type

*type* Parameter used to pass the type.

#### Returns:

GLOBUS\_SUCCESS unless an error occurred, in which case, a globus error object ID is returned

#### See also:

[globus\\_gsi\\_proxy\\_handle\\_set\\_type\(\)](#)

### 3.2.3.21 **globus\_result\_t globus\_gsi\_proxy\_handle\_get\_policy** (globus\_gsi\_proxy\_handle\_t *handle*, unsigned char \*\**policy\_data*, int \**policy\_length*, int \**policy\_NID*)

Get the policy from the GSI Proxy handle. This function gets the policy that is being used in the proxy cert info extension.

#### Parameters:

*handle* The handle to be interrogated.  
*policy\_data* The policy data.  
*policy\_length* The length of the returned policy  
*policy\_NID* The NID of the policy language.

#### Returns:

GLOBUS\_SUCCESS if the handle is valid, otherwise an error is returned

#### See also:

[globus\\_gsi\\_proxy\\_handle\\_set\\_policy\(\)](#)

### 3.2.3.22 **globus\_result\_t globus\_gsi\_proxy\_handle\_get\_pathlen** (globus\_gsi\_proxy\_handle\_t *handle*, int \**pathlen*)

Get the path length from the GSI Proxy handle. This function gets the path length that is being used in the proxy cert info extension.

#### Parameters:

*handle* The handle to be interrogated.  
*pathlen* The maximum allowable path length

#### Returns:

GLOBUS\_SUCCESS if the handle is valid, otherwise an error is returned

#### See also:

[globus\\_gsi\\_proxy\\_handle\\_set\\_pathlen\(\)](#)

### 3.2.3.23 **globus\_result\_t globus\_gsi\_proxy\_handle\_set\_time\_valid** (globus\_gsi\_proxy\_handle\_t *handle*, int *time\_valid*)

Set the validity time of the proxy.

#### Parameters:

*handle* The proxy handle to set the expiration date for  
*time\_valid* desired expiration date of the proxy

#### Returns:

GLOBUS\_SUCCESS unless an error occurred, in which case, a globus error object ID is returned GLOBUS\_SUCCESS

### 3.2.3.24 **globus\_result\_t globus\_gsi\_proxy\_handle\_set\_proxy\_cert\_info (globus\_gsi\_proxy\_handle\_t handle, PROXYCERTINFO \* pci)**

Set the proxy cert info extension stored in the GSI Proxy handle. This function sets the proxy cert info extension in the GSI Proxy handle.

#### **Parameters:**

*handle* The handle for which to set the proxy cert info extension.

*pci* The proxy cert info extension to set.

#### **Returns:**

GLOBUS\_SUCCESS upon success GLOBUS\_GSI\_PROXY\_ERROR\_WITH\_HANDLE if handle is invalid  
GLOBUS\_GSI\_PROXY\_ERROR\_WITH\_PROXYCERTINFO if the pci pointer is invalid or if the set failed.

### 3.2.3.25 **globus\_result\_t globus\_gsi\_proxy\_handle\_set\_common\_name (globus\_gsi\_proxy\_handle\_t handle, char \* common\_name)**

Set the proxy common name stored in the GSI Proxy handle. This function sets the proxy common name in the GSI Proxy handle. Note that the common name is only used for draft compliant proxies.

#### **Parameters:**

*handle* The handle for which to set the proxy common name.

*common\_name* The proxy common name to set.

#### **Returns:**

GLOBUS\_SUCCESS upon success GLOBUS\_GSI\_PROXY\_ERROR\_WITH\_HANDLE if handle is invalid

### 3.2.3.26 **globus\_result\_t globus\_gsi\_proxy\_is\_limited (globus\_gsi\_proxy\_handle\_t handle, globus\_bool\_t \* is\_limited)**

Check to see if the proxy is a limited proxy.

#### **Parameters:**

*handle* the proxy handle to check

*is\_limited* boolean value to set depending on the type of proxy

#### **Returns:**

GLOBUS\_SUCCESS unless an error occurred, in which case, a globus error object ID is returned

## **3.3 Handle Attributes**

Handle attributes are used to control additional features of the GSI Proxy handle.

## Typedefs

- typedef struct globus\_l\_gsi\_proxy\_handle\_attrs\_s \* globus\_gsi\_proxy\_handle\_attrs\_t

## Initialize & Destroy

- globus\_result\_t globus\_gsi\_proxy\_handle\_attrs\_init (globus\_gsi\_proxy\_handle\_attrs\_t \*handle\_attrs)
- globus\_result\_t globus\_gsi\_proxy\_handle\_attrs\_destroy (globus\_gsi\_proxy\_handle\_attrs\_t handle\_attrs)

## Get/Set Key Bits

- globus\_result\_t globus\_gsi\_proxy\_handle\_attrs\_set\_keybits (globus\_gsi\_proxy\_handle\_attrs\_t handle\_attrs, int bits)
- globus\_result\_t globus\_gsi\_proxy\_handle\_attrs\_get\_keybits (globus\_gsi\_proxy\_handle\_attrs\_t handle\_attrs, int \*bits)

## Get/Set Initial Prime Number

- globus\_result\_t globus\_gsi\_proxy\_handle\_attrs\_set\_init\_prime (globus\_gsi\_proxy\_handle\_attrs\_t handle\_attrs, int prime)
- globus\_result\_t globus\_gsi\_proxy\_handle\_attrs\_get\_init\_prime (globus\_gsi\_proxy\_handle\_attrs\_t handle\_attrs, int \*prime)

## Get/Set Signing Algorithm

- globus\_result\_t globus\_gsi\_proxy\_handle\_attrs\_set\_signing\_algorithm (globus\_gsi\_proxy\_handle\_attrs\_t handle\_attrs, EVP\_MD \*algorithm)
- globus\_result\_t globus\_gsi\_proxy\_handle\_attrs\_get\_signing\_algorithm (globus\_gsi\_proxy\_handle\_attrs\_t handle\_attrs, EVP\_MD \*\*algorithm)

## Get/Set Clock Skew Allowable

- globus\_result\_t globus\_gsi\_proxy\_handle\_attrs\_set\_clock\_skew\_allowable (globus\_gsi\_proxy\_handle\_attrs\_t handle\_attrs, int skew)
- globus\_result\_t globus\_gsi\_proxy\_handle\_attrs\_get\_clock\_skew\_allowable (globus\_gsi\_proxy\_handle\_attrs\_t handle\_attrs, int \*skew)

## Get/Set Key Gen Callback

- globus\_result\_t globus\_gsi\_proxy\_handle\_attrs\_get\_key\_gen\_callback (globus\_gsi\_proxy\_handle\_attrs\_t handle\_attrs, void(\*\*callback)(int, int, void \*))
- globus\_result\_t globus\_gsi\_proxy\_handle\_attrs\_set\_key\_gen\_callback (globus\_gsi\_proxy\_handle\_attrs\_t handle\_attrs, void(\*callback)(int, int, void \*))

## Copy Attributes

- globus\_result\_t globus\_gsi\_proxy\_handle\_attrs\_copy (globus\_gsi\_proxy\_handle\_attrs\_t a, globus\_gsi\_proxy\_handle\_attrs\_t \*b)

### 3.3.1 Detailed Description

Handle attributes are used to control additional features of the GSI Proxy handle. These features are operation independent.

Currently there are no attributes.

See also:

[globus\\_gsi\\_proxy\\_handle\\_t](#)

### 3.3.2 Typedef Documentation

#### 3.3.2.1 typedef struct globus\_l\_gsi\_proxy\_handle\_attrs\_s\* globus\_gsi\_proxy\_handle\_attrs\_t

Handle Attributes. A GSI Proxy handle attributes type is used to associate immutable parameter values with a [Handle Management](#) handle. A handle attributes object should be created with immutable parameters and then passed to the proxy handle init function [globus\\_gsi\\_proxy\\_handle\\_init\(\)](#).

See also:

[Handle Management](#)

### 3.3.3 Function Documentation

#### 3.3.3.1 globus\_result\_t globus\_gsi\_proxy\_handle\_attrs\_init (globus\_gsi\_proxy\_handle\_attrs\_t \* handle\_attrs)

Initialize GSI Proxy Handle Attributes. Initialize proxy handle attributes, which can (and should) be associated with a proxy handle. For most purposes, these attributes should primarily be used by the proxy handle.

Currently, no attribute values are initialized.

**Parameters:**

*handle\_attrs* The handle attributes structure to be initialized

**Returns:**

GLOBUS\_SUCCESS unless an error occurred, in which case, a globus error object ID is returned

See also:

[globus\\_gsi\\_proxy\\_handle\\_attrs\\_destroy\(\)](#)

#### 3.3.3.2 globus\_result\_t globus\_gsi\_proxy\_handle\_attrs\_set\_keybits (globus\_gsi\_proxy\_handle\_attrs\_t handle\_attrs, int bits)

Set the length of the public key pair used by the proxy certificate.

**Parameters:**

*handle\_attrs* the attributes to set



*bits* the length to set it to (usually 1024)

**Returns:**

GLOBUS\_SUCCESS

**3.3.3.3 globus\_result\_t globus\_gsi\_proxy\_handle\_attrs\_set\_init\_prime (globus\_gsi\_proxy\_handle\_attrs\_t handle\_attrs, int prime)**

Set the initial prime number used for generating public key pairs in the RSA algorithm.

**Parameters:**

*handle\_attrs* The attributes to set

*prime* The prime number to set it to This value needs to be a prime number

**Returns:**

GLOBUS\_SUCCESS

**3.3.3.4 globus\_result\_t globus\_gsi\_proxy\_handle\_attrs\_set\_signing\_algorithm (globus\_gsi\_proxy\_handle\_attrs\_t handle\_attrs, EVP\_MD \* algorithm)**

Sets the Signing Algorithm to be used to sign the certificate request. In most cases, the signing party will ignore this value, and sign with an algorithm of its choice.

**Parameters:**

*handle\_attrs* The proxy handle to set the signing algorithm of

*algorithm* The signing algorithm to set

**Returns:**

Returns GLOBUS\_SUCCESS if the handle is valid, otherwise an error object is returned.

**3.3.3.5 globus\_result\_t globus\_gsi\_proxy\_handle\_attrs\_set\_clock\_skew\_allowable (globus\_gsi\_proxy\_handle\_attrs\_t handle\_attrs, int skew)**

Sets the clock skew in minutes of the proxy cert request so that time differences between hosts won't cause problems. This value defaults to 5 minutes.

**Parameters:**

*handle\_attrs* the handle\_attrs containing the clock skew to be set

*skew* the amount to skew by (in seconds)

**Returns:**

GLOBUS\_SUCCESS if the handle\_attrs is valid - otherwise an error is returned.

### 3.3.3.6 `globus_result_t globus_gsi_proxy_handle_attrs_get_key_gen_callback` (`globus_gsi_proxy_handle_attrs_t handle_attrs`, `void(**)(int, int, void *) callback`)

Get the public/private key generation callback that provides status during the generation of the keys.

#### Parameters:

*handle\_attrs* The handle\_attrs to get the callback from

*callback* The callback from the handle attributes

#### Returns:

GLOBUS\_SUCCESS if the handle\_attrs is valid, otherwise an error is returned

### 3.3.3.7 `globus_result_t globus_gsi_proxy_handle_attrs_copy` (`globus_gsi_proxy_handle_attrs_t a`, `globus_gsi_proxy_handle_attrs_t * b`)

Make a copy of GSI Proxy handle attributes.

#### Parameters:

*a* The handle attributes to copy

*b* The copy

#### Returns:

GLOBUS\_SUCCESS

### 3.3.3.8 `globus_result_t globus_gsi_proxy_handle_attrs_destroy` (`globus_gsi_proxy_handle_attrs_t` *handle\_attrs*)

Destroy the GSI Proxy handle attributes.

#### Parameters:

*handle\_attrs* The handle attributes to be destroyed.

#### Returns:

GLOBUS\_SUCCESS

#### See also:

[globus\\_gsi\\_proxy\\_handle\\_attrs\\_init\(\)](#)

### 3.3.3.9 `globus_result_t globus_gsi_proxy_handle_attrs_get_keybits` (`globus_gsi_proxy_handle_attrs_t` *handle\_attrs*, `int * bits`)

Gets the length of the public key pair used by the proxy certificate.

**Parameters:**

*handle\_attrs* the attributes to get the key length from  
*bits* the length of the key pair in bits

**Returns:**

GLOBUS\_SUCCESS

**3.3.3.10 globus\_result\_t globus\_gsi\_proxy\_handle\_attrs\_get\_init\_prime (globus\_gsi\_proxy\_handle\_attrs\_t handle\_attrs, int \* prime)**

Get the initial prime number used for generating the public key pair in the RSA algorithm.

**Parameters:**

*handle\_attrs* The attributes to get the initial prime number from  
*prime* The initial prime number taken from the attributes

**Returns:**

GLOBUS\_SUCCESS

**3.3.3.11 globus\_result\_t globus\_gsi\_proxy\_handle\_attrs\_get\_signing\_algorithm (globus\_gsi\_proxy\_handle\_attrs\_t handle\_attrs, EVP\_MD \*\* algorithm)**

Gets the Signing Algorithm to used to sign the certificate request. In most cases, the signing party will ignore this value, and sign with an algorithm of its choice.

**Parameters:**

*handle\_attrs* The proxy handle\_attrs to get the signing algorithm of  
*algorithm* Parameter used to return the signing algorithm used

**Returns:**

Returns GLOBUS\_SUCCESS if the handle is valid, otherwise an error object is returned.

**3.3.3.12 globus\_result\_t globus\_gsi\_proxy\_handle\_attrs\_get\_clock\_skew\_allowable (globus\_gsi\_proxy\_handle\_attrs\_t handle\_attrs, int \* skew)**

Get the allowable clock skew for the proxy certificate.

**Parameters:**

*handle\_attrs* The handle\_attrs to get the clock skew from  
*skew* The allowable clock skew (in seconds) to get from the proxy certificate request. This value gets set by the function, so it needs to be a pointer.

**Returns:**

GLOBUS\_SUCCESS if the handle\_attrs is valid, otherwise an error is returned

### 3.3.3.13 `globus_result_t globus_gsi_proxy_handle_attrs_set_key_gen_callback` (`globus_gsi_proxy_handle_attrs_t handle_attrs`, `void(*) (int, int, void *) callback`)

Set the public/private key generation callback that provides status during the generation of the keys.

#### Parameters:

*handle\_attrs* The handle\_attrs to get the callback from  
*callback* The callback from the handle attributes

#### Returns:

GLOBUS\_SUCCESS if the handle\_attrs is valid, otherwise an error is returned

## 3.4 Proxy Operations

Initiate a proxy operation.

### Create Request

- `globus_result_t globus_gsi_proxy_create_req` (`globus_gsi_proxy_handle_t handle`, `BIO *output_bio`)

### Inquire Request

- `globus_result_t globus_gsi_proxy_inquire_req` (`globus_gsi_proxy_handle_t handle`, `BIO *input_bio`)

### Resign Certificate

- `globus_result_t globus_gsi_proxy_resign_cert` (`globus_gsi_proxy_handle_t handle`, `globus_gsi_cred_handle_t issuer_credential`, `globus_gsi_cred_handle_t peer_credential`, `globus_gsi_cred_handle_t *resigned_credential`)

### Sign Request

- `globus_result_t globus_gsi_proxy_sign_req` (`globus_gsi_proxy_handle_t handle`, `globus_gsi_cred_handle_t issuer_credential`, `BIO *output_bio`)

### Create Signed

- `globus_result_t globus_gsi_proxy_create_signed` (`globus_gsi_proxy_handle_t handle`, `globus_gsi_cred_handle_t issuer`, `globus_gsi_cred_handle_t *proxy_credential`)

### Assemble credential

- `globus_result_t globus_gsi_proxy_assemble_cred` (`globus_gsi_proxy_handle_t handle`, `globus_gsi_cred_handle_t *proxy_credential`, `BIO *input_bio`)

### 3.4.1 Detailed Description

Initiate a proxy operation. This module contains the API functions for a user to request proxy request generation, proxy request inspection and proxy request signature.

### 3.4.2 Function Documentation

#### 3.4.2.1 `globus_result_t globus_gsi_proxy_create_req (globus_gsi_proxy_handle_t handle, BIO * output_bio)`

Create a proxy credential request. This function creates a proxy credential request, ie. a unsigned certificate and the corresponding private key, based on the handle that is passed in. The public part of the request is written to the BIO supplied in the output\_bio parameter. After the request is written, the PROXYCERTINFO extension contained in the handle is written to the BIO. The proxy handle is updated with the private key.

##### Parameters:

*handle* A GSI Proxy handle to use for the request operation.

*output\_bio* A BIO to write the resulting request structure to.

##### Returns:

GLOBUS\_SUCCESS unless an error occurred, in which case, a globus error object ID is returned

#### 3.4.2.2 `globus_result_t globus_gsi_proxy_inquire_req (globus_gsi_proxy_handle_t handle, BIO * input_bio)`

Inquire a proxy credential request. This function reads the public part of a proxy credential request from input\_bio and if the request contains a ProxyCertInfo extension, updates the handle with the information contained in the extension.

##### Parameters:

*handle* A GSI Proxy handle to use for the inquire operation.

*input\_bio* A BIO to read a request structure from.

##### Returns:

GLOBUS\_SUCCESS unless an error occurred, in which case, a globus error object ID is returned

#### 3.4.2.3 `globus_result_t globus_gsi_proxy_resign_cert (globus_gsi_proxy_handle_t handle, globus_gsi_cred_handle_t issuer_credential, globus_gsi_cred_handle_t peer_credential, globus_gsi_cred_handle_t * resigned_credential)`

Resign a existing certificate into a proxy. This function use the public key in a existing certificate to create a new proxy certificate chained to the issuers credentials. This operation will add a ProxyCertInfo extension to the proxy certificate if values contained in the extension are specified in the handle.

**Parameters:**

*handle* A GSI Proxy handle to use for the signing operation.

*issuer\_credential* The credential structure to be used for signing the proxy certificate.

*peer\_credential* The credential structure that contains the certificate to be resigned.

*resigned\_credential* A credential structure that upon return will contain the resigned certificate and associated certificate chain.

**Returns:**

GLOBUS\_SUCCESS unless an error occurred, in which case, a globus error object ID is returned

**3.4.2.4 globus\_result\_t globus\_gsi\_proxy\_sign\_req (globus\_gsi\_proxy\_handle\_t handle, globus\_gsi\_cred\_handle\_t issuer\_credential, BIO \* output\_bio)**

Sign a proxy certificate request. This function signs the public part of a proxy credential request, i.e. the unsigned certificate, previously read by inquire req using the supplied issuer\_credential. This operation will add a ProxyCertInfo extension to the proxy certificate if values contained in the extension are specified in the handle. The resulting signed certificate is written to the output\_bio.

**Parameters:**

*handle* A GSI Proxy handle to use for the signing operation.

*issuer\_credential* The credential structure to be used for signing the proxy certificate.

*output\_bio* A BIO to write the resulting certificate to.

**Returns:**

GLOBUS\_SUCCESS unless an error occurred, in which case, a globus error object ID is returned

**3.4.2.5 globus\_result\_t globus\_gsi\_proxy\_create\_signed (globus\_gsi\_proxy\_handle\_t handle, globus\_gsi\_cred\_handle\_t issuer, globus\_gsi\_cred\_handle\_t \* proxy\_credential)**

Create Signed Proxy Certificate.

**Parameters:**

*handle* The proxy handle used to create and sign the proxy certificate

*issuer* The issuing credential, used for signing the proxy certificate

*proxy\_credential* The new proxy credential, containing the signed cert, private key, etc.

**Returns:**

GLOBUS\_SUCCESS if no error occurred, an error object ID otherwise

### 3.4.2.6 `globus_result_t globus_gsi_proxy_assemble_cred (globus_gsi_proxy_handle_t handle, globus_gsi_cred_handle_t * proxy_credential, BIO * input_bio)`

Assemble a proxy credential. This function assembles a proxy credential. It reads a signed proxy certificate and a associated certificate chain from the `input_bio` and combines them with a private key previously generated by a call to `globus_gsi_proxy_create_req`. The resulting credential is then returned through the `proxy_credential` parameter.

#### Parameters:

*handle* A GSI Proxy handle to use for the assemble operation.

*proxy\_credential* This parameter will contain the assembled credential upon successful return.

*input\_bio* A BIO to read a signed certificate and corresponding certificate chain from.

#### Returns:

GLOBUS\_SUCCESS if no error occurred, an error object ID otherwise

## 3.5 Proxy Constants

#### Enumerations

- enum `globus_gsi_proxy_error_t` {  
    GLOBUS\_GSI\_PROXY\_ERROR\_SUCCESS = 0,  
    GLOBUS\_GSI\_PROXY\_ERROR\_WITH\_HANDLE = 1,  
    GLOBUS\_GSI\_PROXY\_ERROR\_WITH\_HANDLE\_ATTRS = 2,  
    GLOBUS\_GSI\_PROXY\_ERROR\_WITH\_PROXYPOLICY = 3,  
    GLOBUS\_GSI\_PROXY\_ERROR\_WITH\_PROXYCERTINFO = 4,  
    GLOBUS\_GSI\_PROXY\_ERROR\_WITH\_PATHLENGTH = 5,  
    GLOBUS\_GSI\_PROXY\_ERROR\_WITH\_X509\_REQ = 6,  
    GLOBUS\_GSI\_PROXY\_ERROR\_WITH\_X509 = 7,  
    GLOBUS\_GSI\_PROXY\_ERROR\_WITH\_X509\_EXTENSIONS = 8,  
    GLOBUS\_GSI\_PROXY\_ERROR\_WITH\_PRIVATE\_KEY = 9,  
    GLOBUS\_GSI\_PROXY\_ERROR\_WITH\_BIO = 10,  
    GLOBUS\_GSI\_PROXY\_ERROR\_WITH\_CREDENTIAL = 11,  
    GLOBUS\_GSI\_PROXY\_ERROR\_WITH\_CRED\_HANDLE = 12,  
    GLOBUS\_GSI\_PROXY\_ERROR\_WITH\_CRED\_HANDLE\_ATTRS = 13,  
    GLOBUS\_GSI\_PROXY\_ERROR\_ERRNO = 14,  
    GLOBUS\_GSI\_PROXY\_ERROR\_SETTING\_HANDLE\_TYPE = 15,  
    GLOBUS\_GSI\_PROXY\_INVALID\_PARAMETER = 16,  
    GLOBUS\_GSI\_PROXY\_ERROR\_SIGNING = 17,  
    GLOBUS\_GSI\_PROXY\_ERROR\_LAST = 18 }

### 3.5.1 Enumeration Type Documentation

#### 3.5.1.1 enum globus\_gsi\_proxy\_error\_t

Proxy Error codes.

##### Enumerator:

***GLOBUS\_GSI\_PROXY\_ERROR\_SUCCESS*** Success - never used.

***GLOBUS\_GSI\_PROXY\_ERROR\_WITH\_HANDLE*** Invalid proxy handle state.

***GLOBUS\_GSI\_PROXY\_ERROR\_WITH\_HANDLE\_ATTRS*** Invalid proxy handle attributes state.

***GLOBUS\_GSI\_PROXY\_ERROR\_WITH\_PROXYCERTINFO*** Error with ASN.1 proxycertinfo structure.

***GLOBUS\_GSI\_PROXY\_ERROR\_WITH\_PROXYPOLICY*** Error with ASN.1 proxypolicy structure.

***GLOBUS\_GSI\_PROXY\_ERROR\_WITH\_PATHLENGTH*** Error with proxy path length.

***GLOBUS\_GSI\_PROXY\_ERROR\_WITH\_X509\_REQ*** Error with the X.509 request structure.

***GLOBUS\_GSI\_PROXY\_ERROR\_WITH\_X509*** Error with X.509 structure.

***GLOBUS\_GSI\_PROXY\_ERROR\_WITH\_X509\_EXTENSIONS*** Error with X.509 extensions.

***GLOBUS\_GSI\_PROXY\_ERROR\_WITH\_PRIVATE\_KEY*** Error with private key.

***GLOBUS\_GSI\_PROXY\_ERROR\_WITH\_BIO*** Error with OpenSSL's BIO handle.

***GLOBUS\_GSI\_PROXY\_ERROR\_WITH\_CREDENTIAL*** Error with credential.

***GLOBUS\_GSI\_PROXY\_ERROR\_WITH\_CRED\_HANDLE*** Error with credential handle.

***GLOBUS\_GSI\_PROXY\_ERROR\_WITH\_CRED\_HANDLE\_ATTRS*** Error with credential handle attributes.

***GLOBUS\_GSI\_PROXY\_ERROR\_ERRNO*** System error.

***GLOBUS\_GSI\_PROXY\_ERROR\_SETTING\_HANDLE\_TYPE*** Unable to set proxy type.

***GLOBUS\_GSI\_PROXY\_INVALID\_PARAMETER*** Invalid function parameter.

***GLOBUS\_GSI\_PROXY\_ERROR\_SIGNING*** A error occurred while signing the proxy certificate.

***GLOBUS\_GSI\_PROXY\_ERROR\_LAST*** Last marker - never used.



## Index

Activation, [1](#)

globus\_gsi\_proxy\_constants

[GLOBUS\\_GSI\\_PROXY\\_ERROR\\_ERRNO](#), [22](#)  
    [GLOBUS\\_GSI\\_PROXY\\_ERROR\\_LAST](#), [22](#)  
    [GLOBUS\\_GSI\\_PROXY\\_ERROR\\_SETTING\\_-  
        HANDLE\\_TYPE](#), [22](#)  
    [GLOBUS\\_GSI\\_PROXY\\_ERROR\\_SIGNING](#), [22](#)  
    [GLOBUS\\_GSI\\_PROXY\\_ERROR\\_SUCCESS](#), [22](#)  
    [GLOBUS\\_GSI\\_PROXY\\_ERROR\\_WITH\\_BIO](#),  
        [22](#)  
    [GLOBUS\\_GSI\\_PROXY\\_ERROR\\_WITH\\_-  
        CRED\\_HANDLE](#), [22](#)  
    [GLOBUS\\_GSI\\_PROXY\\_ERROR\\_WITH\\_-  
        CRED\\_HANDLE\\_ATTRS](#), [22](#)  
    [GLOBUS\\_GSI\\_PROXY\\_ERROR\\_WITH\\_-  
        CREDENTIAL](#), [22](#)  
    [GLOBUS\\_GSI\\_PROXY\\_ERROR\\_WITH\\_-  
        HANDLE](#), [22](#)  
    [GLOBUS\\_GSI\\_PROXY\\_ERROR\\_WITH\\_-  
        HANDLE\\_ATTRS](#), [22](#)  
    [GLOBUS\\_GSI\\_PROXY\\_ERROR\\_WITH\\_-  
        PATHLENGTH](#), [22](#)  
    [GLOBUS\\_GSI\\_PROXY\\_ERROR\\_WITH\\_-  
        PRIVATE\\_KEY](#), [22](#)  
    [GLOBUS\\_GSI\\_PROXY\\_ERROR\\_WITH\\_-  
        PROXYCERTINFO](#), [22](#)  
    [GLOBUS\\_GSI\\_PROXY\\_ERROR\\_WITH\\_-  
        PROXYPOLICY](#), [22](#)  
    [GLOBUS\\_GSI\\_PROXY\\_ERROR\\_WITH\\_X509](#),  
        [22](#)  
    [GLOBUS\\_GSI\\_PROXY\\_ERROR\\_WITH\\_-  
        X509\\_EXTENSIONS](#), [22](#)  
    [GLOBUS\\_GSI\\_PROXY\\_ERROR\\_WITH\\_-  
        X509\\_REQ](#), [22](#)  
    [GLOBUS\\_GSI\\_PROXY\\_INVALID\\_-  
        PARAMETER](#), [22](#)  
[GLOBUS\\_GSI\\_PROXY\\_ERROR\\_ERRNO](#)  
    [globus\\_gsi\\_proxy\\_constants](#), [22](#)  
[GLOBUS\\_GSI\\_PROXY\\_ERROR\\_LAST](#)  
    [globus\\_gsi\\_proxy\\_constants](#), [22](#)  
[GLOBUS\\_GSI\\_PROXY\\_ERROR\\_SETTING\\_-  
    HANDLE\\_TYPE](#)  
    [globus\\_gsi\\_proxy\\_constants](#), [22](#)  
[GLOBUS\\_GSI\\_PROXY\\_ERROR\\_SIGNING](#)  
    [globus\\_gsi\\_proxy\\_constants](#), [22](#)  
[GLOBUS\\_GSI\\_PROXY\\_ERROR\\_SUCCESS](#)  
    [globus\\_gsi\\_proxy\\_constants](#), [22](#)  
[GLOBUS\\_GSI\\_PROXY\\_ERROR\\_WITH\\_BIO](#)  
    [globus\\_gsi\\_proxy\\_constants](#), [22](#)  
[GLOBUS\\_GSI\\_PROXY\\_ERROR\\_WITH\\_CRED\\_-  
    HANDLE](#)  
    [globus\\_gsi\\_proxy\\_constants](#), [22](#)

[GLOBUS\\_GSI\\_PROXY\\_ERROR\\_WITH\\_CRED\\_-  
    HANDLE\\_ATTRS](#)  
    [globus\\_gsi\\_proxy\\_constants](#), [22](#)  
[GLOBUS\\_GSI\\_PROXY\\_ERROR\\_WITH\\_-  
    CREDENTIAL](#)  
    [globus\\_gsi\\_proxy\\_constants](#), [22](#)  
[GLOBUS\\_GSI\\_PROXY\\_ERROR\\_WITH\\_HANDLE](#)  
    [globus\\_gsi\\_proxy\\_constants](#), [22](#)  
[GLOBUS\\_GSI\\_PROXY\\_ERROR\\_WITH\\_-  
    HANDLE\\_ATTRS](#)  
    [globus\\_gsi\\_proxy\\_constants](#), [22](#)  
[GLOBUS\\_GSI\\_PROXY\\_ERROR\\_WITH\\_-  
    PATHLENGTH](#)  
    [globus\\_gsi\\_proxy\\_constants](#), [22](#)  
[GLOBUS\\_GSI\\_PROXY\\_ERROR\\_WITH\\_PRIVATE\\_-  
    KEY](#)  
    [globus\\_gsi\\_proxy\\_constants](#), [22](#)  
[GLOBUS\\_GSI\\_PROXY\\_ERROR\\_WITH\\_-  
    PROXYCERTINFO](#)  
    [globus\\_gsi\\_proxy\\_constants](#), [22](#)  
[GLOBUS\\_GSI\\_PROXY\\_ERROR\\_WITH\\_-  
    PROXYPOLICY](#)  
    [globus\\_gsi\\_proxy\\_constants](#), [22](#)  
[GLOBUS\\_GSI\\_PROXY\\_ERROR\\_WITH\\_X509](#)  
    [globus\\_gsi\\_proxy\\_constants](#), [22](#)  
[GLOBUS\\_GSI\\_PROXY\\_ERROR\\_WITH\\_X509\\_-  
    EXTENSIONS](#)  
    [globus\\_gsi\\_proxy\\_constants](#), [22](#)  
[GLOBUS\\_GSI\\_PROXY\\_ERROR\\_WITH\\_X509\\_REQ](#)  
    [globus\\_gsi\\_proxy\\_constants](#), [22](#)  
[GLOBUS\\_GSI\\_PROXY\\_INVALID\\_PARAMETER](#)  
    [globus\\_gsi\\_proxy\\_constants](#), [22](#)  
[globus\\_gsi\\_proxy\\_activation](#)  
    [GLOBUS\\_GSI\\_PROXY\\_MODULE](#), [2](#)  
[globus\\_gsi\\_proxy\\_assemble\\_cred](#)  
    [globus\\_gsi\\_proxy\\_operations](#), [20](#)  
[globus\\_gsi\\_proxy\\_constants](#)  
    [globus\\_gsi\\_proxy\\_error\\_t](#), [22](#)  
[globus\\_gsi\\_proxy\\_create\\_req](#)  
    [globus\\_gsi\\_proxy\\_operations](#), [19](#)  
[globus\\_gsi\\_proxy\\_create\\_signed](#)  
    [globus\\_gsi\\_proxy\\_operations](#), [20](#)  
[globus\\_gsi\\_proxy\\_error\\_t](#)  
    [globus\\_gsi\\_proxy\\_constants](#), [22](#)  
[globus\\_gsi\\_proxy\\_handle](#)  
    [globus\\_gsi\\_proxy\\_handle\\_clear\\_cert\\_info](#), [7](#)  
    [globus\\_gsi\\_proxy\\_handle\\_destroy](#), [9](#)  
    [globus\\_gsi\\_proxy\\_handle\\_get\\_clock\\_skew\\_-  
        allowable](#), [8](#)  
    [globus\\_gsi\\_proxy\\_handle\\_get\\_common\\_name](#), [9](#)  
    [globus\\_gsi\\_proxy\\_handle\\_get\\_init\\_prime](#), [8](#)  
    [globus\\_gsi\\_proxy\\_handle\\_get\\_key\\_gen\\_callback](#),  
        [9](#)

- globus\_gsi\_proxy\_handle\_get\_keybits, 8
- globus\_gsi\_proxy\_handle\_get\_pathlen, 11
- globus\_gsi\_proxy\_handle\_get\_policy, 11
- globus\_gsi\_proxy\_handle\_get\_private\_key, 5
- globus\_gsi\_proxy\_handle\_get\_proxy\_cert\_info, 7
- globus\_gsi\_proxy\_handle\_get\_req, 5
- globus\_gsi\_proxy\_handle\_get\_signing\_algorithm, 7
- globus\_gsi\_proxy\_handle\_get\_time\_valid, 6
- globus\_gsi\_proxy\_handle\_get\_type, 5
- globus\_gsi\_proxy\_handle\_init, 4
- globus\_gsi\_proxy\_handle\_set\_common\_name, 12
- globus\_gsi\_proxy\_handle\_set\_is\_limited, 9
- globus\_gsi\_proxy\_handle\_set\_pathlen, 6
- globus\_gsi\_proxy\_handle\_set\_policy, 6
- globus\_gsi\_proxy\_handle\_set\_private\_key, 10
- globus\_gsi\_proxy\_handle\_set\_proxy\_cert\_info, 12
- globus\_gsi\_proxy\_handle\_set\_req, 10
- globus\_gsi\_proxy\_handle\_set\_time\_valid, 12
- globus\_gsi\_proxy\_handle\_set\_type, 10
- globus\_gsi\_proxy\_handle\_t, 4
- globus\_gsi\_proxy\_is\_limited, 12
- globus\_gsi\_proxy\_handle\_attrs
  - globus\_gsi\_proxy\_handle\_attrs\_copy, 16
  - globus\_gsi\_proxy\_handle\_attrs\_destroy, 16
  - globus\_gsi\_proxy\_handle\_attrs\_get\_clock\_skew\_allowable, 17
  - globus\_gsi\_proxy\_handle\_attrs\_get\_init\_prime, 17
  - globus\_gsi\_proxy\_handle\_attrs\_get\_key\_gen\_callback, 16
  - globus\_gsi\_proxy\_handle\_attrs\_get\_keybits, 17
  - globus\_gsi\_proxy\_handle\_attrs\_get\_signing\_algorithm, 17
  - globus\_gsi\_proxy\_handle\_attrs\_init, 14
  - globus\_gsi\_proxy\_handle\_attrs\_set\_clock\_skew\_allowable, 15
  - globus\_gsi\_proxy\_handle\_attrs\_set\_init\_prime, 15
  - globus\_gsi\_proxy\_handle\_attrs\_set\_key\_gen\_callback, 18
  - globus\_gsi\_proxy\_handle\_attrs\_set\_keybits, 15
  - globus\_gsi\_proxy\_handle\_attrs\_set\_signing\_algorithm, 15
  - globus\_gsi\_proxy\_handle\_attrs\_t, 14
- globus\_gsi\_proxy\_handle\_attrs\_copy
  - globus\_gsi\_proxy\_handle\_attrs, 16
- globus\_gsi\_proxy\_handle\_attrs\_destroy
  - globus\_gsi\_proxy\_handle\_attrs, 16
- globus\_gsi\_proxy\_handle\_attrs\_get\_clock\_skew\_allowable
  - globus\_gsi\_proxy\_handle\_attrs, 17
- globus\_gsi\_proxy\_handle\_attrs\_get\_init\_prime
  - globus\_gsi\_proxy\_handle\_attrs, 17
- globus\_gsi\_proxy\_handle\_attrs\_get\_key\_gen\_callback
  - globus\_gsi\_proxy\_handle\_attrs, 16
- globus\_gsi\_proxy\_handle\_attrs\_get\_keybits
  - globus\_gsi\_proxy\_handle\_attrs, 17
- globus\_gsi\_proxy\_handle\_attrs\_get\_signing\_algorithm
  - globus\_gsi\_proxy\_handle\_attrs, 17
- globus\_gsi\_proxy\_handle\_attrs\_init
  - globus\_gsi\_proxy\_handle\_attrs, 14
- globus\_gsi\_proxy\_handle\_attrs\_set\_clock\_skew\_allowable
  - globus\_gsi\_proxy\_handle\_attrs, 15
- globus\_gsi\_proxy\_handle\_attrs\_set\_init\_prime
  - globus\_gsi\_proxy\_handle\_attrs, 15
- globus\_gsi\_proxy\_handle\_attrs\_set\_key\_gen\_callback
  - globus\_gsi\_proxy\_handle\_attrs, 18
- globus\_gsi\_proxy\_handle\_attrs\_set\_keybits
  - globus\_gsi\_proxy\_handle\_attrs, 15
- globus\_gsi\_proxy\_handle\_attrs\_set\_signing\_algorithm
  - globus\_gsi\_proxy\_handle\_attrs, 15
- globus\_gsi\_proxy\_handle\_attrs\_t
  - globus\_gsi\_proxy\_handle\_attrs, 14
- globus\_gsi\_proxy\_handle\_clear\_cert\_info
  - globus\_gsi\_proxy\_handle, 7
- globus\_gsi\_proxy\_handle\_destroy
  - globus\_gsi\_proxy\_handle, 9
- globus\_gsi\_proxy\_handle\_get\_clock\_skew\_allowable
  - globus\_gsi\_proxy\_handle, 8
- globus\_gsi\_proxy\_handle\_get\_common\_name
  - globus\_gsi\_proxy\_handle, 9
- globus\_gsi\_proxy\_handle\_get\_init\_prime
  - globus\_gsi\_proxy\_handle, 8
- globus\_gsi\_proxy\_handle\_get\_key\_gen\_callback
  - globus\_gsi\_proxy\_handle, 9
- globus\_gsi\_proxy\_handle\_get\_keybits
  - globus\_gsi\_proxy\_handle, 8
- globus\_gsi\_proxy\_handle\_get\_pathlen
  - globus\_gsi\_proxy\_handle, 11
- globus\_gsi\_proxy\_handle\_get\_policy
  - globus\_gsi\_proxy\_handle, 11
- globus\_gsi\_proxy\_handle\_get\_private\_key
  - globus\_gsi\_proxy\_handle, 5
- globus\_gsi\_proxy\_handle\_get\_proxy\_cert\_info
  - globus\_gsi\_proxy\_handle, 7
- globus\_gsi\_proxy\_handle\_get\_req
  - globus\_gsi\_proxy\_handle, 5
- globus\_gsi\_proxy\_handle\_get\_signing\_algorithm
  - globus\_gsi\_proxy\_handle, 7
- globus\_gsi\_proxy\_handle\_get\_time\_valid
  - globus\_gsi\_proxy\_handle, 6
- globus\_gsi\_proxy\_handle\_get\_type
  - globus\_gsi\_proxy\_handle, 5
- globus\_gsi\_proxy\_handle\_init
  - globus\_gsi\_proxy\_handle, 4
- globus\_gsi\_proxy\_handle\_set\_common\_name
  - globus\_gsi\_proxy\_handle, 12
- globus\_gsi\_proxy\_handle\_set\_is\_limited
  - globus\_gsi\_proxy\_handle, 9

- globus\_gsi\_proxy\_handle\_set\_pathlen
  - globus\_gsi\_proxy\_handle, [6](#)
- globus\_gsi\_proxy\_handle\_set\_policy
  - globus\_gsi\_proxy\_handle, [6](#)
- globus\_gsi\_proxy\_handle\_set\_private\_key
  - globus\_gsi\_proxy\_handle, [10](#)
- globus\_gsi\_proxy\_handle\_set\_proxy\_cert\_info
  - globus\_gsi\_proxy\_handle, [12](#)
- globus\_gsi\_proxy\_handle\_set\_req
  - globus\_gsi\_proxy\_handle, [10](#)
- globus\_gsi\_proxy\_handle\_set\_time\_valid
  - globus\_gsi\_proxy\_handle, [12](#)
- globus\_gsi\_proxy\_handle\_set\_type
  - globus\_gsi\_proxy\_handle, [10](#)
- globus\_gsi\_proxy\_handle\_t
  - globus\_gsi\_proxy\_handle, [4](#)
- globus\_gsi\_proxy\_inquire\_req
  - globus\_gsi\_proxy\_operations, [19](#)
- globus\_gsi\_proxy\_is\_limited
  - globus\_gsi\_proxy\_handle, [12](#)
- GLOBUS\_GSI\_PROXY\_MODULE
  - globus\_gsi\_proxy\_activation, [2](#)
- globus\_gsi\_proxy\_operations
  - globus\_gsi\_proxy\_assemble\_cred, [20](#)
  - globus\_gsi\_proxy\_create\_req, [19](#)
  - globus\_gsi\_proxy\_create\_signed, [20](#)
  - globus\_gsi\_proxy\_inquire\_req, [19](#)
  - globus\_gsi\_proxy\_resign\_cert, [19](#)
  - globus\_gsi\_proxy\_sign\_req, [20](#)
- globus\_gsi\_proxy\_resign\_cert
  - globus\_gsi\_proxy\_operations, [19](#)
- globus\_gsi\_proxy\_sign\_req
  - globus\_gsi\_proxy\_operations, [20](#)

Handle Attributes, [13](#)

Handle Management, [2](#)

Proxy Constants, [21](#)

Proxy Operations, [18](#)