

# globus gass transfer Reference Manual

7.1

Generated by Doxygen 1.3.9.1

Thu Jan 26 15:11:49 2012

# Contents

<b>1</b>	<b>globus gass transfer Main Page</b>	<b>1</b>
<b>2</b>	<b>globus gass transfer Module Index</b>	<b>1</b>
<b>3</b>	<b>globus gass transfer Data Structure Index</b>	<b>2</b>
<b>4</b>	<b>globus gass transfer Module Documentation</b>	<b>2</b>
<b>5</b>	<b>globus gass transfer Data Structure Documentation</b>	<b>41</b>

## 1 globus gass transfer Main Page

The GASS Transfer API is the core part of the GASS (Global Access to Secondary Storage) component of the Globus Toolkit. The purpose of GASS is to provide a simple way to enable grid applications to securely stage and access data to and from remote file servers using a simple protocol-independent API.

The GASS Transfer API provides a way to implement both **client**(p. 2) and **server**(p. 7) components. These share common data block and request management functionality. Client-specific functions are provided to implement file "get", "put", and "append" operations. Server-specific functions are provided to implement servers which service such requests. Client and server functionality can be included in a single application, so one could implement proxies or cross-protocol bridges.

The GASS Transfer API is easily extensible to support different remote data access protocols. The standard Globus distribution includes client-side support for the http, and https protocols, as well as server-side support for the http and https protocols. An application which requires additional protocol support may add this through the **protocol module interface**(p. 33).

The GASS Transfer API is defined in the header file "globus\_gass\_transfer.h"

The **GLOBUS\_GASS\_TRANSFER\_MODULE**(p. 2) must be activated before calling any functions in this API.

## 2 globus gass transfer Module Index

### 2.1 globus gass transfer Modules

Here is a list of all modules:

<b>Activation</b>	<b>2</b>
<b>Client-Initiated Operations</b>	<b>2</b>
<b>Implementing Servers</b>	<b>7</b>
<b>Sending and Receiving Data</b>	<b>13</b>
<b>Referrals</b>	<b>15</b>
<b>Request Handles</b>	<b>16</b>
<b>Request Attributes</b>	<b>22</b>
<b>Listener attributes</b>	<b>29</b>

<b>Implementing Request Attributes</b>	<b>31</b>
<b>Protocol Modules</b>	<b>33</b>

## 3 globus gass transfer Data Structure Index

### 3.1 globus gass transfer Data Structures

Here are the data structures with brief descriptions:

<b>globus_gass_transfer_listener_proto_s (Protocol module listener handling structure )</b>	<b>41</b>
<b>globus_gass_transfer_proto_descriptor_t (Protocol module descriptor structure )</b>	<b>42</b>
<b>globus_gass_transfer_request_proto_s (Protocol module request handling structure )</b>	<b>44</b>
<b>globus_gass_transfer_request_t</b>	<b>45</b>

## 4 globus gass transfer Module Documentation

### 4.1 Activation

The Globus GASS Transfer library uses the standard module activation and deactivation API to initialize its state.

#### Defines

- **#define GLOBUS\_GASS\_TRANSFER\_MODULE**

#### 4.1.1 Detailed Description

The Globus GASS Transfer library uses the standard module activation and deactivation API to initialize its state.

Before any GASS functions are called, the module must be activated

```
globus_module_activate(GLOBUS_GASS_TRANSFER_MODULE);
```

This function returns GLOBUS\_SUCCESS if the GASS library was successfully initialized. This may be called multiple times.

To deactivate the GASS transfer library, the following must be called

```
globus_module_deactivate(GLOBUS_GASS_TRANSFER_MODULE);
```

#### 4.1.2 Define Documentation

##### 4.1.2.1 #define GLOBUS\_GASS\_TRANSFER\_MODULE

Module descriptor.

### 4.2 Client-Initiated Operations

GASS Transfer Client Operations.

## Functions

- `int globus_gass_transfer_register_get (globus_gass_transfer_request_t *request, globus_gass_transfer_requestattr_t *attr, char *url, globus_gass_transfer_callback_t callback, void *user_arg)`
- `int globus_gass_transfer_get (globus_gass_transfer_request_t *request, globus_gass_transfer_requestattr_t *attr, char *url)`
- `int globus_gass_transfer_register_put (globus_gass_transfer_request_t *request, globus_gass_transfer_requestattr_t *attr, char *url, globus_size_t length, globus_gass_transfer_callback_t callback, void *user_arg)`
- `int globus_gass_transfer_put (globus_gass_transfer_request_t *request, globus_gass_transfer_requestattr_t *attr, char *url, globus_size_t length)`
- `int globus_gass_transfer_register_append (globus_gass_transfer_request_t *request, globus_gass_transfer_requestattr_t *attr, char *url, globus_size_t length, globus_gass_transfer_callback_t callback, void *user_arg)`
- `int globus_gass_transfer_append (globus_gass_transfer_request_t *request, globus_gass_transfer_requestattr_t *attr, char *url, globus_size_t length)`

### 4.2.1 Detailed Description

GASS Transfer Client Operations.

One mode of using the GASS Transfer API is to initiate file transfers. The operations supported by the GASS Transfer API are file get, put, and append. These operations are provided for HTTP, and HTTPS file servers. The **protocol module interface**(p. 33) allows support for additional protocols to be added easily.

The GASS transfer library provides both blocking and non-blocking versions of all its client functions. When a blocking function completes, or the non-blocking function's callback is called, the user should check the request's status to discover whether the transfer was completed successfully, denied, or referred.

### 4.2.2 Function Documentation

#### 4.2.2.1 `int globus_gass_transfer_register_get (globus_gass_transfer_request_t * request, globus_gass_transfer_requestattr_t * attr, char * url, globus_gass_transfer_callback_t callback, void * user_arg)`

Nonblocking file get.

This function initiates a new "get" request of the file named by *url*. The entire file will be transferred from the server if the file exists and user is authorized to do so. This function does not block; instead, the user's callback function will be called once the GASS library has determined whether the file can be retrieved or not.

Upon returning from this function, the request handle is initialized to refer to the *get* request's state.

If the server can't store the file at *url*, but has an alternative location for the user to store to, then the callback function will be called with the request's status set to *GLOBUS\_GASS\_TRANSFER\_REQUEST\_REFERRED*.

#### Parameters:

*request* A pointer to an uninitialized request handle.

*attr* Request attributes.

*url* URL to get

*callback* Function to call once the URL has been accepted, referred, or denied by the server.

*user\_arg* User-supplied argument to the callback function.

#### Return values:

*GLOBUS\_SUCCESS* The get was successfully initiated.

*GLOBUS\_GASS\_TRANSFER\_ERROR\_NULL\_POINTER* One of request, attr, or callback was GLOBUS\_NULL. The get could not be processed.

***GLOBUS\_GASS\_TRANSFER\_ERROR\_INTERNAL\_ERROR*** An internal resource was not available to process the get.

***GLOBUS\_GASS\_TRANSFER\_ERROR\_NOT\_IMPLEMENTED*** No protocol handler for doing a get on this URL type is implemented.

***GLOBUS\_GASS\_TRANSFER\_ERROR\_BAD\_URL*** The URL could not be parsed.

See also:

`globus_gass_transfer_get()`(p. 4)

**4.2.2.2** `int globus_gass_transfer_get (globus_gass_transfer_request_t * request, globus_gass_transfer_requestattr_t * attr, char * url)`

Blocking file get.

This function initiates a new "get" request of the file named by *url*. The entire file will be transferred from the server if the file exists and user is authorized to do so. This function blocks until the GASS library has determined whether the file may be retrieved by the caller, may not because it is a referral to another URL or URLs, or the server has denied the request.

Upon returning from this function, the request handle is initialized to refer to the *get* request's state. This request handle must be destroyed after the user has finished processing the data associated with the callback.

If the file doesn't exist at *url*, but a referral does, then this function will return with the request's status set to *GLOBUS\_GASS\_TRANSFER\_REQUEST\_REFERRED*.

**Parameters:**

*request* A pointer to an uninitialized request handle.

*attr* Request attributes.

*url* URL to get

**Return values:**

***GLOBUS\_SUCCESS*** The get was successfully initiated.

***GLOBUS\_GASS\_TRANSFER\_ERROR\_NULL\_POINTER*** One of request or attr was *GLOBUS\_NULL*. The get could not be processed.

***GLOBUS\_GASS\_TRANSFER\_ERROR\_INTERNAL\_ERROR*** An internal resource was not available to process the get.

***GLOBUS\_GASS\_TRANSFER\_ERROR\_NOT\_IMPLEMENTED*** No protocol handler for doing a get on this URL type is implemented.

***GLOBUS\_GASS\_TRANSFER\_ERROR\_BAD\_URL*** The URL could not be parsed.

See also:

`globus_gass_transfer_register_get()`(p. 3)

**4.2.2.3** `int globus_gass_transfer_register_put (globus_gass_transfer_request_t * request, globus_gass_transfer_requestattr_t * attr, char * url, globus_size_t length, globus_gass_transfer_callback_t callback, void * user_arg)`

Nonblocking file put.

This function initiates a new "put" request of the file named by *url*. The entire file will be transferred to the server if the user is authorized to do so. This function does not block; instead, the user's callback function will be called once the GASS library has determined whether the file may be stored or not.

Upon returning from this function, the request handle is initialized to refer to the *put* request's state.

If the server can't store the file at *url*, but has an alternative location for the user to store to, then the callback function will be called with the request's status set to *GLOBUS\_GASS\_TRANSFER\_REQUEST\_REFERRED*.

**Parameters:**

**request** A pointer to an uninitialized request handle.

**attr** Request attributes.

**url** URL to put.

**length** The length of the data to put to the url, if known. If this parameter is set to *GLOBUS\_GASS\_LENGTH\_UNKNOWN*, then the put may fail if the protocol does not support arbitrarily-length files.

**callback** Function to call once the URL has been accepted, referred, or denied by the server.

**user\_arg** User-supplied argument to the callback function.

**Return values:**

**GLOBUS\_SUCCESS** The put was successfully initiated.

**GLOBUS\_GASS\_TRANSFER\_ERROR\_NULL\_POINTER** One of request, attr, or callback was GLOBUS\_NULL. The put could not be processed.

**GLOBUS\_GASS\_TRANSFER\_ERROR\_INTERNAL\_ERROR** An internal resource was not available to process the put.

**GLOBUS\_GASS\_TRANSFER\_ERROR\_NOT\_IMPLEMENTED** No protocol handler for doing a put on this URL type is implemented.

**GLOBUS\_GASS\_TRANSFER\_ERROR\_BAD\_URL** The URL could not be parsed.

**See also:**

`globus_gass_transfer_put()`(p. 5)

#### 4.2.2.4 `int globus_gass_transfer_put (globus_gass_transfer_request_t * request, globus_gass_transfer_requestattr_t * attr, char * url, globus_size_t length)`

Blocking file put.

This function initiates a new "put" request of the file named by *url*. The entire file will be transferred to the server if the user is authorized to do so. This function blocks until the GASS library has determined whether the file may be retrieved by the caller, may not because it is a referral to another URL or URLs, or the server has denied the request.

Upon returning from this function, the request handle is initialized to refer to the *put* request's state.

If the server can't store the file at *url*, but has an alternative location for the user to store to, then this function return with request's status set to *GLOBUS\_GASS\_TRANSFER\_REQUEST\_REFERRED*.

**Parameters:**

**request** A pointer to an uninitialized request handle.

**attr** Request attributes.

**url** URL to put.

**length** The length of the data to put to the url, if known. If this parameter is set to *GLOBUS\_GASS\_LENGTH\_UNKNOWN*, then the put may fail if the protocol does not support arbitrarily-length files.

**Return values:**

**GLOBUS\_SUCCESS** The get was successfully initiated.

**GLOBUS\_GASS\_TRANSFER\_ERROR\_NULL\_POINTER** One of request or attr was GLOBUS\_NULL. The get could not be processed.

**GLOBUS\_GASS\_TRANSFER\_ERROR\_INTERNAL\_ERROR** An internal resource was not available to process the get.

**GLOBUS\_GASS\_TRANSFER\_ERROR\_NOT\_IMPLEMENTED** No protocol handler for doing a put on this URL type is implemented.

***GLOBUS\_GASS\_TRANSFER\_ERROR\_BAD\_URL*** The URL could not be parsed.

See also:

***globus\_gass\_transfer\_register\_put()***(p. 4)

**4.2.2.5** ***int globus\_gass\_transfer\_register\_append (globus\_gass\_transfer\_request\_t \* request, globus\_gass\_transfer\_requestattr\_t \* attr, char \* url, globus\_size\_t length, globus\_gass\_transfer\_callback\_t callback, void \* user\_arg)***

Nonblocking file append.

This function initiates a new "append" request of the file named by *url*. The entire file will be transferred to the server if the user is authorized to do so. This function does not block; instead, the user's callback function will be called once the GASS library has determined whether the file may be stored or not.

Upon returning from this function, the request handle is initialized to refer to the *append* request's state.

If the server can't store the file at *url*, but has an alternative location for the user to store to, then the callback function will be called with the request's status set to ***GLOBUS\_GASS\_TRANSFER\_REQUEST\_REFERRED***.

**Parameters:**

***request*** A pointer to an uninitialized request handle.

***attr*** Request attributes.

***url*** URL to append to.

***length*** The length of the data to append to the url, if known. If this parameter is set to ***GLOBUS\_GASS\_LENGTH\_UNKNOWN***, then the append may fail if the protocol does not support arbitrarily-length files.

***callback*** Function to call once the URL has been accepted, referred, or denied by the server.

***user\_arg*** User-supplied argument to the callback function.

**Return values:**

***GLOBUS\_SUCCESS*** The put was successfully initiated.

***GLOBUS\_GASS\_TRANSFER\_ERROR\_NULL\_POINTER*** One of *request*, *attr*, or *callback* was ***GLOBUS\_NULL***. The put could not be processed.

***GLOBUS\_GASS\_TRANSFER\_ERROR\_INTERNAL\_ERROR*** An internal resource was not available to process the put.

***GLOBUS\_GASS\_TRANSFER\_ERROR\_NOT\_IMPLEMENTED*** No protocol handler for doing a append on this URL type is implemented.

***GLOBUS\_GASS\_TRANSFER\_ERROR\_BAD\_URL*** The URL could not be parsed.

See also:

***globus\_gass\_transfer\_append()***(p. 6)

**4.2.2.6** ***int globus\_gass\_transfer\_append (globus\_gass\_transfer\_request\_t \* request, globus\_gass\_transfer\_requestattr\_t \* attr, char \* url, globus\_size\_t length)***

Blocking file append.

This function initiates a new "append" request of the file named by *url*. The entire file will be transferred to the server if the user is authorized to do so. This function blocks until the GASS library has determined whether the file may be retrieved by the caller, may not because it is a referral to another URL or URLs, or the server has denied the request.

Upon returning from this function, the request handle is initialized to refer to the *append* request's state.

If the server can't store the file at *url*, but has an alternative location for the user to store to, then this function return with request's status set to ***GLOBUS\_GASS\_TRANSFER\_REQUEST\_REFERRED***.

**Parameters:**

- request* A pointer to an uninitialized request handle.
- attr* Request attributes.
- url* URL to append to.
- length* The length of the data to append to the url, if known. If this parameter is set to *GLOBUS\_GASS\_LENGTH\_UNKNOWN*, then the append may fail if the protocol does not support arbitrarily-length files.

**Return values:**

- GLOBUS\_SUCCESS* The put was successfully initiated.
- GLOBUS\_GASS\_TRANSFER\_ERROR\_NULL\_POINTER* One of request, attr, or callback was GLOBUS\_NULL. The put could not be processed.
- GLOBUS\_GASS\_TRANSFER\_ERROR\_INTERNAL\_ERROR* An internal resource was not available to process the put.
- GLOBUS\_GASS\_TRANSFER\_ERROR\_NOT\_IMPLEMENTED* No protocol handler for doing a append on this URL type is implemented.
- GLOBUS\_GASS\_TRANSFER\_ERROR\_BAD\_URL* The URL could not be parsed.

**See also:**

`globus_gass_transfer_register_append()`(p. 6)

## 4.3 Implementing Servers

GASS Server Implementation API.

**Typedefs**

- typedef void(\* **globus\_gass\_transfer\_close\_callback\_t** )(void \*callback\_arg, globus\_gass\_transfer\_listener\_t listener)
- typedef void(\* **globus\_gass\_transfer\_listen\_callback\_t** )(void \*callback\_arg, globus\_gass\_transfer\_listener\_t listener)

**Functions**

- int **globus\_gass\_transfer\_create\_listener** (globus\_gass\_transfer\_listener\_t \*listener, globus\_gass\_transfer\_listenerattr\_t \*attr, char \*scheme)
- int **globus\_gass\_transfer\_close\_listener** (globus\_gass\_transfer\_listener\_t listener, **globus\_gass\_transfer\_close\_callback\_t** callback, void \*user\_arg)
- int **globus\_gass\_transfer\_register\_listen** (globus\_gass\_transfer\_listener\_t listener, **globus\_gass\_transfer\_listen\_callback\_t** callback, void \*user\_arg)
- int **globus\_gass\_transfer\_register\_accept** (**globus\_gass\_transfer\_request\_t** \*request, globus\_gass\_transfer\_requestattr\_t \*attr, globus\_gass\_transfer\_listener\_t listener, globus\_gass\_transfer\_callback\_t callback, void \*user\_arg)
- void \* **globus\_gass\_transfer\_listener\_get\_user\_pointer** (globus\_gass\_transfer\_listener\_t listener)
- int **globus\_gass\_transfer\_listener\_set\_user\_pointer** (globus\_gass\_transfer\_listener\_t listener, void \*user\_pointer)
- char \* **globus\_gass\_transfer\_listener\_get\_base\_url** (globus\_gass\_transfer\_listener\_t listener)
- int **globus\_gass\_transfer\_refer** (**globus\_gass\_transfer\_request\_t** request, char \*\*urls, globus\_size\_t num\_urls)
- int **globus\_gass\_transfer\_authorize** (**globus\_gass\_transfer\_request\_t** request, globus\_size\_t total\_length)
- int **globus\_gass\_transfer\_deny** (**globus\_gass\_transfer\_request\_t** request, int reason, char \*message)



### 4.3.1 Detailed Description

GASS Server Implementation API.

Another mode of using the GASS Transfer API is to implement data servers. The primary difference between the client and server parts of the GASS Transfer API are how requests are generated.

To implement a server, the user would call **globus\_gass\_transfer\_create\_listener()**(p. 8) to create a new server port on which a specific protocol will be used to request file transfer operations. The user may obtain the URL that the listener is bound to by calling **globus\_gass\_transfer\_listener\_get\_base\_url()**(p. 11).

Once the listener is created, the user can call **globus\_gass\_transfer\_register\_listen()**(p. 9) to wait for clients to connect to it. Once the server has detected an attempt to connect by a client, the user can call **globus\_gass\_transfer\_register\_accept()**(p. 10) to accept the connection from the client and parse the request.

In the callback associated with **globus\_gass\_transfer\_register\_accept()**(p. 10), the server can decide how to process the request. The user may choose to authorize the request by calling **globus\_gass\_transfer\_authorize()**(p. 12), refer it to another URL or URLs by calling **globus\_gass\_transfer\_refer()**(p. 11) or deny the client access to the URL by calling **globus\_gass\_transfer\_deny()**(p. 12).

### 4.3.2 Typedef Documentation

**4.3.2.1 typedef void(\* globus\_gass\_transfer\_close\_callback\_t)(void \*callback\_arg, globus\_gass\_transfer\_listener\_t listener)**

Listener close callback.

**Parameters:**

*callback\_arg*

*listener*

**4.3.2.2 typedef void(\* globus\_gass\_transfer\_listen\_callback\_t)(void \*callback\_arg, globus\_gass\_transfer\_listener\_t listener)**

Listen callback.

**Parameters:**

*callback\_arg*

*listener*

### 4.3.3 Function Documentation

**4.3.3.1 int globus\_gass\_transfer\_create\_listener (globus\_gass\_transfer\_listener\_t \*listener, globus\_gass\_transfer\_listenerattr\_t \*attr, char \*scheme)**

Create a new protocol-specific listener socket for a GASS server.

This function creates a new socket to listen for client connections as a GASS server. The listener handle pointer is initialized to contain the a new handle which can be used in subsequent server operations.

After calling this function, a user may call the **globus\_gass\_transfer\_register\_listen()**(p. 9) or **globus\_gass\_transfer\_close\_listener()**(p. 9) functions with this listener handle.

**Parameters:**

*listener* A new listener handle to initialize.

*attr* Protocol-specific attributes for the new listener.

*scheme* The protocol scheme to implement for the listener.

**Return values:**

**GLOBAL\_SUCCESS** The listener was successfully created.

**GLOBAL\_GASS\_TRANSFER\_ERROR\_NULL\_POINTER** The *listener* or *scheme* parameter was NULL

**GLOBAL\_GASS\_TRANSFER\_ERROR\_NOT\_IMPLEMENTED** The *scheme* is not supported by any protocol module.

**GLOBAL\_GASS\_TRANSFER\_ERROR\_MALLOC\_FAILED** Data structures associated with the transfer could not be allocated.

**4.3.3.2 int globus\_gass\_transfer\_close\_listener (globus\_gass\_transfer\_listener\_t listener, globus\_gass\_transfer\_close\_callback\_t callback, void \* user\_arg)**

Close a GASS listener.

This function calls the protocol specific function needed to close a GASS server listener port. Callbacks for any outstanding accepts will be called before the close callback is invoked.

**Parameters:**

*listener* Listener handle created by calling **globus\_gass\_transfer\_create\_listener()**(p. 8).

*callback* Function to call once the listener handle has been closed.

*user\_arg* Argument to be passed to the *callback* function.

**Return values:**

**GLOBAL\_SUCCESS** The close operation was successfully registered on the listener.

**GLOBAL\_GASS\_TRANSFER\_ERROR\_INVALID\_USE** The listener handle was invalid.

**GLOBAL\_GASS\_TRANSFER\_ERROR\_NOT\_INITIALIZED** The listener handle was not properly initialized.

**GLOBAL\_GASS\_TRANSFER\_ERROR\_DONE** A close has already been registered on the listener.

**4.3.3.3 int globus\_gass\_transfer\_register\_listen (globus\_gass\_transfer\_listener\_t listener, globus\_gass\_transfer\_listen\_callback\_t callback, void \* user\_arg)**

Listen for new client connections.

This function causes the listener handle to listen for new client connections. When one is ready, it calls the specified *callback* function, letting the server implementer continue to accept the connection and process the request.

**Parameters:**

*listener* The listener handle to register for new connections.

*callback* Function to call when a new connection may be accepted.

*user\_arg* Argument to be passed to the callback function.

**Return values:**

**GLOBAL\_SUCCESS** The listen callback has been registered with the protocol module.

**GLOBAL\_GASS\_TRANSFER\_ERROR\_INVALID\_USE** An invalid *listener* handle was passed to this function.

**GLOBAL\_GASS\_TRANSFER\_ERROR\_NOT\_INITIALIZED** An uninitialized *listener* handle was passed to this function.

**GLOBAL\_GASS\_TRANSFER\_ERROR\_ALREADY\_REGISTERED** The listener has already been registered for a new connection.

**GLOBAL\_GASS\_TRANSFER\_ERROR\_DONE** The listener has been registered for closing.

See also:

**globus\_gass\_transfer\_register\_accept**(p. 10);

**4.3.3.4 int globus\_gass\_transfer\_register\_accept (globus\_gass\_transfer\_request\_t \* *request*, globus\_gass\_transfer\_requestattr\_t \* *attr*, globus\_gass\_transfer\_listener\_t *listener*, globus\_gass\_transfer\_callback\_t *callback*, void \* *user\_arg*)**

Accept new client connections.

This function causes the listener handle to accept a new connection on the listener and parse the file request. Once the file request has been parsed, the specified *callback* function will be called. The server implementation must then either authorize, deny, or refer this request.

**Parameters:**

***request*** A pointer to a new request handle. This request handle will be initialized when the callback function is invoked.

***attr*** Request attributes.

***listener*** The listener handle to register for the new request.

***callback*** Function to call when the protocol module has parsed the file request.

***user\_arg*** Argument to be passed to the callback function.

**Return values:**

**GLOBAL\_SUCCESS** The listen callback has been registered with the protocol module.

**GLOBAL\_GASS\_TRANSFER\_ERROR\_INVALID\_USE** An invalid *listener* handle was passed to this function.

**GLOBAL\_GASS\_TRANSFER\_ERROR\_NOT\_INITIALIZED** An uninitialized *listener* handle was passed to this function.

**GLOBAL\_GASS\_TRANSFER\_ERROR\_INTERNAL\_ERROR** The request could not be initialized due to some internal resource depletion.

**GLOBAL\_GASS\_NOT\_REGISTERED.** The **globus\_gass\_transfer\_register\_listen**(p. 9) function has not yet been called.

**GLOBAL\_GASS\_ALREADY\_REGISTERED.** The listener is already processing a new request.

**GLOBAL\_GASS\_TRANSFER\_ERROR\_DONE** The listener has been registered for closing.

See also:

**globus\_gass\_transfer\_register\_listen**(p. 9);

**4.3.3.5 void\* globus\_gass\_transfer\_listener\_get\_user\_pointer (globus\_gass\_transfer\_listener\_t *listener*)**

Get the user pointer associated with a listener.

This function will query the listener's *user\_pointer* field and return its value.

**Parameters:**

***listener*** The listener handle.

**Returns:**

If the listener handle is invalid or the `user_pointer`'s value has not been set, then `GLOBUS_NULL` will be returned. Otherwise, the value of the user pointer will be returned.

**See also:**

`globus_gass_transfer_listener_set_user_pointer()`(p. 11)

**4.3.3.6 int globus\_gass\_transfer\_listener\_set\_user\_pointer (globus\_gass\_transfer\_listener\_t listener, void \* user\_pointer)**

Set the user pointer associated with a listener.

This function will set the listener's `user_pointer` field. The pointer may be used to associate any pointer-sized data with a listener handle.

**Parameters:**

*listener* The listener handle.

*user\_pointer* The value of the user pointer.

**Return values:**

*GLOBUS\_SUCCESS* The user pointer was successfully set.

*GLOBUS\_GASS\_TRANSFER\_ERROR\_INVALID\_USE* The *listener* handle was invalid.

**See also:**

`globus_gass_transfer_listener_get_user_pointer()`(p. 10)

**4.3.3.7 char\* globus\_gass\_transfer\_listener\_get\_base\_url (globus\_gass\_transfer\_listener\_t listener)**

Get the base URL of a listener.

This function queries a listener handle for the base URL which the server is listening on. For most protocols, this contains the protocol scheme, host, and port that the listener has registered itself on.

**Parameters:**

*listener* The listener handle to query.

**Returns:**

This function returns a pointer to a string containing the base URL. This string must not be freed or modified by the caller. It may not be referred to after the function `globus_gass_transfer_listener_close()` has been called.

**4.3.3.8 int globus\_gass\_transfer\_refer (globus\_gass\_transfer\_request\_t request, char \*\* urls, globus\_size\_t num\_urls)**

Refer a request.

This function causes the request to be referred to another URL or list of URLs. It should be called in response to a request accept callback when the server wants to refer the client to another server or servers to process the request.

**Parameters:**

*request* A new request handle, passed to the server in an accept callback.

*urls* An array of strings, each being a URL pointing to sources of the same data as the original URL.

*num\_urls* The length of the *urls* array.

**Return values:**

**GLOBAL\_GASS\_TRANSFER\_ERROR\_INVALID\_USE** The request handle was not valid, not created by calling `globus_gass_transfer_register_accept()`(p. 10), or has already been denied or authorized.

**GLOBAL\_GASS\_TRANSFER\_ERROR\_NOT\_IMPLEMENTED** The protocol module does not support referrals.

**See also:**

`globus_gass_transfer_deny()`(p. 12), `globus_gass_transfer_authorize()`(p. 12)

**4.3.3.9 int globus\_gass\_transfer\_authorize (globus\_gass\_transfer\_request\_t request, globus\_size\_t total\_length)**

Authorize a request.

This function causes the request to be authorized for processing. It should be called in response to a request accept callback when the server wants to agree to process this request. After calling this function, the server implementation should call `globus_gass_transfer_send_bytes()`(p. 13) or `globus_gass_transfer_receive_bytes()`(p. 14) to send or receive the data associated with the URL.

**Parameters:**

**request** A new request handle, passed to the server in an accept callback.

**total\_length** For a "get" request, the total\_length of the file to be retrieved if known. This value may be `GLOBAL_GASS_LENGTH_UNKNOWN` if the protocol supports transferring arbitrarily-sized files.

**Return values:**

**GLOBAL\_GASS\_TRANSFER\_ERROR\_INVALID\_USE** The request handle was not valid, not created by calling `globus_gass_transfer_register_accept()`(p. 10), or has already been denied or authorized.

**GLOBAL\_GASS\_TRANSFER\_ERROR\_NOT\_IMPLEMENTED** The protocol module does not support authorizing requests.

**See also:**

`globus_gass_transfer_refer()`(p. 11), `globus_gass_transfer_deny()`(p. 12)

**4.3.3.10 int globus\_gass\_transfer\_deny (globus\_gass\_transfer\_request\_t request, int reason, char \* message)**

Deny a request.

This function causes the request to be denied for further processing. It should be called in response to a request accept callback when the server wants to refuse processing this request for the client. After calling this function, the server implementation need do nothing further with the request handle.

**Parameters:**

**request** A new request handle, passed to the server in an accept callback.

**reason** A protocol-specific reason code.

**message** An informational message to be sent to the client.

**Return values:**

**GLOBAL\_GASS\_TRANSFER\_ERROR\_INVALID\_USE** The request handle was not valid, not created by calling `globus_gass_transfer_register_accept()`(p. 10), or has already been denied or authorized.

**GLOBAL\_GASS\_TRANSFER\_ERROR\_NOT\_IMPLEMENTED** The protocol module does not support denying requests.

**See also:**

`globus_gass_transfer_refer()`(p. 11), `globus_gass_transfer_authorize()`(p. 12)

## 4.4 Sending and Receiving Data

### Typedefs

- typedef void(\* **globus\_gass\_transfer\_bytes\_callback\_t**)(void \*arg, **globus\_gass\_transfer\_request\_t** request, globus\_byte\_t \*bytes, globus\_size\_t length, globus\_bool\_t last\_data)

### Functions

- int **globus\_gass\_transfer\_send\_bytes** (**globus\_gass\_transfer\_request\_t** request, globus\_byte\_t \*bytes, globus\_size\_t send\_length, globus\_bool\_t last\_data, **globus\_gass\_transfer\_bytes\_callback\_t** callback, void \*user\_arg)
- int **globus\_gass\_transfer\_receive\_bytes** (**globus\_gass\_transfer\_request\_t** request, globus\_byte\_t \*bytes, globus\_size\_t max\_length, globus\_size\_t wait\_for\_length, **globus\_gass\_transfer\_bytes\_callback\_t** callback, void \*user\_arg)

#### 4.4.1 Typedef Documentation

**4.4.1.1** typedef void(\* **globus\_gass\_transfer\_bytes\_callback\_t**)(void \*arg, **globus\_gass\_transfer\_request\_t** request, globus\_byte\_t \*bytes, globus\_size\_t length, globus\_bool\_t last\_data)

Byte send or receive callback function.

#### Parameters:

*arg* The user\_arg passed to the function which registered this callback. The user may use this value for any purpose.

*request* The request handle associated with this byte array.

*bytes* The byte array which was sent or received.

*length* The length of data which was sent or received.

*last\_data* Boolean flag whether this is the final byte array for this request.

#### See also:

**globus\_gass\_transfer\_send\_bytes**(p. 13), **globus\_gass\_transfer\_receive\_bytes**(p. 14)

#### 4.4.2 Function Documentation

**4.4.2.1** int **globus\_gass\_transfer\_send\_bytes** (**globus\_gass\_transfer\_request\_t** request, globus\_byte\_t \*bytes, globus\_size\_t send\_length, globus\_bool\_t last\_data, **globus\_gass\_transfer\_bytes\_callback\_t** callback, void \*user\_arg)

Send a byte array associated with a request handle.

This function sends a block of data to a server or client as part of the processing for a request. Multiple data blocks may be registered with the GASS transfer library at once.

When processing a server request, this function may only be used in conjunction with "get" requests. The user must call **globus\_gass\_transfer\_authorize**(p. 12) before calling this function.

When processing a client request, this function may only be used in conjunction with "put" or "append" requests. This function may not be called before either the callback function has been invoked, or the blocking **globus\_gass\_transfer\_put**() or **globus\_gass\_transfer\_append**(p. 6) function has returned.

#### Parameters:

*request* The request handle with which this block of bytes is associated.

*bytes* A user-supplied buffer containing the data associated with the request.

***send\_length*** The length of the *bytes* array.

***last\_data*** A flag to indicate whether this is the final block of data for the request. If this is true, then the *callback* function will be delayed until the server acknowledges that the file has been completely received.

***callback*** Function to call once the *bytes* array has been sent.

***user\_arg*** Argument to be passed to the *callback* function.

**Return values:**

***GLOBUS\_SUCCESS*** The *bytes* array was successfully registered with the GASS transfer library. The *callback* function will be invoked once it has been sent.

***GLOBUS\_GASS\_TRANSFER\_ERROR\_NULL\_POINTER*** The *bytes* or *callback* parameter was NULL.

***GLOBUS\_GASS\_TRANSFER\_ERROR\_INVALID\_USER*** The *request* was invalid, or it is not one on which bytes can be sent.

***GLOBUS\_GASS\_TRANSFER\_ERROR\_NOT\_INITIALIZED*** The callback to a non-blocking file request has not been invoked yet, a blocking file request has not returned, or the request has not yet been authorized.

***GLOBUS\_GASS\_TRANSFER\_ERROR\_REQUEST\_FAILED*** The *request* has failed by either the client, server, or protocol module implementation.

***GLOBUS\_GASS\_TRANSFER\_ERROR\_DONE*** The *request* has already been completed.

**4.4.2.2** `int globus_gass_transfer_receive_bytes (globus_gass_transfer_request_t request, globus_byte_t * bytes, globus_size_t max_length, globus_size_t wait_for_length, globus_gass_transfer_bytes_callback_t callback, void * user_arg)`

Receive a byte array associated with a request handle.

This function receives a block of data from a server or client as part of the processing for a request. Multiple data blocks may be registered with the GASS transfer library at once.

When processing a server request, this function may only be used in conjunction with "put" or "append" requests. The user must call `globus_gass_transfer_authorize()`(p. 12) before calling this function.

When processing a client request, this function may only be used in conjunction with "get" requests. This function may not be called before either the callback function has been invoked, or the blocking `globus_gass_transfer_put()` or `globus_gass_transfer_append()`(p. 6) function has returned.

**Parameters:**

***request*** The request handle with which this block of bytes is associated.

***bytes*** A user-supplied buffer containing the data associated with the request.

***max\_length*** The length of the *bytes* array.

***wait\_for\_length*** The minimum amount of data to wait for before invoking the *callback* function. A partial byte array of at least this amount will be returned in the callback, unless end-of-file is reached before this amount.

***callback*** Function to call once the *bytes* array has been received.

***user\_arg*** Argument to be passed to the *callback* function.

**Return values:**

***GLOBUS\_SUCCESS*** The *bytes* array was successfully registered with the GASS transfer library. The *callback* function will be invoked once it has been received.

***GLOBUS\_GASS\_TRANSFER\_ERROR\_NULL\_POINTER*** The *bytes* or *callback* parameter was NULL.

***GLOBUS\_GASS\_TRANSFER\_ERROR\_INVALID\_USER*** The *request* was invalid, or it is not one on which bytes can be sent.

***GLOBUS\_GASS\_TRANSFER\_ERROR\_NOT\_INITIALIZED*** The callback to a non-blocking file request has not been invoked yet, a blocking file request has not returned, or the request has not yet been authorized.

***GLOBUS\_GASS\_TRANSFER\_ERROR\_REQUEST\_FAILED*** The *request* has failed by either the client, server, or protocol module implementation.

***GLOBUS\_GASS\_TRANSFER\_ERROR\_DONE*** The *request* has already been completed.

## 4.5 Referrals

The GASS Transfer API supports referring URL requests to alternate URLs via referrals.

### Functions

- globus\_size\_t **globus\_gass\_transfer\_referral\_get\_count** (globus\_gass\_transfer\_referral\_t \*referral)
- char \* **globus\_gass\_transfer\_referral\_get\_url** (globus\_gass\_transfer\_referral\_t \*referral, globus\_size\_t index)
- int **globus\_gass\_transfer\_referral\_destroy** (globus\_gass\_transfer\_referral\_t \*referral)

#### 4.5.1 Detailed Description

The GASS Transfer API supports referring URL requests to alternate URLs via referrals.

Referrals are essentially pointers to another URL or URLs which contain the same file as the original location which a client has requested of a server. Referrals may span multiple protocol schemes, though not all protocols may be able to generate referrals. For example, an HTTP server may refer a client to another HTTP server, an HTTPS server.

Upon receiving a referred response from a server, a client should query the request handle to determine from where the file can be retrieved.

#### 4.5.2 Function Documentation

##### 4.5.2.1 globus\_size\_t globus\_gass\_transfer\_referral\_get\_count (globus\_gass\_transfer\_referral\_t \* *referral*)

Get the number of URLs in this referral.

This function examines the referral to determine if the number of URLs which are contained in it. Each of these URLs should either point to another referral, or to a URL containing the equivalent file as the original URL request which caused this referral.

##### Parameters:

*referral* The referral structure to query.

##### Returns:

This function returns the number of URL entries in the referral, or 0, if there are none.

##### 4.5.2.2 char\* globus\_gass\_transfer\_referral\_get\_url (globus\_gass\_transfer\_referral\_t \* *referral*, globus\_size\_t *index*)

Get a URL string from a referral.

This function examines the referral to retrieve a URL string from it. A valid referral will contain one or more strings. They are indexed from 0 to the value returned by **globus\_gass\_transfer\_referral\_get\_count()**(p. 15) - 1.



The string returned by this function must not be freed by the caller. It will remain valid until the referral structure is destroyed.

**Parameters:**

- referral* The referral structure to query.
- index* The URL to extract from the referral.

**Returns:**

This function returns a string pointer containing the URL, or NULL if the index or referral were invalid.

#### 4.5.2.3 `int globus_gass_transfer_referral_destroy (globus_gass_transfer_referral_t * referral)`

Free all memory used by a referral.

This function frees all memory used by this referral. After calling this function, the strings returned by calling `globus_gass_transfer_referral_get_url()`(p. 15) must not be accessed. Any further attempts to extract information from this referral will fail.

**Parameters:**

- referral* The referral to destroy.

**Return values:**

- GLOBUS\_SUCCESS** The referral was successfully destroyed.
- GLOBUS\_GASS\_TRANSFER\_ERROR\_NULL\_POINTER** The referral parameter was GLOBUS\_NULL. It could not be destroyed.

## 4.6 Request Handles

Request handles are used by the GASS Transfer API to associate operations with a single file transfer request.

### Data Structures

- struct `globus_gass_transfer_request_t`

### Enumerations

- enum `globus_gass_transfer_request_type_t` {  
    **GLOBUS\_GASS\_TRANSFER\_REQUEST\_TYPE\_INVALID,**  
    **GLOBUS\_GASS\_TRANSFER\_REQUEST\_TYPE\_GET,**  
    **GLOBUS\_GASS\_TRANSFER\_REQUEST\_TYPE\_PUT,**  
    **GLOBUS\_GASS\_TRANSFER\_REQUEST\_TYPE\_APPEND** }  
• enum `globus_gass_transfer_request_status_t` {  
    **GLOBUS\_GASS\_TRANSFER\_REQUEST\_INVALID,**  
    **GLOBUS\_GASS\_TRANSFER\_REQUEST\_STARTING,**  
    **GLOBUS\_GASS\_TRANSFER\_REQUEST\_PENDING,**  
    **GLOBUS\_GASS\_TRANSFER\_REQUEST\_FAILED,**  
    **GLOBUS\_GASS\_TRANSFER\_REQUEST\_REFERRED,**  
    **GLOBUS\_GASS\_TRANSFER\_REQUEST\_DENIED,**  
    **GLOBUS\_GASS\_TRANSFER\_REQUEST\_DONE** }

## Functions

- `globus_gass_transfer_request_type_t globus_gass_transfer_request_get_type (globus_gass_transfer_request_t request)`
- `void * globus_gass_transfer_request_get_user_pointer (globus_gass_transfer_request_t request)`
- `int globus_gass_transfer_request_set_user_pointer (globus_gass_transfer_request_t request, void *user_pointer)`
- `globus_gass_transfer_request_status_t globus_gass_transfer_request_get_status (globus_gass_transfer_request_t request)`
- `int globus_gass_transfer_request_get_referral (globus_gass_transfer_request_t request, globus_gass_transfer_referral_t *referral)`
- `char * globus_gass_transfer_request_get_url (globus_gass_transfer_request_t request)`
- `globus_size_t globus_gass_transfer_request_get_length (globus_gass_transfer_request_t request)`
- `int globus_gass_transfer_request_set_type (globus_gass_transfer_request_t request, globus_gass_transfer_request_type_t type)`
- `int globus_gass_transfer_request_set_url (globus_gass_transfer_request_t request, char *url)`
- `void globus_gass_transfer_request_set_length (globus_gass_transfer_request_t request, globus_size_t length)`
- `int globus_gass_transfer_request_get_denial_reason (globus_gass_transfer_request_t request)`
- `char * globus_gass_transfer_request_get_denial_message (globus_gass_transfer_request_t request)`
- `char * globus_gass_transfer_request_get_subject (globus_gass_transfer_request_t request)`
- `int globus_gass_transfer_request_destroy (globus_gass_transfer_request_t request)`

### 4.6.1 Detailed Description

Request handles are used by the GASS Transfer API to associate operations with a single file transfer request.

Specifically, they are used to register multiple byte range buffers with a file transfer request, and to query the state of a transfer in-progress.

To implement a server, the request handle is populated by the protocol module implementation. The server may use the functions in this section to determine information about what the client is requesting.

To implement a client, the request handle should be queried after the blocking call or initial callback has been invoked to determine if the request has been authorized or referred, and after EOF, to determine whether the request has completed successfully.

A request handle contains a pointer which may be used by the handler of the request to store a pointer to arbitrary application-specific data.

### 4.6.2 Enumeration Type Documentation

#### 4.6.2.1 enum globus\_gass\_transfer\_request\_type\_t

Type of operation associated with a request handle.

Enumeration values:

***GLOBUS\_GASS\_TRANSFER\_REQUEST\_TYPE\_INVALID*** Handle no longer valid.

***GLOBUS\_GASS\_TRANSFER\_REQUEST\_TYPE\_GET*** A get request.

***GLOBUS\_GASS\_TRANSFER\_REQUEST\_TYPE\_PUT*** A put request.

***GLOBUS\_GASS\_TRANSFER\_REQUEST\_TYPE\_APPEND*** An append request.

#### 4.6.2.2 enum globus\_gass\_transfer\_request\_status\_t

Request Status.

##### Enumeration values:

**GLOBUS\_GASS\_TRANSFER\_REQUEST\_INVALID** Handle is no longer valid.

**GLOBUS\_GASS\_TRANSFER\_REQUEST\_STARTING** Initial connection and authorization is not yet completed.

**GLOBUS\_GASS\_TRANSFER\_REQUEST\_PENDING** Request is authorized.

**GLOBUS\_GASS\_TRANSFER\_REQUEST\_FAILED** Request failed due to protocol error or client or server aborting the request.

**GLOBUS\_GASS\_TRANSFER\_REQUESTREFERRED** Request can not be processed by this server, referred to another URL or URLs.

**GLOBUS\_GASS\_TRANSFER\_REQUEST\_DENIED** The server denied this request.

**GLOBUS\_GASS\_TRANSFER\_REQUEST\_DONE** All callbacks have completed.

#### 4.6.3 Function Documentation

##### 4.6.3.1 globus\_gass\_transfer\_request\_type\_t globus\_gass\_transfer\_request\_get\_type (globus\_gass\_transfer\_request\_t request)

Determine the type of a request.

This function is used by GASS server implementations to discover what type of operation the client is requesting for an URL.

##### Parameters:

*request* The request to query.

##### Returns:

The **type**(p. 17) of the request.

##### 4.6.3.2 void\* globus\_gass\_transfer\_request\_get\_user\_pointer (globus\_gass\_transfer\_request\_t request)

Get the user pointer associated with a request.

This function extracts the user pointer from a request handle. The user-pointer may be used by the application which is generating or servicing the request to store a pointer to any application-specific piece of data.

##### Parameters:

*request* The request to query.

##### Returns:

The user pointer's value.

##### 4.6.3.3 int globus\_gass\_transfer\_request\_set\_user\_pointer (globus\_gass\_transfer\_request\_t request, void \* user\_pointer)

Set the user pointer associated with a request handle.

This function sets the user pointer from a request handle. The user-pointer may be used by the application which is generating or servicing the request to store a pointer to any application-specific piece of data.

**Parameters:**

*request* The request to modify.

*user\_pointer* The new value of the user pointer for the request.

**Return values:**

**GLOBAL\_SUCCESS** The user pointer's value was set.

**GLOBAL\_GASS\_TRANSFER\_ERROR\_INVALID\_USE** An invalid request handle was passed to this function

#### 4.6.3.4 **globus\_gass\_transfer\_request\_status\_t globus\_gass\_transfer\_request\_get\_status (globus\_gass\_transfer\_request\_t request)**

Check the status of a request.

This function queries a request to determine the status of the request. This function should be called after EOF has been reached, or after the initial get, put, or append has returned or had its callback function called to determine if it is possible to proceed, or whether the file transfer was successfully processed.

**Parameters:**

*request* The request handle to query.

**Returns:**

A **globus\_gass\_transfer\_request\_status\_t**(p. 18) indicating the current status of the request.

#### 4.6.3.5 **int globus\_gass\_transfer\_request\_get\_referral (globus\_gass\_transfer\_request\_t request, globus\_gass\_transfer\_referral\_t \* referral)**

Extract referral information from a request handle.

This function queries the request handle to determine any referral information that it contains. This function should only be called on request handles in the **GLOBAL\_GASS\_TRANSFER\_REQUEST\_REFERRED** state. If no referral information is stored in the request handle, then the referral will be initialized to an empty referral. The referral must be destroyed by calling **globus\_gass\_transfer\_referral\_destroy()**(p. 16) by the caller.

**Parameters:**

*request* The request handle to query.

*referral* A pointer to an uninitialized referral structure. It will be populated by calling this function.

**Return values:**

**GLOBAL\_SUCCESS** The referral was successfully extracted from the request handle.

**GLOBAL\_GASS\_TRANSFER\_ERROR\_NULL\_POINTER** The referral pointer was **GLOBAL\_NULL**;

#### 4.6.3.6 **char\* globus\_gass\_transfer\_request\_get\_url (globus\_gass\_transfer\_request\_t request)**

Get the URL from a request handle.

This function queries the request handle to determine the URL associated with the request. This function is intended to be useful to GASS server implementors.

**Parameters:**

*request* The request handle to query.

**Returns:**

A pointer to the URL, or **GLOBAL\_NULL** if the request handle is invalid. The string which is returned must not be freed by the caller. It may not be accessed after the request has been destroyed.

#### 4.6.3.7 `globus_size_t globus_gass_transfer_request_get_length (globus_gass_transfer_request_t request)`

Get the length of a file to be transferred using GASS.

This function queries the request handle to determine the amount of data that will be transferred to copy the URL. The length may be `GLOBUS_GASS_TRANSFER_LENGTH_UNKNOWN` if the sender can not determine the length before making or authorizing the request.

**Parameters:**

*request* The request to query.

**Returns:**

The length of the file located at the request's URL, or `GLOBUS_GASS_TRANSFER_LENGTH_UNKNOWN` if that cannot be determined.

#### 4.6.3.8 `int globus_gass_transfer_request_set_type (globus_gass_transfer_request_t request, globus_gass_transfer_request_type_t type)`

Set the type of a request.

This function modifies a request handle by setting the type of operation that it is being used for. This function may only be called once per handle, and only from a GASS protocol module implementation.

**Parameters:**

*request* The request handle to modify.

*type* The type of operation that this request handle will be used for.

**Return values:**

`GLOBUS_SUCCESS` The request handle's type has been set.

`GLOBUS_GASS_TRANSFER_ERROR_INVALID_USE` The request handle was invalid or it's type was already set. The request handle was not modified.

**Note:**

Only GASS Protocol modules may call this function.

#### 4.6.3.9 `int globus_gass_transfer_request_set_url (globus_gass_transfer_request_t request, char * url)`

Set the URL to which a request handle refers.

This function modifies the given request handle so that it's URL field is set to string pointed to by *url*.

No copy is made of the string, so the caller must not free it. It must be allocated by calling one of the memory allocators in `globus_libc`, as it will be freed when the request handle is destroyed.

This function must only be called by protocol modules when constructing a request handle when accepting a new request. This function can only be called once per request handle.

**Parameters:**

*request* A handle to the request to modify.

*url* A string containing the URL that this request will be associated with.

**Return values:**

`GLOBUS_SUCCESS` The URL was set for the request handle.

`GLOBUS_GASS_TRANSFER_ERROR_INVALID_USE` The request handle was invalid, or the URL had already been set.

#### **4.6.3.10 void globus\_gass\_transfer\_request\_set\_length (globus\_gass\_transfer\_request\_t *request*, globus\_size\_t *length*)**

Set the length of a transfer associated request handle.

This function modifies the given request handle so that its length field is set to give length parameter.

This function must only be called by protocol modules when constructing a request handle when receiving the response to a get request. This function can only be called once per request handle.

##### **Parameters:**

*request* A handle to the request to modify.

*length* The length of the file request.

##### **Return values:**

**GLOBAL\_SUCCESS** The URL was set for the request handle.

**GLOBAL\_GASS\_TRANSFER\_ERROR\_INVALID\_USE** The request handle was invalid, or the URL had already been set.

#### **4.6.3.11 int globus\_gass\_transfer\_request\_get\_denial\_reason (globus\_gass\_transfer\_request\_t *request*)**

Get an integer code describing why the request was denied.

This function queries a request which was denied by a server to determine why it was denied. The denial reason will be expressed in a protocol-specific response code. Knowledge of the protocol is needed to understand this response.

##### **Parameters:**

*request* A handle to the request to query.

##### **Returns:**

A protocol-specific integer indicating why the request was denied. If the request handle is invalid or the request was not denied, then this function returns 0.

##### **See also:**

`globus_gass_transfer_request_get_denial_message()`(p. 21)

#### **4.6.3.12 char\* globus\_gass\_transfer\_request\_get\_denial\_message (globus\_gass\_transfer\_request\_t *request*)**

Get an string describing why a request was denied.

This function queries a request which was denied by a server to determine why it was denied. The denial reason will be expressed as a response string. The string must be freed by the caller.

##### **Parameters:**

*request* A handle to the request to query.

##### **Returns:**

A string indicating why the request was denied. If the request handle is invalid or the request was not denied, then this function returns GLOBAL\_NULL.

##### **See also:**

`globus_gass_transfer_request_get_denial_reason()`(p. 21)

#### 4.6.3.13 `char* globus_gass_transfer_request_get_subject (globus_gass_transfer_request_t request)`

Get the subject string associated with a request.

This function queries a request handle to determine the subject identity of the client who initiated the request. The string must not be freed by the caller.

##### Parameters:

*request* A handle to the request to query.

##### Returns:

A string containing the request initiator's subject identity. If the request handle is invalid or a credential was not used to initiate the request, this value will be `GLOBUS_NULL`.

#### 4.6.3.14 `int globus_gass_transfer_request_destroy (globus_gass_transfer_request_t request)`

Destroy a request handle.

This function destroys the caller's reference to a request handle. It must be called for all request handles which are created by calling functions in the "@ref globus\_gass\_transfer\_client" or "@ref globus\_gass\_transfer\_server" sections of this manual. After calling the function, the caller must not attempt to use the request handle for any purpose.

##### Parameters:

*request* The request to destroy.

##### Return values:

***GLOBUS\_SUCCESS*** The request handle reference was successfully destroyed.

***GLOBUS\_GASS\_TRANSFER\_ERROR\_INVALID\_USE*** Either an invalid request handle or one which is actively being used was passed to this function as the *request* parameter.

## 4.7 Request Attributes

The GASS Transfer library uses Globus objects to provide an extensible way of creating protocol-specific attributes.

### Proxy Server

- `int globus_gass_transfer_requestattr_set_proxy_url (globus_gass_transfer_requestattr_t *attr, char *proxy_url)`
- `int globus_gass_transfer_requestattr_get_proxy_url (globus_gass_transfer_requestattr_t *attr, char **proxy_url)`

### Block Size

- `int globus_gass_transfer_requestattr_set_block_size (globus_gass_transfer_requestattr_t *attr, globus_size_t block_size)`
- `int globus_gass_transfer_requestattr_get_block_size (globus_gass_transfer_requestattr_t *attr, globus_size_t *block_size)`

## File Mode

- int **globus\_gass\_transfer\_requestattr\_set\_file\_mode** (globus\_gass\_transfer\_requestattr\_t \*attr, globus\_gass\_transfer\_file\_mode\_t file\_mode)
- int **globus\_gass\_transfer\_requestattr\_get\_file\_mode** (globus\_gass\_transfer\_requestattr\_t \*attr, globus\_gass\_transfer\_file\_mode\_t \*file\_mode)

## Connection Reuse

- int **globus\_gass\_transfer\_requestattr\_set\_connection\_reuse** (globus\_gass\_transfer\_requestattr\_t \*attr, globus\_bool\_t connection\_reuse)
- int **globus\_gass\_transfer\_requestattr\_get\_connection\_reuse** (globus\_gass\_transfer\_requestattr\_t \*attr, globus\_bool\_t \*connection\_reuse)

## Socket Send Buffer Size

- int **globus\_gass\_transfer\_requestattr\_set\_socket\_sndbuf** (globus\_gass\_transfer\_requestattr\_t \*attr, int sndbuf)
- int **globus\_gass\_transfer\_requestattr\_get\_socket\_sndbuf** (globus\_gass\_transfer\_requestattr\_t \*attr, int \*sndbuf)

## Receive Socket Buffer

- int **globus\_gass\_transfer\_requestattr\_set\_socket\_rcvbuf** (globus\_gass\_transfer\_requestattr\_t \*attr, int rcvbuf)
- int **globus\_gass\_transfer\_requestattr\_get\_socket\_rcvbuf** (globus\_gass\_transfer\_requestattr\_t \*attr, int \*rcvbuf)

## TCP Nodelay

- int **globus\_gass\_transfer\_requestattr\_set\_socket\_nodelay** (globus\_gass\_transfer\_requestattr\_t \*attr, globus\_bool\_t nodelay)
- int **globus\_gass\_transfer\_requestattr\_get\_socket\_nodelay** (globus\_gass\_transfer\_requestattr\_t \*attr, globus\_bool\_t \*nodelay)

## Authorization

- int **globus\_gass\_transfer\_secure\_requestattr\_set\_authorization** (globus\_gass\_transfer\_requestattr\_t \*attr, globus\_gass\_transfer\_authorization\_t mode, char \*subject)
- int **globus\_gass\_transfer\_secure\_requestattr\_get\_authorization** (globus\_gass\_transfer\_requestattr\_t \*attr, globus\_gass\_transfer\_authorization\_t \*mode, char \*\*subject)

## Functions

- int **globus\_gass\_transfer\_requestattr\_init** (globus\_gass\_transfer\_requestattr\_t \*attr, char \*url\_scheme)
- int **globus\_gass\_transfer\_requestattr\_destroy** (globus\_gass\_transfer\_requestattr\_t \*attr)

### 4.7.1 Detailed Description

The GASS Transfer library uses Globus objects to provide an extensible way of creating protocol-specific attributes.



## 4.7.2 Function Documentation

### 4.7.2.1 `int globus_gass_transfer_requestattr_init (globus_gass_transfer_requestattr_t * attr, char * url_scheme)`

Initialize a request attribute.

This function initializes the *attr* to contain a new protocol-specific request attribute.

#### Parameters:

*attr* The attribute set to be initialized.

*url\_scheme* The scheme which the attribute will be used for.

#### Return values:

**GLOBAL\_SUCCESS** The attribute was successfully initialized.

**GLOBAL\_GASS\_TRANSFER\_ERROR\_NULL\_POINTER** Either *attr* or *url\_scheme* was GLOBAL\_NULL.

**GLOBAL\_GASS\_TRANSFER\_ERROR\_NOT\_IMPLEMENTED** No protocol module currently registered with GASS Transfer Library handles URLs with the specified *url\_scheme*.

### 4.7.2.2 `int globus_gass_transfer_requestattr_destroy (globus_gass_transfer_requestattr_t * attr)`

Destroy a request attribute.

This function destroys the attribute set specified in *attr*.

#### Parameters:

*attr* The attribute set to be destroyed.

#### Return values:

**GLOBAL\_SUCCESS** The attribute was successfully destroyed.

**GLOBAL\_GASS\_TRANSFER\_ERROR\_NULL\_POINTER** The *attr* was GLOBAL\_NULL.

### 4.7.2.3 `int globus_gass_transfer_requestattr_set_proxy_url (globus_gass_transfer_requestattr_t * attr, char * proxy_url)`

Set/Get the proxy server attribute for a GASS transfer attribute set.

This attribute allows the user to use a proxy server to handle a URL request.

#### Parameters:

*attr* The attribute set to be modified

*proxy\_url* The new value of the proxy\_url attribute. This may be GLOBAL\_NULL if no proxy is to be used to access URLs with this attribute set.

#### Return values:

**GLOBAL\_SUCCESS** The attribute was successfully updated.

**GLOBAL\_GASS\_TRANSFER\_ERROR\_NULL\_POINTER** The *attr* was GLOBAL\_NULL.

#### 4.7.2.4 `int globus_gass_transfer_requestattr_set_block_size (globus_gass_transfer_requestattr_t * attr, globus_size_t block_size)`

Set/Get the block size attribute for a GASS transfer attribute set.

This attribute allows the user to suggest a preferred block size of a server to handle a URL request.

##### Parameters:

*attr* The attribute set to query or modify.

*block\_size* The data block size that should be used to process requests with this attribute set.

##### Return values:

**GLOBAL\_SUCCESS** The attribute was successfully updated.

**GLOBAL\_GASS\_TRANSFER\_ERROR\_NULL\_POINTER** The *attr* was GLOBAL\_NULL.

#### 4.7.2.5 `int globus_gass_transfer_requestattr_set_file_mode (globus_gass_transfer_requestattr_t * attr, globus_gass_transfer_file_mode_t file_mode)`

Set/Get the file mode attribute for a GASS transfer attribute set.

This attribute allows the user to control whether the file will be transferred in ASCII or binary file mode.

##### Parameters:

*attr* The attribute set to query or modify.

*file\_mode* The value of the file mode attribute.

##### Return values:

**GLOBAL\_SUCCESS** The attribute was successfully updated.

**GLOBAL\_GASS\_TRANSFER\_ERROR\_NULL\_POINTER** The *attr* was GLOBAL\_NULL.

#### 4.7.2.6 `int globus_gass_transfer_requestattr_set_connection_reuse (globus_gass_transfer_requestattr_t * attr, globus_bool_t connection_reuse)`

Set/Get the connection reuse attribute for a GASS transfer attribute set.

This attribute allows the user to control whether the connection associated with a GASS Transfer request should be reused after the file transfer has completed.

##### Parameters:

*attr* The attribute set to query or modify.

*connection\_reuse* The value of the connection reuse attribute.

##### Return values:

**GLOBAL\_SUCCESS** The attribute was successfully updated.

**GLOBAL\_GASS\_TRANSFER\_ERROR\_NULL\_POINTER** The *attr* was GLOBAL\_NULL.

#### 4.7.2.7 `int globus_gass_transfer_requestattr_set_socket_sndbuf (globus_gass_transfer_requestattr_t * attr, int sndbuf)`

Set/Get the send buffer size attribute for a GASS transfer attribute set.

This attribute allows the user to control the socket send buffer associated with a GASS Transfer request should be reused after the file transfer has completed.

**Parameters:**

*attr* The attribute set to query or modify.

*sndbuf* The value of the socket buffer.

**Return values:**

**GLOBUS\_SUCCESS** The attribute was successfully updated.

**GLOBUS\_GASS\_TRANSFER\_ERROR\_NULL\_POINTER** The *attr* was GLOBUS\_NULL.

**4.7.2.8 int globus\_gass\_transfer\_requestattr\_set\_socket\_rcvbuf (globus\_gass\_transfer\_requestattr\_t \* attr, int rcvbuf)**

Set/Get the receive buffer size attribute for a GASS transfer attribute set.

This attribute allows the user to control the socket receive buffer associated with a GASS Transfer request should be reused after the file transfer has completed.

**Parameters:**

*attr* The attribute set to query or modify.

*rcvbuf* The value of the socket buffer.

**Return values:**

**GLOBUS\_SUCCESS** The attribute was successfully updated.

**GLOBUS\_GASS\_TRANSFER\_ERROR\_NULL\_POINTER** The *attr* was GLOBUS\_NULL.

**4.7.2.9 int globus\_gass\_transfer\_requestattr\_set\_socket\_nodelay (globus\_gass\_transfer\_requestattr\_t \* attr, globus\_bool\_t nodelay)**

Set/Get the TCP nodelay attribute for a GASS transfer attribute set.

This attribute allows the user to control the socket receive buffer associated with a GASS Transfer request should be reused after the file transfer has completed.

**Parameters:**

*attr* The attribute set to query or modify.

*nodelay* The value of the nodelay attribute.

**Return values:**

**GLOBUS\_SUCCESS** The attribute was successfully updated.

**GLOBUS\_GASS\_TRANSFER\_ERROR\_NULL\_POINTER** The *attr* was GLOBUS\_NULL.

**4.7.2.10 int globus\_gass\_transfer\_secure\_requestattr\_set\_authorization (globus\_gass\_transfer\_requestattr\_t \* attr, globus\_gass\_transfer\_authorization\_t mode, char \* subject)**

Set/Get the authorization attribute for a GASS transfer attribute set.

This attribute allows the user to control what type of authorization should be done when GASS Transfer requests are processed.

**Parameters:**

*attr* The attribute set to query or modify.

*mode* The authorization mode to use.

*subject* The subject name of the authorized subject, if *mode* is GLOBUS\_GASS\_TRANSFER\_AUTHORIZE\_SUBJECT

**Return values:**

**GLOBUS\_SUCCESS** The attribute was successfully updated.

**GLOBUS\_GASS\_TRANSFER\_ERROR\_NULL\_POINTER** The *attr* was GLOBUS\_NULL.

**4.7.2.11** `int globus_gass_transfer_requestattr_get_proxy_url (globus_gass_transfer_requestattr_t * attr, char ** proxy_url)`

Set/Get the proxy server attribute for a GASS transfer attribute set.

This attribute allows the user to use a proxy server to handle a URL request.

**Parameters:**

*attr* The attribute set to be modified

*proxy\_url* The new value of the proxy\_url attribute. This may be GLOBUS\_NULL if no proxy is to be used to access URLs with this attribute set.

**Return values:**

**GLOBUS\_SUCCESS** The attribute was successfully updated.

**GLOBUS\_GASS\_TRANSFER\_ERROR\_NULL\_POINTER** The *attr* was GLOBUS\_NULL.

**4.7.2.12** `int globus_gass_transfer_requestattr_get_block_size (globus_gass_transfer_requestattr_t * attr, globus_size_t * block_size)`

Set/Get the block size attribute for a GASS transfer attribute set.

This attribute allows the user to suggest a preferred block size of a server to handle a URL request.

**Parameters:**

*attr* The attribute set to query or modify.

*block\_size* The data block size that should be used to process requests with this attribute set.

**Return values:**

**GLOBUS\_SUCCESS** The attribute was successfully updated.

**GLOBUS\_GASS\_TRANSFER\_ERROR\_NULL\_POINTER** The *attr* was GLOBUS\_NULL.

**4.7.2.13** `int globus_gass_transfer_requestattr_get_file_mode (globus_gass_transfer_requestattr_t * attr, globus_gass_transfer_file_mode_t * file_mode)`

Set/Get the file mode attribute for a GASS transfer attribute set.

This attribute allows the user to control whether the file will be transferred in ASCII or binary file mode.

**Parameters:**

*attr* The attribute set to query or modify.

*file\_mode* The value of the file mode attribute.

**Return values:**

**GLOBUS\_SUCCESS** The attribute was successfully updated.

**GLOBUS\_GASS\_TRANSFER\_ERROR\_NULL\_POINTER** The *attr* was GLOBUS\_NULL.

**4.7.2.14** `int globus_gass_transfer_requestattr_get_connection_reuse (globus_gass_transfer_requestattr_t * attr, globus_bool_t * connection_reuse)`

Set/Get the connection reuse attribute for a GASS transfer attribute set.

This attribute allows the user to control whether the connection associated with a GASS Transfer request should be reused after the file transfer has completed.

**Parameters:**

*attr* The attribute set to query or modify.

*connection\_reuse* The value of the connection reuse attribute.

**Return values:**

**GLOBUS\_SUCCESS** The attribute was successfully updated.

**GLOBUS\_GASS\_TRANSFER\_ERROR\_NULL\_POINTER** The *attr* was GLOBUS\_NULL.

**4.7.2.15** `int globus_gass_transfer_requestattr_get_socket_sndbuf (globus_gass_transfer_requestattr_t * attr, int * sndbuf)`

Set/Get the send buffer size attribute for a GASS transfer attribute set.

This attribute allows the user to control the socket send buffer associated with a GASS Transfer request should be reused after the file transfer has completed.

**Parameters:**

*attr* The attribute set to query or modify.

*sndbuf* The value of the socket buffer.

**Return values:**

**GLOBUS\_SUCCESS** The attribute was successfully updated.

**GLOBUS\_GASS\_TRANSFER\_ERROR\_NULL\_POINTER** The *attr* was GLOBUS\_NULL.

**4.7.2.16** `int globus_gass_transfer_requestattr_get_socket_rcvbuf (globus_gass_transfer_requestattr_t * attr, int * rcvbuf)`

Set/Get the receive buffer size attribute for a GASS transfer attribute set.

This attribute allows the user to control the socket receive buffer associated with a GASS Transfer request should be reused after the file transfer has completed.

**Parameters:**

*attr* The attribute set to query or modify.

*rcvbuf* The value of the socket buffer.

**Return values:**

**GLOBUS\_SUCCESS** The attribute was successfully updated.

**GLOBUS\_GASS\_TRANSFER\_ERROR\_NULL\_POINTER** The *attr* was GLOBUS\_NULL.

#### 4.7.2.17 **int globus\_gass\_transfer\_requestattr\_get\_socket\_nodelay (globus\_gass\_transfer\_requestattr\_t \* attr, globus\_bool\_t \* nodelay)**

Set/Get the TCP nodelay attribute for a GASS transfer attribute set.

This attribute allows the user to control the socket receive buffer associated with a GASS Transfer request should be reused after the file transfer has completed.

##### **Parameters:**

- attr* The attribute set to query or modify.
- nodelay* The value of the nodelay attribute.

##### **Return values:**

- GLOBAL\_SUCCESS** The attribute was successfully updated.
- GLOBAL\_GASS\_TRANSFER\_ERROR\_NULL\_POINTER** The *attr* was GLOBAL\_NULL.

#### 4.7.2.18 **int globus\_gass\_transfer\_secure\_requestattr\_get\_authorization (globus\_gass\_transfer\_requestattr\_t \* attr, globus\_gass\_transfer\_authorization\_t \* mode, char \*\* subject)**

Set/Get the authorization attribute for a GASS transfer attribute set.

This attribute allows the user to control what type of authorization should be done when GASS Transfer requests are processed.

##### **Parameters:**

- attr* The attribute set to query or modify.
- mode* The authorization mode to use.
- subject* The subject name of the authorized subject, if *mode* is GLOBAL\_GASS\_TRANSFER\_AUTHORIZE\_SUBJECT

##### **Return values:**

- GLOBAL\_SUCCESS** The attribute was successfully updated.
- GLOBAL\_GASS\_TRANSFER\_ERROR\_NULL\_POINTER** The *attr* was GLOBAL\_NULL.

## 4.8 Listener attributes

### Listener Backlog

- **int globus\_gass\_transfer\_listenerattr\_set\_backlog (globus\_gass\_transfer\_listenerattr\_t \*attr, int backlog)**
- **int globus\_gass\_transfer\_listenerattr\_get\_backlog (globus\_gass\_transfer\_listenerattr\_t \*attr, int \*backlog)**

### Listener Port

- **int globus\_gass\_transfer\_listenerattr\_set\_port (globus\_gass\_transfer\_listenerattr\_t \*attr, unsigned short port)**
- **int globus\_gass\_transfer\_listenerattr\_get\_port (globus\_gass\_transfer\_listenerattr\_t \*attr, unsigned short \*port)**

### Functions

- **int globus\_gass\_transfer\_listenerattr\_init (globus\_gass\_transfer\_listenerattr\_t \*attr, char \*url\_scheme)**

## 4.8.1 Function Documentation

### 4.8.1.1 `int globus_gass_transfer_listenerattr_init (globus_gass_transfer_listenerattr_t * attr, char * url_scheme)`

Initialize a listener attribute.

This function initializes the *attr* to contain a new protocol-specific listener attribute.

#### Parameters:

*attr* The attribute set to be initialized.

*url\_scheme* The scheme which the attribute will be used for.

#### Return values:

**GLOBAL\_SUCCESS** The attribute was successfully initialized.

**GLOBAL\_GASS\_TRANSFER\_ERROR\_NULL\_POINTER** Either *attr* or *url\_scheme* was GLOBAL\_NULL.

**GLOBAL\_GASS\_TRANSFER\_ERROR\_NOT\_IMPLEMENTED** No protocol module currently registered with GASS Transfer Library handles URLs with the specified *url\_scheme*.

### 4.8.1.2 `int globus_gass_transfer_listenerattr_set_backlog (globus_gass_transfer_listenerattr_t * attr, int backlog)`

Set/Get the backlog attribute for a GASS transfer attribute set.

This attribute allows the user to control then number of pending connections which may exist for this listener.

#### Parameters:

*attr* The attribute set to query or modify.

*backlog* The number of outstanding connections to allow.

#### Return values:

**GLOBAL\_SUCCESS** The attribute was successfully updated.

**GLOBAL\_GASS\_TRANSFER\_ERROR\_NULL\_POINTER** The *attr* was GLOBAL\_NULL.

### 4.8.1.3 `int globus_gass_transfer_listenerattr_set_port (globus_gass_transfer_listenerattr_t * attr, unsigned short port)`

Set/Get the port attribute for a GASS transfer attribute set.

This attribute allows the user to set the port to be used by a GASS Transfer listener.

#### Parameters:

*attr* The attribute set to query or modify.

*port* The TCP or UDP port number to use.

#### Return values:

**GLOBAL\_SUCCESS** The attribute was successfully updated.

**GLOBAL\_GASS\_TRANSFER\_ERROR\_NULL\_POINTER** The *attr* was GLOBAL\_NULL.

#### 4.8.1.4 `int globus_gass_transfer_listenerattr_get_backlog (globus_gass_transfer_listenerattr_t * attr, int * backlog)`

Set/Get the backlog attribute for a GASS transfer attribute set.

This attribute allows the user to control then number of pending connections which may exist for this listener.

##### Parameters:

*attr* The attribute set to query or modify.

*backlog* The number of outstanding connections to allow.

##### Return values:

**GLOBUS\_SUCCESS** The attribute was successfully updated.

**GLOBUS\_GASS\_TRANSFER\_ERROR\_NULL\_POINTER** The *attr* was GLOBUS\_NULL.

#### 4.8.1.5 `int globus_gass_transfer_listenerattr_get_port (globus_gass_transfer_listenerattr_t * attr, unsigned short * port)`

Set/Get the port attribute for a GASS transfer attribute set.

This attribute allows the user to set the port to be used by a GASS Transfer listener.

##### Parameters:

*attr* The attribute set to query or modify.

*port* The TCP or UDP port number to use.

##### Return values:

**GLOBUS\_SUCCESS** The attribute was successfully updated.

**GLOBUS\_GASS\_TRANSFER\_ERROR\_NULL\_POINTER** The *attr* was GLOBUS\_NULL.

## 4.9 Implementing Request Attributes

### Functions

- `globus_object_t * globus_gass_transfer_requestattr_initialize (globus_object_t *obj, char *proxy_url, globus_size_t block_size, globus_gass_transfer_file_mode_t file_mode, globus_bool_t connection_reuse)`
- `globus_object_t * globus_gass_transfer_socket_requestattr_initialize (globus_object_t *obj, char *proxy_url, globus_size_t block_size, globus_gass_transfer_file_mode_t file_mode, globus_bool_t connection_reuse, int sndbuf, int rcvbuf, globus_bool_t nodelay)`
- `globus_object_t * globus_gass_transfer_secure_requestattr_initialize (globus_object_t *obj, char *proxy_url, globus_size_t block_size, globus_gass_transfer_file_mode_t file_mode, globus_bool_t connection_reuse, int sndbuf, int rcvbuf, globus_bool_t nodelay, globus_gass_transfer_authorization_t authorization, char *subject)`
- `globus_object_t * globus_gass_transfer_listenerattr_initialize (globus_object_t *obj, int backlog, unsigned short port)`

### 4.9.1 Function Documentation

#### 4.9.1.1 `globus_object_t* globus_gass_transfer_requestattr_initialize (globus_object_t * obj, char * proxy_url, globus_size_t block_size, globus_gass_transfer_file_mode_t file_mode, globus_bool_t connection_reuse)`

Initialize a base request attribute.



**Parameters:**

*obj*  
*proxy\_url*  
*block\_size*  
*file\_mode*  
*connection\_reuse*

**Returns:**

Returns the *obj* pointer if the object inherited from the *GLOBUS\_GASS\_OBJECT\_TYPE\_REQUESTATTR* type and the attribute could be initialized; *GLOBUS\_NULL* otherwise.

**4.9.1.2** *globus\_object\_t\* globus\_gass\_transfer\_socket\_requestattr\_initialize (globus\_object\_t \* obj, char \* proxy\_url, globus\_size\_t block\_size, globus\_gass\_transfer\_file\_mode\_t file\_mode, globus\_bool\_t connection\_reuse, int sndbuf, int rcvbuf, globus\_bool\_t nodelay)*

Initialize a socket request attribute.

**Parameters:**

*obj*  
*proxy\_url*  
*block\_size*  
*file\_mode*  
*connection\_reuse*  
*sndbuf*  
*rcvbuf*  
*nodelay*

**Returns:**

Returns the *obj* pointer if the object inherited from the *GLOBUS\_GASS\_OBJECT\_TYPE\_SOCKET\_REQUESTATTR* type and the attribute could be initialized; *GLOBUS\_NULL* otherwise.

**4.9.1.3** *globus\_object\_t\* globus\_gass\_transfer\_secure\_requestattr\_initialize (globus\_object\_t \* obj, char \* proxy\_url, globus\_size\_t block\_size, globus\_gass\_transfer\_file\_mode\_t file\_mode, globus\_bool\_t connection\_reuse, int sndbuf, int rcvbuf, globus\_bool\_t nodelay, globus\_gass\_transfer\_authorization\_t authorization, char \* subject)*

Initialize a secure request attribute.

**Parameters:**

*obj*  
*proxy\_url*  
*block\_size*  
*file\_mode*  
*connection\_reuse*  
*sndbuf*  
*rcvbuf*  
*nodelay*  
*authorization*

*subject*

**Returns:**

Returns the *obj* pointer if the object inherited from the *GLOBUS\_GASS\_OBJECT\_TYPE\_SECURE\_REQUESTATTR* type and the attribute could be initialized; *GLOBUS\_NULL* otherwise.

**4.9.1.4 globus\_object\_t\* globus\_gass\_transfer\_listenerattr\_initialize (globus\_object\_t \* *obj*, int *backlog*, unsigned short *port*)**

Initialize a base listener attribute.

**Parameters:**

*obj*

*backlog*

*port*

**Returns:**

Returns the *obj* pointer if the object inherited from the *GLOBUS\_GASS\_OBJECT\_TYPE\_LISTENERATTR* type and the attribute could be initialized; *GLOBUS\_NULL* otherwise.

## 4.10 Protocol Modules

Protocol Implementation API.

**Data Structures**

- struct **globus\_gass\_transfer\_request\_proto\_s**  
*Protocol module request handling structure.*
- struct **globus\_gass\_transfer\_listener\_proto\_s**  
*Protocol module listener handling structure.*
- struct **globus\_gass\_transfer\_proto\_descriptor\_t**  
*Protocol module descriptor structure.*

**Typedefs**

- typedef **globus\_gass\_transfer\_request\_proto\_s** **globus\_gass\_transfer\_request\_proto\_t**
- typedef **globus\_gass\_transfer\_listener\_proto\_s** **globus\_gass\_transfer\_listener\_proto\_t**
- typedef void(\* **globus\_gass\_transfer\_proto\_send\_t** )(globus\_gass\_transfer\_request\_proto\_t \*proto, globus\_gass\_transfer\_request\_t request, globus\_byte\_t \*bytes, globus\_size\_t send\_length, globus\_bool\_t last\_data)
- typedef void(\* **globus\_gass\_transfer\_proto\_receive\_t** )(globus\_gass\_transfer\_request\_proto\_t \*proto, globus\_gass\_transfer\_request\_t request, globus\_byte\_t \*bytes, globus\_size\_t bytes\_length, globus\_size\_t wait\_for\_length)
- typedef void(\* **globus\_gass\_transfer\_proto\_func\_t** )(globus\_gass\_transfer\_request\_proto\_t \*proto, globus\_gass\_transfer\_request\_t request)
- typedef void(\* **globus\_gass\_transfer\_proto\_new\_request\_t** )(globus\_gass\_transfer\_request\_t request, globus\_gass\_transfer\_requestattr\_t \*attr)

- `typedef int(* globus_gass_transfer_proto_create_listener_t )(globus_gass_transfer_listener_t listener, globus_gass_transfer_listenerattr_t *attr, char *scheme, char **base_url, globus_gass_transfer_listener_proto_t **proto)`
- `typedef void(* globus_gass_transfer_proto_listener_t )(globus_gass_transfer_listener_proto_t *proto, globus_gass_transfer_listener_t listener)`
- `typedef globus_object_t *(* globus_gass_transfer_proto_new_attr_t )(char *url_scheme)`
- `typedef void(* globus_gass_transfer_proto_accept_t )(globus_gass_transfer_listener_proto_t *proto, globus_gass_transfer_listener_t listener, globus_gass_transfer_request_t request, globus_gass_transfer_requestattr_t *attr)`

## Functions

- `void globus_gass_transfer_proto_send_complete (globus_gass_transfer_request_t request, globus_byte_t *bytes, globus_size_t nbytes, globus_bool_t failed, globus_bool_t last_data)`
- `void globus_gass_transfer_proto_receive_complete (globus_gass_transfer_request_t request, globus_byte_t *bytes, globus_size_t nbytes, globus_bool_t failed, globus_bool_t last_data)`
- `void globus_gass_transfer_proto_listener_ready (globus_gass_transfer_listener_t listener)`
- `int globus_gass_transfer_proto_register_protocol (globus_gass_transfer_proto_descriptor_t *proto_desc)`
- `int globus_gass_transfer_proto_unregister_protocol (globus_gass_transfer_proto_descriptor_t *proto_desc)`
- `void globus_gass_transfer_proto_request_ready (globus_gass_transfer_request_t request, globus_gass_transfer_request_proto_t *proto)`
- `void globus_gass_transfer_proto_new_listener_request (globus_gass_transfer_listener_t listener, globus_gass_transfer_request_t request, globus_gass_transfer_request_proto_t *proto)`
- `void globus_gass_transfer_proto_request_denied (globus_gass_transfer_request_t request, int reason, char *message)`
- `void globus_gass_transfer_proto_request_referred (globus_gass_transfer_request_t request, char **url, globus_size_t num_urls)`

### 4.10.1 Detailed Description

#### Protocol Implementation API.

The GASS Protocol Module API is designed to make it possible to extend the GASS client and server APIs to support additional protocols without making any changes to the core of the GASS implementation. GASS protocol modules are intended to handle protocol-specific connection and data handling. The GASS Transfer library includes protocol modules which implement the HTTP, HTTPS, FTP, and GSI-FTP protocols.

Every protocol module implementation must include the following header file:

```
#include "globus_gass_transfer_proto.h"
```

To implement a protocol module, one must create a **globus\_gass\_transfer\_proto\_descriptor\_t**(p. 42) structure which indicates what the protocol module is able to do. This structure contains the URL scheme which the protocol module supports, and function pointers which indicate what type of operations (client or server) that the module implements. To implement a client-side protocol module, the `new_requestattr` and `new_request` fields must be set to the protocol module's implementations of those functions. To implement a server-side protocol module, the `new_listenerattr` and `new_listener` functions must be set to the protocol module's implementations of those functions.

A protocol module implementor registers a protocol module with the GASS Transfer library by calling the function **globus\_gass\_transfer\_proto\_register\_protocol**(p. 39), and unregisters the module by calling **globus\_gass\_transfer\_proto\_unregister\_protocol**(p. 40). This functions must be called after the **GLOBUS\_GASS\_TRANSFER\_MODULE**(p. 2) has already been activated. Once registered, applications may use URLs of the scheme type provided by the protocol module for the standard **client**(p. 2) or **server**(p. 7) operations.

## 4.10.2 Typedef Documentation

### 4.10.2.1 typedef struct globus\_gass\_transfer\_request\_proto\_s globus\_gass\_transfer\_request\_proto\_t

Protocol module request handling structure.

See also:

`globus_gass_transfer_request_proto_s`(p. 44)

### 4.10.2.2 typedef struct globus\_gass\_transfer\_listener\_proto\_s globus\_gass\_transfer\_listener\_proto\_t

Protocol module listener handling structure.

See also:

`globus_gass_transfer_listener_proto_s`(p. 41)

### 4.10.2.3 typedef void(\* globus\_gass\_transfer\_proto\_send\_t)(globus\_gass\_transfer\_request\_proto\_t \*proto, globus\_gass\_transfer\_request\_t request, globus\_byte\_t \*bytes, globus\_size\_t send\_length, globus\_bool\_t last\_data)

Protocol module function type to handle sending data.

A function pointer of this type is associated with the `globus_gass_transfer_request_proto_t`(p. 35) associated with a request handle. It is called when client or server has registered a bytes array for sending to the client or server which is handling the request. The GASS Transfer Library will only pass one *bytes* array to the protocol module for processing per request at any given time.

Once the protocol module has processed the array, it must call `globus_gass_transfer_proto_send_complete()`(p. 38) to let the GASS Transfer library continue to process the request.

#### Parameters:

*proto* The protocol module's request handler.

*request* The request handle with which this block of bytes is associated.

*bytes* The user-supplied byte array containing the data associated with the request.

*bytes\_length* The length of the *bytes* array.

*last\_data* A flag to indicate whether this is the final block of data for the request. If this is true, then the *callback* function will be delayed until the server acknowledges that the file has been completely received.

See also:

`globus_gass_transfer_send_bytes()`(p. 13)

### 4.10.2.4 typedef void(\* globus\_gass\_transfer\_proto\_receive\_t)(globus\_gass\_transfer\_request\_proto\_t \*proto, globus\_gass\_transfer\_request\_t request, globus\_byte\_t \*bytes, globus\_size\_t bytes\_length, globus\_size\_t wait\_for\_length)

Protocol module function type to handle receiving data.

A function pointer of this type is associated with the `globus_gass_transfer_request_proto_t`(p. 35) associated with a request handle. It is called when client or server has registered a bytes array for receiving from the client or server which is handling the request. The GASS Transfer Library will only pass one *bytes* array to the protocol module for processing per request at any given time.

Once the protocol module has processed the array, it must call `globus_gass_transfer_proto_receive_complete()`(p. 39) to let the GASS Transfer library continue to process the request.

**Parameters:**

*proto* The protocol module's request handler.

*request* The request handle with which this block of bytes is associated.

*bytes* The user-supplied byte array containing the data associated with the request.

*bytes\_length* The length of the *bytes* array.

*wait\_for\_length* The minimum amount of data to receive before calling **globus\_gass\_transfer\_proto\_receive\_complete()**(p. 39) for the request. The GASS Transfer protocol module may call that function with a smaller value for the amount received if EOF has been reached.

**See also:**

**globus\_gass\_transfer\_receive\_bytes()**(p. 14)

**4.10.2.5 typedef void(\* globus\_gass\_transfer\_proto\_func\_t)(globus\_gass\_transfer\_request\_proto\_t \*proto, globus\_gass\_transfer\_request\_t request)**

Protocol module implementation function type.

Function pointers of this type are associated with the **globus\_gass\_transfer\_request\_proto\_t**(p. 35) associated with a particular request handle. They are called when certain functions which modify the status of a request have been called by a client or server.

A function of this type is used for the fail, deny, refer, authorize, and destroy fields of the **globus\_gass\_transfer\_request\_proto\_t**(p. 35). A protocol module can query the request handle to determine the status and, if applicable, denial reasons if necessary.

**Parameters:**

*proto* The protocol module's request handler.

*request* The request handle.

**4.10.2.6 typedef void(\* globus\_gass\_transfer\_proto\_new\_request\_t)(globus\_gass\_transfer\_request\_t request, globus\_gass\_transfer\_requestattr\_t \*attr)**

Protocol module implementation function type for new client requests.

A function pointer of this type is associated with the **globus\_gass\_transfer\_proto\_descriptor\_t**(p. 42) for a particular protocol module's implementation. It is called when the client has begun a file transfer request by calling one of the functions in the "[ref globus\\_gass\\_transfer\\_client](#)" section of this manual.

When this function is called for a protocol module, the module should query the request handle to determine the URL which is being requested by the client, and the operation being done on that URL. The protocol module should initiate the request, and once it has determined that it has been authorized, denied, or referred, one of **globus\_gass\_transfer\_proto\_request\_ready()**(p. 40), **globus\_gass\_transfer\_proto\_request\_denied()**(p. 41), or **globus\_gass\_transfer\_proto\_request\_referred()**(p. 41) must be called.

**Parameters:**

*request* The request handle containing the information about the request.

*attr* A protocol-specific attribute set, created by calling the protocol module's **new\_requestattr** function pointer(p. 43).

**4.10.2.7 typedef int(\* globus\_gass\_transfer\_proto\_create\_listener\_t)(globus\_gass\_transfer\_listener\_t listener, globus\_gass\_transfer\_listenerattr\_t \*attr, char \*scheme, char \*\*base\_url, globus\_gass\_transfer\_listener\_proto\_t \*\*proto)**

Protocol module implementation function type for new server listeners.

A function pointer of this type is associated with the **globus\_gass\_transfer\_proto\_descriptor\_t**(p. 42) for a particular protocol module's implementation. It is called when the server has called **globus\_gass\_transfer\_create\_listener**(p. 8).

**Parameters:**

*listener* The listener handle to associate with the new *proto* created by the protocol module.

*attr* A protocol-specific attribute set, created by calling the protocol module's **new\_listenerattr function pointer**(p. 43).

*scheme* The URL scheme that the server has requested for the new listener. This will be one the scheme associated with a particular protocol module.

*base\_url* A pointer to be set the value of the base url of this listener. For most protocols, this will contain the scheme, hostname, and port number of the listener. This string must be allocated using one of the memory allocators defined in the globus\_common library. It will be freed by the GASS Transfer library when the listener is closed.

*proto* A pointer to be set to a new **globus\_gass\_transfer\_listener\_proto\_t** which will be associated with this listener. This must be allocated by the protocol module using one of the memory allocators defined in the globus\_common library. It will be freed by the GASS Transfer library when the listener is closed.

**Returns:**

A GASS error value, or GLOBUS\_SUCCESS.

**4.10.2.8 typedef void(\* globus\_gass\_transfer\_proto\_listener\_t)(globus\_gass\_transfer\_listener\_proto\_t \*proto, globus\_gass\_transfer\_listener\_t listener)**

Protocol module implementation function type for server operations.

Function pointers of this type are associated with the **globus\_gass\_transfer\_listener\_proto\_t**(p. 35) associated with a particular listener handle. They are called when a server implementation wants to close the listener, listen for new connections, or destroy the listener.

**Parameters:**

*proto* The protocol-specific implementation of the **globus\_gass\_transfer\_listener\_proto\_t**(p. 35) for a particular listener.

*listener* The listener handle associated with the *proto*.

**See also:**

**globus\_gass\_transfer\_proto\_create\_listener\_t**(p. 36)

**4.10.2.9 typedef globus\_object\_t(\* globus\_gass\_transfer\_proto\_new\_attr\_t)(char \*url\_scheme)**

Protocol module implementation function type for attribute creation.

A function pointer of this type is associated with the **globus\_gass\_transfer\_proto\_descriptor\_t**(p. 42) defining a protocol module. It is called when a client requests a new request attribute set be created for a URL scheme handled by a protocol module. The function implementation must create a new request attribute usable by the protocol.

The returned attribute must be a globus object which inherits from one of the base attributes defined in the GASS Transfer API. A client or server operation will use a request attribute generated by this function when creating a new **globus\_gass\_transfer\_request\_proto\_t**(p. 35) to handle a request.

**Parameters:**

*url\_scheme* The URL scheme that the request attribute should be compatible with.

**Returns:**

A `globus_object_t`-based request attribute.

**See also:**

`globus_gass_transfer_proto_new_request_t`(p. 36), `globus_gass_transfer_proto_accept_t`(p. 38)

**4.10.2.10** `typedef void(* globus_gass_transfer_proto_accept_t)(globus_gass_transfer_listener_proto_t *proto, globus_gass_transfer_listener_t listener, globus_gass_transfer_request_t request, globus_gass_transfer_requestattr_t *attr)`

Protocol module implementation function type for server request parsing.

Function pointers of this type are associated with the `globus_gass_transfer_listener_proto_t`(p. 35) associated with a particular listener handle. They are called when a server implementation wants to accept a new connection from the listener. A new request is generated based on the protocol-specific request done on the new connection.

The new request will be created with the attributes specified in the *attr* parameter. Once the protocol module has parsed the request, it must call `globus_gass_transfer_proto_new_listener_request()`(p. 40) to let the server implementation decide how to process this request.

The protocol module should update the *request* to indicate the type of operation being requested, the size of the file (if applicable), and the identity of the client (if applicable).

**Parameters:**

*proto* The protocol specific listener data structure associated with the listener handle.

*listener* The listener handle which the user requested the listen on.

*request* The new request handle.

*attr* The request attribute set to be used when processing this request.

**4.10.3 Function Documentation**

**4.10.3.1** `void globus_gass_transfer_proto_send_complete (globus_gass_transfer_request_t request, globus_byte_t *bytes, globus_size_t nbytes, globus_bool_t failed, globus_bool_t last_data)`

Data send complete.

A protocol module must call this function once a byte range registered for sending via the protocol module's `send_buffer` method has been completely processed. This function is called for protocol modules implementing either server or client functionality.

**Parameters:**

*request* The request handle associated with this byte array.

*bytes* The byte array which was sent. This should be the same as the pointer passed in the `send_buffer` method.

*nbytes* The number of bytes which were sent from this byte array. This may be different than length passed to the `send_buffer` method if an error occurred.

*failed* A boolean indicating whether this byte range was successfully sent or not. This should be set to `GLOBUS_TRUE` if either a protocol error or a user-generated abort has occurred while processing the byte range. If this is set to `GLOBUS_TRUE`, then the *last\_data* parameter must also be set to `GLOBUS_TRUE`.

*last\_data* A boolean indicating whether this byte range was the final one which can be processed for this request. This should be set to `GLOBUS_TRUE` if an error occurred while processing this byte range, and user-generated abort occurred, or this is the final byte range in the data transfer.

**See also:**

`globus_gass_transfer_send_bytes()`(p. 13)

#### 4.10.3.2 void globus\_gass\_transfer\_proto\_receive\_complete (globus\_gass\_transfer\_request\_t request, globus\_byte\_t \* bytes, globus\_size\_t nbytes, globus\_bool\_t failed, globus\_bool\_t last\_data)

Data receive complete.

A protocol module must call this function once a byte range registered for receive via the protocol module's `recv_-buffer` method has been completely processed. This function is called for protocol modules implementing either server or client functionality.

##### Parameters:

**request** The request handle associated with this byte array.

**bytes** The byte array which was received into. This should be the same as the pointer passed in the `send_-buffer` method.

**nbytes** The number of bytes which were sent from this byte array. This may be different than length passed to the `send_buffer` method if an error occurred or EOF was reached while receiving the data.

**failed** A boolean indicating whether this byte range was successfully received or not. This should be set to `GLOBUS_TRUE` if either a protocol error or a user-generated abort has occurred while processing the byte range. If this is set to `GLOBUS_TRUE`, then the `last_data` parameter must also be set to `GLOBUS_TRUE`.

**last\_data** A boolean indicating whether this byte range was the final one which can be processed for this request. This should be set to `GLOBUS_TRUE` if an error occurred while processing this byte range, and user-generated abort occurred, or this is the final byte range in the data transfer.

##### See also:

`globus_gass_transfer_send_bytes()`(p. 13)

#### 4.10.3.3 void globus\_gass\_transfer\_proto\_listener\_ready (globus\_gass\_transfer\_listener\_t listener)

Server listener ready.

This function notifies the GASS Transfer Library that the protocol module has decided that a new request can be accepted on this particular listener. It must only be called after the GASS Transfer Library has called the **listen function**(p. 42) in a `globus_gass_transfer_listener_proto_t`(p. 35) protocol module-specific listener structure.

##### Parameters:

**listener** The listener handle which is now ready for accepting a new connection.

#### 4.10.3.4 int globus\_gass\_transfer\_proto\_register\_protocol (globus\_gass\_transfer\_proto\_descriptor\_t \* proto\_desc)

Register protocol.

This function registers a protocol module handler with the GASS Transfer library. If this succeeds, then users of the library may use the URLs with the protocol scheme implemented by this module descriptor in GASS Transfer operations.

##### Parameters:

**proto\_desc** The protocol module descriptor. See the "**Protocol Modules**(p. 33)" section of the manual for information on this structure.

##### Return values:

**GLOBUS\_SUCCESS** The protocol module was successfully registered with GASS.

**GLOBUS\_GASS\_TRANSFER\_ERROR\_NULL\_POINTER** The `proto_desc` parameter was `GLOBUS_NULL`.

**GLOBUS\_GASS\_TRANSFER\_ERROR\_ALREADY\_REGISTERED** A protocol module has already been registered with GASS to handle this URL scheme.



#### 4.10.3.5 `int globus_gass_transfer_proto_unregister_protocol (globus_gass_transfer_proto_descriptor_t * proto_desc)`

Unregister protocol.

This function unregisters a protocol module handler from the GASS Transfer library. If this succeeds, then users of the library may no longer use URLs with the protocol scheme implemented by this module descriptor in GASS Transfer operations.

##### Parameters:

*proto\_desc* The protocol module descriptor. See the "**Protocol Modules**(p. 33)" section of the manual for information on this structure.

##### Return values:

**GLOBAL\_SUCCESS** The protocol module was successfully registered with GASS.

**GLOBAL\_GASS\_TRANSFER\_ERROR\_NULL\_POINTER** The *proto\_desc* parameter was **GLOBAL\_NULL**.

**GLOBAL\_GASS\_TRANSFER\_ERROR\_INVALID\_USE** A protocol module has not been registered with GASS to handle this URL scheme.

#### 4.10.3.6 `void globus_gass_transfer_proto_request_ready (globus_gass_transfer_request_t request, globus_gass_transfer_request_proto_t * proto)`

Request ready.

This function notifies the GASS Transfer Library that new request generated by a client has begun processing in a protocol module, and that protocol module is now ready to send or receive data to handle this request.

##### Parameters:

*request* The request handle used for this request. This was created by the user calling one of the functions in the "**Client-Initiated Operations**(p. 2)" section of this manual.

*proto* The protocol-module specific request structure. This structure contains a set of function pointers to allow GASS to continue to process this request.

##### See also:

`globus_gass_transfer_proto_request_referred()`(p. 41), `globus_gass_transfer_proto_request_denied()`(p. 41)

#### 4.10.3.7 `void globus_gass_transfer_proto_new_listener_request (globus_gass_transfer_listener_t listener, globus_gass_transfer_request_t request, globus_gass_transfer_request_proto_t * proto)`

New listener request.

This function notifies the GASS Transfer Library that new request generated by a server calling the **globus\_gass\_transfer\_register\_accept()**(p. 10) function has begun processing in a protocol module, and that protocol module is now ready to send or receive data to handle this request.

##### Parameters:

*listener* The listener handle used to accept this request.

*request* The request handle used for this request. This was created by the user calling one of the functions in the "**Client-Initiated Operations**(p. 2)" section of this manual.

*proto* The protocol-module specific request structure. This structure contains a set of function pointers to allow GASS to continue to process this request.

#### 4.10.3.8 void globus\_gass\_transfer\_proto\_request\_denied (globus\_gass\_transfer\_request\_t request, int reason, char \* message)

Request denied.

This function notifies the GASS Transfer Library that new request generated by a client calling one of the functions in the "@ref globus\_gass\_transfer\_client" section of the manual has been denied by the server, and so cannot be processed by the protocol module.

##### Parameters:

*request* The request handle used for this request. This was created by the user calling one of the functions in the "**Client-Initiated Operations**(p. 2)" section of this manual.

*reason* A protocol-specific reason code.

*message* A string containing a message describing why the request was denied. The GASS Transfer library is responsible for freeing this message. It must be allocated using one of the memory allocators defined in the Globus Common Library.

##### See also:

globus\_gass\_transfer\_proto\_request\_ready()(p. 40), globus\_gass\_transfer\_proto\_request\_referred()(p. 41)

#### 4.10.3.9 void globus\_gass\_transfer\_proto\_request\_referred (globus\_gass\_transfer\_request\_t request, char \*\* url, globus\_size\_t num\_urls)

Request referred.

This function notifies the GASS Transfer Library that new request generated by a client calling one of the functions in the "@ref globus\_gass\_transfer\_client" section of the manual has been referred to another URL by the server, and so processing has stopped.

##### Parameters:

*request* The request handle used for this request. This was created by the user calling one of the functions in the "**Client-Initiated Operations**(p. 2)" section of this manual.

*url* An array of url strings containing alternate locations for this file. The GASS transfer library is responsible for freeing this array. It must be allocated using one of the memory allocators defined in the Globus Common Library.

*num\_urls* The length of the *url* array.

##### See also:

globus\_gass\_transfer\_proto\_request\_ready()(p. 40), globus\_gass\_transfer\_proto\_request\_denied()(p. 41), globus\_gass\_transfer\_proto\_request\_referred()(p. 41)

## 5 globus gass transfer Data Structure Documentation

### 5.1 globus\_gass\_transfer\_listener\_proto\_s Struct Reference

Protocol module listener handling structure.

#### Data Fields

- globus\_gass\_transfer\_proto\_listener\_t close\_listener
- globus\_gass\_transfer\_proto\_listener\_t listen
- globus\_gass\_transfer\_proto\_accept\_t accept
- globus\_gass\_transfer\_proto\_listener\_t destroy

### 5.1.1 Detailed Description

Protocol module listener handling structure.

### 5.1.2 Field Documentation

#### 5.1.2.1 `globus_gass_transfer_proto_listener_t globus_gass_transfer_listener_proto_s::close_listener`

Close listener.

#### 5.1.2.2 `globus_gass_transfer_proto_listener_t globus_gass_transfer_listener_proto_s::listen`

Listen.

#### 5.1.2.3 `globus_gass_transfer_proto_accept_t globus_gass_transfer_listener_proto_s::accept`

Accept.

#### 5.1.2.4 `globus_gass_transfer_proto_listener_t globus_gass_transfer_listener_proto_s::destroy`

Destroy.

## 5.2 `globus_gass_transfer_proto_descriptor_t` Struct Reference

Protocol module descriptor structure.

### Data Fields

- `char * url_scheme`
- `globus_gass_transfer_proto_new_attr_t new_requestattr`
- `globus_gass_transfer_proto_new_request_t new_request`
- `globus_gass_transfer_proto_new_attr_t new_listenerattr`
- `globus_gass_transfer_proto_create_listener_t new_listener`

### 5.2.1 Detailed Description

Protocol module descriptor structure.

See also:

`globus_gass_transfer_proto_register_protocol()`(p. 39), `globus_gass_transfer_proto_unregister_protocol()`(p. 40)

### 5.2.2 Field Documentation

#### 5.2.2.1 `char* globus_gass_transfer_proto_descriptor_t::url_scheme`

URL Scheme.

The URL scheme which this protocol module supports. The scheme is the first part of a URL, which names the protocol which is used to access the resource named by the URL, for example "http" or "ftp".

The GASS Transfer library allows only one protocol module to be registered to handle a particular *url\_scheme*. However, a protocol module may implement only the client or only the server part of the protocol. If a protocol

has several variations with different scheme names (for example http and https), each scheme must be registered with GASS in order to be used.

#### 5.2.2.2 `globus_gass_transfer_proto_new_attr_t` `globus_gass_transfer_proto_descriptor_t::new_requestattr`

New request attributes.

The function pointed to by this pointer is used by GASS to forward requests to create a request attribute for this protocol's *url\_scheme* to the protocol module. The function returns a request attribute which inherits from one of the GASS Transfer request attributes.

See also:

`globus_gass_transfer_proto_new_attr_t`(p. 37)

#### 5.2.2.3 `globus_gass_transfer_proto_new_request_t` `globus_gass_transfer_proto_descriptor_t::new_request`

New request.

The function pointed to by this pointer is used by GASS to initiate a new file transfer request by a protocol module. The request handle has been initialized with the parameters passed to one of the functions in the **Client-Initiated Operations**(p. 2) section of the GASS Transfer API.

The protocol module should begin processing this request by sending appropriate messages to the file server. Once the request is authorized, denied, or referred, the protocol module calls `globus_gass_transfer_proto_request_ready()`(p. 40), `globus_gass_transfer_proto_request_denied()`(p. 41), or `globus_gass_transfer_proto_request_referred()`(p. 41).

See also:

`globus_gass_transfer_proto_new_request_t`(p. 36)

#### 5.2.2.4 `globus_gass_transfer_proto_new_attr_t` `globus_gass_transfer_proto_descriptor_t::new_listenerattr`

New listener attributes.

The function pointed to by this pointer is used by GASS to forward requests to create a listener attribute for this protocol's *url\_scheme* to the protocol module. The function returns a listener attribute which inherits from one of the GASS Transfer request attributes.

See also:

`globus_gass_transfer_proto_new_attr_t`(p. 37)

#### 5.2.2.5 `globus_gass_transfer_proto_create_listener_t` `globus_gass_transfer_proto_descriptor_t::new_listener`

New listener.

The function pointed to by this pointer is used by GASS to create a new listener handle. The listener handle has been initialized with the parameters passed to one of the functions in the **Client-Initiated Operations**(p. 2) section of the GASS Transfer API.

The protocol module should begin processing this request by sending appropriate messages to the file server. Once the request is authorized, denied, or referred, the protocol module calls `globus_gass_transfer_proto_request_ready()`(p. 40), `globus_gass_transfer_proto_request_denied()`(p. 41), or `globus_gass_transfer_proto_request_referred()`(p. 41).

See also:

`globus_gass_transfer_proto_new_request_t`(p. 36)

## 5.3 globus\_gass\_transfer\_request\_proto\_s Struct Reference

Protocol module request handling structure.

### Data Fields

- `globus_gass_transfer_proto_send_t` `send_buffer`
- `globus_gass_transfer_proto_receive_t` `recv_buffer`
- `globus_gass_transfer_proto_func_t` `fail`
- `globus_gass_transfer_proto_func_t` `deny`
- `globus_gass_transfer_proto_func_t` `refer`
- `globus_gass_transfer_proto_func_t` `authorize`
- `globus_gass_transfer_proto_func_t` `destroy`

### 5.3.1 Detailed Description

Protocol module request handling structure.

This structure is created by a GASS transfer protocol module to handle a particular request. It is created in response to a **listener's accept method**(p. 42) or a **protocol module's new\_request method**(p. 43).

Memory management of this structure is the responsibility of the protocol module. The destroy method will be called when the GASS Transfer library is finished dealing with it.

A protocol module may create an extension to this structure to contain protocol-specific information, as long as the first fields of the structure match this type.

See also:

`globus_gass_transfer_proto_request_ready()`(p. 40)

### 5.3.2 Field Documentation

#### 5.3.2.1 `globus_gass_transfer_proto_send_t` `globus_gass_transfer_request_proto_s::send_buffer`

Send bytes.

See also:

`globus_gass_transfer_proto_send_t`(p. 35)

#### 5.3.2.2 `globus_gass_transfer_proto_receive_t` `globus_gass_transfer_request_proto_s::recv_buffer`

Receive bytes.

See also:

`globus_gass_transfer_proto_recv_t`

#### 5.3.2.3 `globus_gass_transfer_proto_func_t` `globus_gass_transfer_request_proto_s::fail`

Fail a request.

This function is called when the application calls `globus_gass_transfer_fail()` on a request.

#### **5.3.2.4 globus\_gass\_transfer\_proto\_func\_t globus\_gass\_transfer\_request\_proto\_s::deny**

Deny a request.

#### **5.3.2.5 globus\_gass\_transfer\_proto\_func\_t globus\_gass\_transfer\_request\_proto\_s::refer**

Refer a request.

#### **5.3.2.6 globus\_gass\_transfer\_proto\_func\_t globus\_gass\_transfer\_request\_proto\_s::authorize**

Authorize a request.

#### **5.3.2.7 globus\_gass\_transfer\_proto\_func\_t globus\_gass\_transfer\_request\_proto\_s::destroy**

Destroy a request.

### **5.4 globus\_gass\_transfer\_request\_t Struct Reference**

#### **5.4.1 Detailed Description**

Request handle.

A request handle is associated with each file transfer operation. The same structure is used for both client- and server- side requests. For client operations, the initial call to **globus\_gass\_transfer\_get()**(p. 4), **globus\_gass\_transfer\_register\_get()**(p. 3), **globus\_gass\_transfer\_get()**(p. 4), **globus\_gass\_transfer\_register\_put()**(p. 4), **globus\_gass\_transfer\_append()**(p. 6), **globus\_gass\_transfer\_register\_append()**(p. 6) initializes the request. For server operations, the request is initialized by calling **globus\_gass\_transfer\_accept()**.

The functions in the **request section**(p. 16) of this manual describe the functions available for accessing information from a request handle.

Each request handle should be destroyed by calling **globus\_gass\_transfer\_request\_destroy()**(p. 22) once the user has completed processing the request.

## Index

- accept
  - globus\_gass\_transfer\_listener\_proto\_s, 42
- Activation, 2
- authorize
  - globus\_gass\_transfer\_request\_proto\_s, 45
- Client-Initiated Operations, 2
- close\_listener
  - globus\_gass\_transfer\_listener\_proto\_s, 42
- deny
  - globus\_gass\_transfer\_request\_proto\_s, 44
- destroy
  - globus\_gass\_transfer\_listener\_proto\_s, 42
  - globus\_gass\_transfer\_request\_proto\_s, 45
- fail
  - globus\_gass\_transfer\_request\_proto\_s, 44
- globus\_gass\_transfer\_activation
  - GLOBUS\_GASS\_TRANSFER\_MODULE, 2
- globus\_gass\_transfer\_append
  - globus\_gass\_transfer\_client, 6
- globus\_gass\_transfer\_authorize
  - globus\_gass\_transfer\_server, 12
- globus\_gass\_transfer\_bytes\_callback\_t
  - globus\_gass\_transfer\_data, 13
- globus\_gass\_transfer\_client
  - globus\_gass\_transfer\_append, 6
  - globus\_gass\_transfer\_get, 4
  - globus\_gass\_transfer\_put, 5
  - globus\_gass\_transfer\_register\_append, 6
  - globus\_gass\_transfer\_register\_get, 3
  - globus\_gass\_transfer\_register\_put, 4
- globus\_gass\_transfer\_close\_callback\_t
  - globus\_gass\_transfer\_server, 8
- globus\_gass\_transfer\_close\_listener
  - globus\_gass\_transfer\_server, 9
- globus\_gass\_transfer\_create\_listener
  - globus\_gass\_transfer\_server, 8
- globus\_gass\_transfer\_data
  - globus\_gass\_transfer\_bytes\_callback\_t, 13
  - globus\_gass\_transfer\_receive\_bytes, 14
  - globus\_gass\_transfer\_send\_bytes, 13
- globus\_gass\_transfer\_deny
  - globus\_gass\_transfer\_server, 12
- globus\_gass\_transfer\_get
  - globus\_gass\_transfer\_client, 4
- globus\_gass\_transfer\_listen\_callback\_t
  - globus\_gass\_transfer\_server, 8
- globus\_gass\_transfer\_listener\_get\_base\_url
  - globus\_gass\_transfer\_server, 11
- globus\_gass\_transfer\_listener\_get\_user\_pointer
  - globus\_gass\_transfer\_server, 10
- globus\_gass\_transfer\_listener\_proto\_s, 41
  - accept, 42
  - close\_listener, 42
  - destroy, 42
  - listen, 42
- globus\_gass\_transfer\_listener\_proto\_t
  - globus\_gass\_transfer\_protocol, 35
- globus\_gass\_transfer\_listener\_set\_user\_pointer
  - globus\_gass\_transfer\_server, 11
- globus\_gass\_transfer\_listenerattr
  - globus\_gass\_transfer\_listenerattr\_get\_backlog, 30
  - globus\_gass\_transfer\_listenerattr\_get\_port, 31
  - globus\_gass\_transfer\_listenerattr\_init, 30
  - globus\_gass\_transfer\_listenerattr\_set\_backlog, 30
  - globus\_gass\_transfer\_listenerattr\_set\_port, 30
- globus\_gass\_transfer\_listenerattr\_get\_backlog
  - globus\_gass\_transfer\_listenerattr, 30
- globus\_gass\_transfer\_listenerattr\_get\_port
  - globus\_gass\_transfer\_listenerattr, 31
- globus\_gass\_transfer\_listenerattr\_init
  - globus\_gass\_transfer\_listenerattr, 30
- globus\_gass\_transfer\_listenerattr\_initialize
  - globus\_gass\_transfer\_requestattr\_implementation, 33
- globus\_gass\_transfer\_listenerattr\_set\_backlog
  - globus\_gass\_transfer\_listenerattr, 30
- globus\_gass\_transfer\_listenerattr\_set\_port
  - globus\_gass\_transfer\_listenerattr, 30
- GLOBUS\_GASS\_TRANSFER\_MODULE
  - globus\_gass\_transfer\_activation, 2
- globus\_gass\_transfer\_proto\_accept\_t
  - globus\_gass\_transfer\_protocol, 38
- globus\_gass\_transfer\_proto\_create\_listener\_t
  - globus\_gass\_transfer\_protocol, 36
- globus\_gass\_transfer\_proto\_descriptor\_t, 42
  - new\_listener, 43
  - new\_listenerattr, 43
  - new\_request, 43
  - new\_requestattr, 43
  - url\_scheme, 42
- globus\_gass\_transfer\_proto\_func\_t
  - globus\_gass\_transfer\_protocol, 36
- globus\_gass\_transfer\_proto\_listener\_ready
  - globus\_gass\_transfer\_protocol, 39
- globus\_gass\_transfer\_proto\_listener\_t
  - globus\_gass\_transfer\_protocol, 37
- globus\_gass\_transfer\_proto\_new\_attr\_t
  - globus\_gass\_transfer\_protocol, 37
- globus\_gass\_transfer\_proto\_new\_listener\_request
  - globus\_gass\_transfer\_protocol, 40

globus_gass_transfer_proto_new_request_t	globus_gass_transfer_referral_get_count, 15
globus_gass_transfer_protocol, 36	globus_gass_transfer_referral_get_url, 15
globus_gass_transfer_proto_receive_complete	globus_gass_transfer_referral_destroy
globus_gass_transfer_protocol, 38	globus_gass_transfer_referral, 16
globus_gass_transfer_proto_receive_t	globus_gass_transfer_referral_get_count
globus_gass_transfer_protocol, 35	globus_gass_transfer_referral, 15
globus_gass_transfer_proto_register_protocol	globus_gass_transfer_referral_get_url
globus_gass_transfer_protocol, 39	globus_gass_transfer_referral, 15
globus_gass_transfer_proto_request_denied	globus_gass_transfer_register_accept
globus_gass_transfer_protocol, 40	globus_gass_transfer_server, 10
globus_gass_transfer_proto_request_ready	globus_gass_transfer_register_append
globus_gass_transfer_protocol, 40	globus_gass_transfer_client, 6
globus_gass_transfer_proto_request_referred	globus_gass_transfer_register_get
globus_gass_transfer_protocol, 41	globus_gass_transfer_client, 3
globus_gass_transfer_proto_send_complete	globus_gass_transfer_register_listen
globus_gass_transfer_protocol, 38	globus_gass_transfer_server, 9
globus_gass_transfer_proto_send_t	globus_gass_transfer_register_put
globus_gass_transfer_protocol, 35	globus_gass_transfer_client, 4
globus_gass_transfer_proto_unregister_protocol	globus_gass_transfer_request
globus_gass_transfer_protocol, 39	GLOBAL_GASS_TRANSFER_REQUEST_-
globus_gass_transfer_protocol	DENIED, 18
globus_gass_transfer_listener_proto_t, 35	globus_gass_transfer_request_destroy, 22
globus_gass_transfer_proto_accept_t, 38	GLOBAL_GASS_TRANSFER_REQUEST_-
globus_gass_transfer_proto_create_listener_t,	DONE, 18
36	GLOBAL_GASS_TRANSFER_REQUEST_-
globus_gass_transfer_proto_func_t, 36	FAILED, 18
globus_gass_transfer_proto_listener_ready, 39	globus_gass_transfer_request_get_denial_-
globus_gass_transfer_proto_listener_t, 37	message, 21
globus_gass_transfer_proto_new_attr_t, 37	globus_gass_transfer_request_get_denial_-
globus_gass_transfer_proto_new_listener_-	reason, 21
request, 40	globus_gass_transfer_request_get_length, 19
globus_gass_transfer_proto_new_request_t, 36	globus_gass_transfer_request_get_referral, 19
globus_gass_transfer_proto_receive_complete,	globus_gass_transfer_request_get_status, 19
38	globus_gass_transfer_request_get_subject, 21
globus_gass_transfer_proto_receive_t, 35	globus_gass_transfer_request_get_type, 18
globus_gass_transfer_proto_register_protocol,	globus_gass_transfer_request_get_url, 19
39	globus_gass_transfer_request_get_user_-
globus_gass_transfer_proto_request_denied,	pointer, 18
40	GLOBAL_GASS_TRANSFER_REQUEST_-
globus_gass_transfer_proto_request_ready, 40	INVALID, 18
globus_gass_transfer_proto_request_referred,	GLOBAL_GASS_TRANSFER_REQUEST_-
41	PENDING, 18
globus_gass_transfer_proto_send_complete,	GLOBAL_GASS_TRANSFER_REQUEST_-
38	REFERRED, 18
globus_gass_transfer_proto_send_t, 35	globus_gass_transfer_request_set_length, 20
globus_gass_transfer_proto_unregister_-	globus_gass_transfer_request_set_type, 20
protocol, 39	globus_gass_transfer_request_set_url, 20
globus_gass_transfer_request_proto_t, 35	globus_gass_transfer_request_set_user_-
globus_gass_transfer_put	pointer, 18
globus_gass_transfer_client, 5	GLOBAL_GASS_TRANSFER_REQUEST_-
globus_gass_transfer_receive_bytes	STARTING, 18
globus_gass_transfer_data, 14	globus_gass_transfer_request_status_t, 17
globus_gass_transfer_refer	GLOBAL_GASS_TRANSFER_REQUEST_-
globus_gass_transfer_server, 11	TYPE_APPEND, 17
globus_gass_transfer_referral	GLOBAL_GASS_TRANSFER_REQUEST_-
globus_gass_transfer_referral_destroy, 16	TYPE_GET, 17



GLOBUS\_GASS\_TRANSFER\_REQUEST\_-  
 TYPE\_INVALID, 17  
 GLOBUS\_GASS\_TRANSFER\_REQUEST\_-  
 TYPE\_PUT, 17  
 globus\_gass\_transfer\_request\_type\_t, 17  
 GLOBUS\_GASS\_TRANSFER\_REQUEST\_-  
 DENIED  
 globus\_gass\_transfer\_request, 18  
 globus\_gass\_transfer\_request\_destroy  
 globus\_gass\_transfer\_request, 22  
 GLOBUS\_GASS\_TRANSFER\_REQUEST\_-  
 DONE  
 globus\_gass\_transfer\_request, 18  
 GLOBUS\_GASS\_TRANSFER\_REQUEST\_-  
 FAILED  
 globus\_gass\_transfer\_request, 18  
 globus\_gass\_transfer\_request\_get\_denial\_message  
 globus\_gass\_transfer\_request, 21  
 globus\_gass\_transfer\_request\_get\_denial\_reason  
 globus\_gass\_transfer\_request, 21  
 globus\_gass\_transfer\_request\_get\_length  
 globus\_gass\_transfer\_request, 19  
 globus\_gass\_transfer\_request\_get\_referral  
 globus\_gass\_transfer\_request, 19  
 globus\_gass\_transfer\_request\_get\_status  
 globus\_gass\_transfer\_request, 19  
 globus\_gass\_transfer\_request\_get\_subject  
 globus\_gass\_transfer\_request, 21  
 globus\_gass\_transfer\_request\_get\_type  
 globus\_gass\_transfer\_request, 18  
 globus\_gass\_transfer\_request\_get\_url  
 globus\_gass\_transfer\_request, 19  
 globus\_gass\_transfer\_request\_get\_user\_pointer  
 globus\_gass\_transfer\_request, 18  
 GLOBUS\_GASS\_TRANSFER\_REQUEST\_-  
 INVALID  
 globus\_gass\_transfer\_request, 18  
 GLOBUS\_GASS\_TRANSFER\_REQUEST\_-  
 PENDING  
 globus\_gass\_transfer\_request, 18  
 globus\_gass\_transfer\_request\_proto\_s, 44  
 authorize, 45  
 deny, 44  
 destroy, 45  
 fail, 44  
 recv\_buffer, 44  
 refer, 45  
 send\_buffer, 44  
 globus\_gass\_transfer\_request\_proto\_t  
 globus\_gass\_transfer\_protocol, 35  
 GLOBUS\_GASS\_TRANSFER\_REQUEST\_-  
 REFERRED  
 globus\_gass\_transfer\_request, 18  
 globus\_gass\_transfer\_request\_set\_length  
 globus\_gass\_transfer\_request, 20  
 globus\_gass\_transfer\_request\_set\_type

globus\_gass\_transfer\_request, 20  
 globus\_gass\_transfer\_request\_set\_url  
 globus\_gass\_transfer\_request, 20  
 globus\_gass\_transfer\_request\_set\_user\_pointer  
 globus\_gass\_transfer\_request, 18  
 GLOBUS\_GASS\_TRANSFER\_REQUEST\_-  
 STARTING  
 globus\_gass\_transfer\_request, 18  
 globus\_gass\_transfer\_request\_status\_t  
 globus\_gass\_transfer\_request, 17  
 globus\_gass\_transfer\_request\_t, 45  
 GLOBUS\_GASS\_TRANSFER\_REQUEST\_-  
 TYPE\_APPEND  
 globus\_gass\_transfer\_request, 17  
 GLOBUS\_GASS\_TRANSFER\_REQUEST\_-  
 TYPE\_GET  
 globus\_gass\_transfer\_request, 17  
 GLOBUS\_GASS\_TRANSFER\_REQUEST\_-  
 TYPE\_INVALID  
 globus\_gass\_transfer\_request, 17  
 GLOBUS\_GASS\_TRANSFER\_REQUEST\_-  
 TYPE\_PUT  
 globus\_gass\_transfer\_request, 17  
 globus\_gass\_transfer\_request\_type\_t  
 globus\_gass\_transfer\_request, 17  
 globus\_gass\_transfer\_requestattr  
 globus\_gass\_transfer\_requestattr\_destroy, 24  
 globus\_gass\_transfer\_requestattr\_get\_block\_-  
 size, 27  
 globus\_gass\_transfer\_requestattr\_get\_-  
 connection\_reuse, 27  
 globus\_gass\_transfer\_requestattr\_get\_file\_-  
 mode, 27  
 globus\_gass\_transfer\_requestattr\_get\_proxy\_-  
 url, 27  
 globus\_gass\_transfer\_requestattr\_get\_socket\_-  
 nodelay, 28  
 globus\_gass\_transfer\_requestattr\_get\_socket\_-  
 rcvbuf, 28  
 globus\_gass\_transfer\_requestattr\_get\_socket\_-  
 sndbuf, 28  
 globus\_gass\_transfer\_requestattr\_init, 24  
 globus\_gass\_transfer\_requestattr\_set\_block\_-  
 size, 24  
 globus\_gass\_transfer\_requestattr\_set\_-  
 connection\_reuse, 25  
 globus\_gass\_transfer\_requestattr\_set\_file\_-  
 mode, 25  
 globus\_gass\_transfer\_requestattr\_set\_proxy\_-  
 url, 24  
 globus\_gass\_transfer\_requestattr\_set\_socket\_-  
 nodelay, 26  
 globus\_gass\_transfer\_requestattr\_set\_socket\_-  
 rcvbuf, 26  
 globus\_gass\_transfer\_requestattr\_set\_socket\_-  
 sndbuf, 25

- globus\_gass\_transfer\_secure\_requestattr\_get\_-  
authorization, 29
- globus\_gass\_transfer\_secure\_requestattr\_set\_-  
authorization, 26
- globus\_gass\_transfer\_requestattr\_destroy  
globus\_gass\_transfer\_requestattr, 24
- globus\_gass\_transfer\_requestattr\_get\_block\_size  
globus\_gass\_transfer\_requestattr, 27
- globus\_gass\_transfer\_requestattr\_get\_connection\_-  
reuse  
globus\_gass\_transfer\_requestattr, 27
- globus\_gass\_transfer\_requestattr\_get\_file\_mode  
globus\_gass\_transfer\_requestattr, 27
- globus\_gass\_transfer\_requestattr\_get\_proxy\_url  
globus\_gass\_transfer\_requestattr, 27
- globus\_gass\_transfer\_requestattr\_get\_socket\_-  
nodelay  
globus\_gass\_transfer\_requestattr, 28
- globus\_gass\_transfer\_requestattr\_get\_socket\_-  
rcvbuf  
globus\_gass\_transfer\_requestattr, 28
- globus\_gass\_transfer\_requestattr\_get\_socket\_-  
sndbuf  
globus\_gass\_transfer\_requestattr, 28
- globus\_gass\_transfer\_requestattr\_implementation  
globus\_gass\_transfer\_listenerattr\_initialize, 33
- globus\_gass\_transfer\_requestattr\_initialize, 31
- globus\_gass\_transfer\_secure\_requestattr\_-  
initialize, 32
- globus\_gass\_transfer\_socket\_requestattr\_-  
initialize, 32
- globus\_gass\_transfer\_requestattr\_init  
globus\_gass\_transfer\_requestattr, 24
- globus\_gass\_transfer\_requestattr\_initialize  
globus\_gass\_transfer\_requestattr\_-  
implementation, 31
- globus\_gass\_transfer\_requestattr\_set\_block\_size  
globus\_gass\_transfer\_requestattr, 24
- globus\_gass\_transfer\_requestattr\_set\_connection\_-  
reuse  
globus\_gass\_transfer\_requestattr, 25
- globus\_gass\_transfer\_requestattr\_set\_file\_mode  
globus\_gass\_transfer\_requestattr, 25
- globus\_gass\_transfer\_requestattr\_set\_proxy\_url  
globus\_gass\_transfer\_requestattr, 24
- globus\_gass\_transfer\_requestattr\_set\_socket\_-  
nodelay  
globus\_gass\_transfer\_requestattr, 26
- globus\_gass\_transfer\_requestattr\_set\_socket\_rcvbuf  
globus\_gass\_transfer\_requestattr, 26
- globus\_gass\_transfer\_requestattr\_set\_socket\_-  
sndbuf  
globus\_gass\_transfer\_requestattr, 25
- globus\_gass\_transfer\_secure\_requestattr\_get\_-  
authorization
- globus\_gass\_transfer\_requestattr, 29

- globus\_gass\_transfer\_secure\_requestattr\_initialize  
globus\_gass\_transfer\_requestattr\_-  
implementation, 32
- globus\_gass\_transfer\_secure\_requestattr\_set\_-  
authorization  
globus\_gass\_transfer\_requestattr, 26
- globus\_gass\_transfer\_send\_bytes  
globus\_gass\_transfer\_data, 13
- globus\_gass\_transfer\_server  
globus\_gass\_transfer\_authorize, 12
- globus\_gass\_transfer\_close\_callback\_t, 8
- globus\_gass\_transfer\_close\_listener, 9
- globus\_gass\_transfer\_create\_listener, 8
- globus\_gass\_transfer\_deny, 12
- globus\_gass\_transfer\_listen\_callback\_t, 8
- globus\_gass\_transfer\_listener\_get\_base\_url,  
11
- globus\_gass\_transfer\_listener\_get\_user\_-  
pointer, 10
- globus\_gass\_transfer\_listener\_set\_user\_-  
pointer, 11
- globus\_gass\_transfer\_refer, 11
- globus\_gass\_transfer\_register\_accept, 10
- globus\_gass\_transfer\_register\_listen, 9
- globus\_gass\_transfer\_socket\_requestattr\_initialize  
globus\_gass\_transfer\_requestattr\_-  
implementation, 32

Implementing Request Attributes, 31

Implementing Servers, 7

listen

- globus\_gass\_transfer\_listener\_proto\_s, 42

Listener attributes, 29

new\_listener

- globus\_gass\_transfer\_proto\_descriptor\_t, 43

new\_listenerattr

- globus\_gass\_transfer\_proto\_descriptor\_t, 43

new\_request

- globus\_gass\_transfer\_proto\_descriptor\_t, 43

new\_requestattr

- globus\_gass\_transfer\_proto\_descriptor\_t, 43

Protocol Modules, 33

recv\_buffer

- globus\_gass\_transfer\_request\_proto\_s, 44

refer

- globus\_gass\_transfer\_request\_proto\_s, 45

Referrals, 15

Request Attributes, 22

Request Handles, 16

send\_buffer

- globus\_gass\_transfer\_request\_proto\_s, 44

Sending and Receiving Data, 13

url\_scheme

globus\_gass\_transfer\_proto\_descriptor\_t, 42