globus gram protocol

7.4

Generated by Doxygen 1.6.1

Sun Oct 11 03:02:07 2009

Contents

1 Globus GRAM Protocol

The Globus GRAM Protocol Library implements the GRAM protocol. It is used by the GRAM Client and GRAM Job Manager. It provides the constants used by in the sending and receiving of GRAM messages. It also provides functions to encode GRAM requests and replies, and to send and receive the GRAM queries.

- GRAM Protocol Functions

2 GRAM Protocol Definition

The GRAM Protocol is used to handle communication between the Gatekeeper, Job Manager, and GRAM Clients.

The protocol is based on a subset of the HTTP/1.1 protocol, with a small set of message types and responses sent as the body of the HTTP requests and responses. This document describes GRAM Protocol version 2.

Framing

GRAM messages are framed in HTTP/1.1 messages. However, only a small subset of the HTTP specification is used or understood by the GRAM system. All GRAM requests are HTTP POST messages. Only the following HTTP headers are understood:

- Host
- Content-Type (set to "application/x-globus-gram" in all cases)
- Content-Length
- Connection (set to "close" in all HTTP responses)

Only the following status codes are supported in response's HTTP Status-Lines:

- 200 OK
- 403 Forbidden
- 404 Not Found
- 500 Internal Server Error
- 400 Bad Request

Message Format

All messages use the carriage return (ASCII value 13) followed by line feed (ASCII value 10) sequence to delimit lines. In all cases, a blank line separates the HTTP header from the message body. All **application/x-globus-gram** message bodies consist of attribute names followed by a colon, a space, and then the value of the attribute. When the value may contain a newline or double-quote character, a special escaping rule is used to encapsulate the complete string. This encapsulation consists of surrounding the string with double-quotes, and escaping all double-quote and backslash characters within the string with a backslash. All other characters are sent without modification. For example, the string

```
rsl: &( executable = "/bin/echo" )
 ( arguments = "hello" )
```

becomes

```
rsl: "&( executable = \"bin/echo\" )
  (arguments = \"hello\" )"
```

This is the only form of quoting which **application/x-globus-gram** messages support. Use of % HEX HEX escapes (such as seen in URL encodings) is not meaningful for this protocol.

Message Types

Ping Request

A ping request is used to verify that the gatekeeper is configured properly to handle a named service. The ping request consists of the following:

```
POST ping/ job-manager-name HTTP/1.1
Host:  host-name
Content-Type: application/x-globus-gram
Content-Length:  message-size

protocol-version:  version
```

The values of the message-specific strings are

job-manager-name The name of the service to have the gatekeeper check. The service name corresponds to one of the gatekeeper's configured grid-services, and is usually of the form "jobmanager-scheduler-type".

host-name The name of the host on which the gatekeeper is running. This exists only for compatibility with the HTTP/1.1 protocol.

message-size The length of the content of the message, not including the HTTP/1.1 header.

version The version of the GRAM protocol which is being used. For the protocol defined in this document, the value must be the string "2".

Job Request

A job request is used to scheduler a job remotely using GRAM. The ping request consists of the HTTP framing described above with the request-URI consisting of *job-manager-name*, where *job-manager name* is the name of the service to use to schedule the job. The format of a job request message consists of the following:

```
POST  job-manager-name[@ user-name] HTTP/1.1
Host:  host-name
Content-Type: application/x-globus-gram
Content-Length:  message-size

protocol-version:  version
job-state-mask:  mask
callback-url:  callback-contact
rsl:  rsl-description
```

The values of the emphasized text items are as below:

job-manager-name The name of the service to submit the job request to. The service name corresponds to one of the gatekeeper's configured grid-services, and is usually of the form "jobmanager-scheduler-type".

user-name Starting with GT4.0, a client may request that a certain account by used by the gatekeeper to start the job manager. This is done optionally by appending the @ symbol and the local user name that the job should be run as to the *job-manager-name*. If the @ and username are not present, then the first grid map entry will be used. If the client credential is not authorized in the grid map to use the specified account, an authorization error will occur in the gatekeeper.

host-name The name of the host on which the gatekeeper is running. This exists only for compatibility with the HTTP/1.1 protocol.

message-size The length of the content of the message, not including the HTTP/1.1 header.

version The version of the GRAM protocol which is being used. For the protocol defined in this document, the value must be the string "2".

mask An integer representation of the job state mask. This value is obtained from a bitwise-OR of the job state values which the client wishes to receive job status callbacks about. These meanings of the various job state values are defined in the GRAM Protocol API documentation.

callback-contact A https URL which defines a GRAM protocol listener which will receive job state updates. The from a bitwise-OR of the job state values which the client wishes to receive job status callbacks about. The job status update messages are defined below.

rsl-description A quoted string containing the RSL description of the job request.

Status Request

A status request is used by a GRAM client to get the current job state of a running job. This type of message can only be sent to a job manager's job-contact (as returned in the reply to a job request message). The format of a job request message consists of the following:

```
POST  job-contact HTTP/1.1
Host:  host-name
Content-Type: application/x-globus-gram
Content-Length:  message-size

protocol-version:  version
"status"
```

The values of the emphasized text items are as below:

job-contact The job contact string returned in a response to a job request message, or determined by querying the MDS system.

host-name The name of the host on which the job manager is running. This exists only for compatibility with the HTTP/1.1 protocol.

message-size The length of the content of the message, not including the HTTP/1.1 header.

version The version of the GRAM protocol which is being used. For the protocol defined in this document, the value must be the string "2".

Callback Register Request

A callback register request is used by a GRAM client to register a new callback contact to receive GRAM job state updates. This type of message can only be sent to a job manager's job-contact (as returned in the reply to a job request message). The format of a job request message consists of the following:

```
POST  job-contact HTTP/1.1
Host:  host-name
Content-Type: application/x-globus-gram
Content-Length:  message-size

protocol-version:  version
"register <em>mask</em> <em>callback-contact</em>"
```

The values of the emphasized text items are as below:

job-contact The job contact string returned in a response to a job request message, or determined by querying the MDS system.

host-name The name of the host on which the job manager is running. This exists only for compatibility with the HTTP/1.1 protocol.

message-size The length of the content of the message, not including the HTTP/1.1 header.

version The version of the GRAM protocol which is being used. For the protocol defined in this document, the value must be the string "2".

mask An integer representation of the job state mask. This value is obtained from a bitwise-OR of the job state values which the client wishes to receive job status callbacks about. These meanings of the various job state values are defined in the GRAM Protocol API documentation.

callback-contact A https URL which defines a GRAM protocol listener which will receive job state updates. The from a bitwise-OR of the job state values which the client wishes to receive job status callbacks about. The job status update messages are defined below.

Callback Unregister Request

A callback unregister request is used by a GRAM client to request that the job manager no longer send job state updates to the specified callback contact. This type of message can only be sent to a job manager's job-contact (as returned in the reply to a job request message). The format of a job request message consists of the following:

```
POST  job-contact HTTP/1.1
Host:  host-name
Content-Type: application/x-globus-gram
Content-Length:  message-size

protocol-version:  version
"unregister <em>callback-contact</em>"
```

The values of the emphasized text items are as below:

job-contact The job contact string returned in a response to a job request message, or determined by querying the MDS system.

host-name The name of the host on which the job manager is running. This exists only for compatibility with the HTTP/1.1 protocol.

message-size The length of the content of the message, not including the HTTP/1.1 header.

version The version of the GRAM protocol which is being used. For the protocol defined in this document, the value must be the string "2".

callback-contact A https URL which defines a GRAM protocol listener which should no longer receive job state updates. The from a bitwise-OR of the job state values which the client wishes to receive job status callbacks about. The job status update messages are defined below.

Job Cancel Request

A job cancel request is used by a GRAM client to request that the job manager terminate a job. This type of message can only be sent to a job manager's job-contact (as returned in the reply to a job request message). The format of a job request message consists of the following:

```
POST  job-contact HTTP/1.1
Host:  host-name
Content-Type: application/x-globus-gram
Content-Length:  message-size

protocol-version:  version
"cancel"
```

The values of the emphasized text items are as below:

job-contact The job contact string returned in a response to a job request message, or determined by querying the MDS system.

host-name The name of the host on which the job manager is running. This exists only for compatibility with the HTTP/1.1 protocol.

message-size The length of the content of the message, not including the HTTP/1.1 header.

version The version of the GRAM protocol which is being used. For the protocol defined in this document, the value must be the string "2".

Job Signal Request

A job signal request is used by a GRAM client to request that the job manager process a signal for a job. The arguments to the various signals are discussed in the globus_gram_protocol_job_signal_t documentation.

```
POST  job-contact HTTP/1.1
Host:  host-name
Content-Type: application/x-globus-gram
Content-Length:  message-size

protocol-version:  version
"<em>signal</em>"
```

The values of the emphasized text items are as below:

job-contact The job contact string returned in a response to a job request message, or determined by querying the MDS system.

host-name The name of the host on which the job manager is running. This exists only for compatibility with the HTTP/1.1 protocol.

message-size The length of the content of the message, not including the HTTP/1.1 header.

version The version of the GRAM protocol which is being used. For the protocol defined in this document, the value must be the string "2".

signal A quoted string containing the signal number and it's parameters.

Job State Updates

A job status update message is sent by the job manager to all registered callback contacts when the job's status changes. The format of the job status update messages is as follows:

```
POST  callback-contact HTTP/1.1
Host:  host-name
Content-Type: application/x-globus-gram
Content-Length:  message-size
```

```
protocol-version:  version
job-manager-url:  job-contact
status:  status-code
failure-code:  failure-code
```

The values of the emphasized text items are as below:

callback-contact The callback contact string registered with the job manager either by being passed as the *callback-contact* in a job request message or in a callback register message.

host-name The host part of the callback-contact URL. This exists only for compatibility with the HTTP/1.1 protocol.

message-size The length of the content of the message, not including the HTTP/1.1 header.

version The version of the GRAM protocol which is being used. For the protocol defined in this document, the value must be the string "2".

job-contact The job contact of the job which has changed states.

Proxy Delegation

A proxy delegation message is sent by the client to the job manager to initiate a delegation handshake to generate a new proxy credential for the job manager. This credential is used by the job manager or the job when making further secured connections. The format of the delegation message is as follows:

```
POST  callback-contact HTTP/1.1
Host:  host-name
Content-Type: application/x-globus-gram
Content-Length:  message-size

protocol-version:  version
"renew"
```

If a successful (200) reply is sent in response to this message, then the client will procede with a GSI delegation handshake. The tokens in this handshake will be framed with a 4 byte big-endian token length header. The framed tokens will then be wrapped using the GLOBUS_IO_SECURE_CHANNEL_MODE_SSL_WRAP wrapping mode. The job manager will frame response tokens in the same manner. After the job manager receives its final delegation token, it will respond with another response message that indicates whether the delegation was processed or not. This response message is a standard GRAM response message.

Note on Security Attributes The following security attributes are needed to communicate with the Gatekeeper:

- Authentication must be done using GSSAPI mutual authentication
- Messages must be wrapped with support for the delegation message. When using Globus I/O, this is accomplished by using the the GLOBUS_IO_SECURE_CHANNEL_MODE_GSI_WRAP wrapping mode.

Changes 2004-08-11 Added information about gridmap choosing

3 Module Index

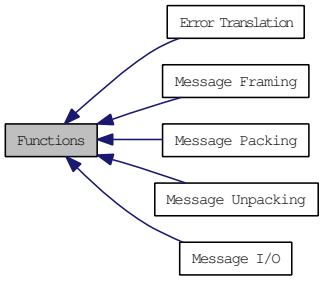
3.1 Modules

Here is a list of all modules:

4 Module Documentation

4.1 Functions

Collaboration diagram for Functions:



Modules

- Error Translation
- Message Framing
- Message I/O
- Message Packing
- Message Unpacking

4.2 GRAM Signals

Enumerations

- enum globus_gram_protocol_job_signal_t {
  GLOBUS_GRAM_PROTOCOL_JOB_SIGNAL_CANCEL = 1,
  GLOBUS_GRAM_PROTOCOL_JOB_SIGNAL_SUSPEND = 2,
  GLOBUS_GRAM_PROTOCOL_JOB_SIGNAL_RESUME = 3,
  GLOBUS_GRAM_PROTOCOL_JOB_SIGNAL_PRIORITY = 4,

GLOBUS_GRAM_PROTOCOL_JOB_SIGNAL_COMMIT_REQUEST = 5,
GLOBUS_GRAM_PROTOCOL_JOB_SIGNAL_COMMIT_EXTEND = 6,
GLOBUS_GRAM_PROTOCOL_JOB_SIGNAL_STDIO_UPDATE = 7,
GLOBUS_GRAM_PROTOCOL_JOB_SIGNAL_STDIO_SIZE = 8,
GLOBUS_GRAM_PROTOCOL_JOB_SIGNAL_STOP_MANAGER = 9,
GLOBUS_GRAM_PROTOCOL_JOB_SIGNAL_COMMIT_END = 10 }

4.2.1 Detailed Description

4.2.2 Enumeration Type Documentation

4.2.2.1 enum globus_gram_protocol_job_signal_t

GRAM Signals.

Enumerator:

GLOBUS_GRAM_PROTOCOL_JOB_SIGNAL_CANCEL Cancel a job.

GLOBUS_GRAM_PROTOCOL_JOB_SIGNAL_SUSPEND Suspend a job.

GLOBUS_GRAM_PROTOCOL_JOB_SIGNAL_RESUME Resume a previously suspended job.

GLOBUS_GRAM_PROTOCOL_JOB_SIGNAL_PRIORITY Change the priority of a job.

GLOBUS_GRAM_PROTOCOL_JOB_SIGNAL_COMMIT_REQUEST Signal the job manager to commence with a job submission if the job request was accompanied by the (two_state=yes) RSL attribute.

GLOBUS_GRAM_PROTOCOL_JOB_SIGNAL_COMMIT_EXTEND Signal the job manager to wait an additional number of seconds (specified by an integer value string as the signal's argument) before timing out a two-phase job commit.

GLOBUS_GRAM_PROTOCOL_JOB_SIGNAL_STDIO_UPDATE Signal the job manager to change the way it is currently handling standard output and/or standard error. The argument for this signal is an RSL containing new *stdout*, *stderr*, *stdout_position*, *stderr_position*, or *remote_io_url* relations.

GLOBUS_GRAM_PROTOCOL_JOB_SIGNAL_STDIO_SIZE Signal the job manager to verify that streamed I/O has been completely received. The argument to this signal contains the number of bytes of stdout and stderr received, seperated by a space. The reply to this signal will be a SUCCESS message if these matched the amount sent by the job manager. Otherwise, an error reply indicating GLOBUS_GRAM_PROTOCOL_ERROR_STDIO_SIZE is returned. If standard output and standard error are merged, only one number should be sent as an argument to this signal. An argument of -1 for either stream size indicates that the client is not interested in the size of that stream.

GLOBUS_GRAM_PROTOCOL_JOB_SIGNAL_STOP_MANAGER Signal the job manager to stop managing the current job and terminate. The job continues to run as normal. The job manager will send a state change callback with the job status being FAILED and the error GLOBUS_GRAM_PROTOCOL_ERROR_JM_STOPPED.

GLOBUS_GRAM_PROTOCOL_JOB_SIGNAL_COMMIT_END Signal the job manager to clean up after the completion of the job if the job RSL contained the (two-phase = yes) relation.

4.3 GRAM Job States

The globus_gram_protocol_job_state_t contains information about the current state of the job as known by the job manager.

Enumerations

- enum globus_gram_protocol_job_state_t {
  GLOBUS_GRAM_PROTOCOL_JOB_STATE_PENDING = 1,
  GLOBUS_GRAM_PROTOCOL_JOB_STATE_ACTIVE = 2,
  GLOBUS_GRAM_PROTOCOL_JOB_STATE_FAILED = 4,
  GLOBUS_GRAM_PROTOCOL_JOB_STATE_DONE = 8,
  GLOBUS_GRAM_PROTOCOL_JOB_STATE_SUSPENDED = 16,
  GLOBUS_GRAM_PROTOCOL_JOB_STATE_UNSUBMITTED = 32,
  GLOBUS_GRAM_PROTOCOL_JOB_STATE_STAGE_IN = 64,
  GLOBUS_GRAM_PROTOCOL_JOB_STATE_STAGE_OUT = 128,
  GLOBUS_GRAM_PROTOCOL_JOB_STATE_ALL = 0xFFFFF }

4.3.1 Detailed Description

The globus_gram_protocol_job_state_t contains information about the current state of the job as known by the job manager. Job state changes are sent by the Job Manager to all registered clients. A client may ask for information from the job manager via the status request.

4.3.2 Enumeration Type Documentation

4.3.2.1 enum globus_gram_protocol_job_state_t

GRAM Job States.

Enumerator:

GLOBUS_GRAM_PROTOCOL_JOB_STATE_PENDING The job is waiting for resources to become available to run.

GLOBUS_GRAM_PROTOCOL_JOB_STATE_ACTIVE The job has received resources and the application is executing.

GLOBUS_GRAM_PROTOCOL_JOB_STATE_FAILED The job terminated before completion because an error, user-triggered cancel, or system-triggered cancel.

GLOBUS_GRAM_PROTOCOL_JOB_STATE_DONE The job completed successfully.

GLOBUS_GRAM_PROTOCOL_JOB_STATE_SUSPENDED The job has been suspended. Resources which were allocated for this job may have been released due to some scheduler-specific reason.

GLOBUS_GRAM_PROTOCOL_JOB_STATE_UNSUBMITTED The job has not been submitted to the scheduler yet, pending the reception of the GLOBUS_GRAM_PROTOCOL_JOB_SIGNAL_-COMMIT_REQUEST signal from a client.

GLOBUS_GRAM_PROTOCOL_JOB_STATE_STAGE_IN The job manager is staging in files to run the job.

GLOBUS_GRAM_PROTOCOL_JOB_STATE_STAGE_OUT The job manager is staging out files generated by the job.

GLOBUS_GRAM_PROTOCOL_JOB_STATE_ALL A mask of all job states.

4.4 GRAM Error codes

Enumerations

- enum globus_gram_protocol_error_t

4.4.1 Detailed Description

4.4.2 Enumeration Type Documentation

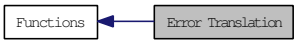
4.4.2.1 enum globus_gram_protocol_error_t

GRAM Error codes.

4.5 Error Translation

Functions in this section handle translating GRAM error codes to strings which can help the user diagnose GRAM problems.

Collaboration diagram for Error Translation:



Functions

- const char * globus_gram_protocol_error_string (int error_code)
- void globus_gram_protocol_error_7_hack_replace_message (const char *message)

4.5.1 Detailed Description

Functions in this section handle translating GRAM error codes to strings which can help the user diagnose GRAM problems.

4.5.2 Function Documentation

4.5.2.1 const char* globus_gram_protocol_error_string (int *error_code*)

Error code translation. This function takes the error code value and returns the associated error code string. The string is statically allocated by the Globus GRAM Protocol library and should not be modified or freed.

Parameters:

error_code The error code to look up.

Returns:

An error string containing the reason for the error. The error string is written to be used in the context "[operation] failed because [error_string]".

4.5.2.2 void globus_gram_protocol_error_7_hack_replace_message (const char * *message*)

GSI specific error code hack. This function creates a custom version of the error message for the error GLOBUS_-GRAM_PROTOCOL_ERROR_AUTHORIZATION. **This function should really only used by the GRAM client library.**

Parameters:

message The new message to be associated with error code 7.

4.6 Message Framing

The functions in this section take GRAM request (or query) and reply messages, and frame them with HTTP headers, so that they can be sent.

Collaboration diagram for Message Framing:

Functions

- int globus_gram_protocol_frame_request (const char *url, const globus_byte_t *msg, globus_size_t msg-size, globus_byte_t **framedmsg, globus_size_t *framedsize)
- int globus_gram_protocol_frame_reply (int code, const globus_byte_t *msg, globus_size_t msgsize, globus_byte_t **framedmsg, globus_size_t *framedsize)

4.6.1 Detailed Description

The functions in this section take GRAM request (or query) and reply messages, and frame them with HTTP headers, so that they can be sent. These functions should be used when an application wants to control the way that the GRAM Protocol messages are sent, while still using the standard message formatting and framing routines. An alternative set of functions in the Message I/O section of the manual combine message framing with callback-driven I/O.

4.6.2 Function Documentation

4.6.2.1 int globus_gram_protocol_frame_request (const char * *url*, const globus_byte_t * *msg*, globus_size_t *msgsize*, globus_byte_t ** *framedmsg*, globus_size_t * *framedsize*)

Frame a GRAM query

Adds an HTTP frame around a GRAM protocol message. The frame is constructed from the URL, the GRAM protocol message type header, and a message length header. The framed message is returned in a new string pointed to by *framedmsg* parameter and the length of the framed message is returned in the *framedsize* parameter.

Parameters:

url The URL of the GRAM resource to contact.

msg The message to be framed.

msgsize The length of the unframed message.

framedmsg A return parameter, which will contain the framed message upon this function's return.

framedsize A return parameter, which will contain the length of the framed message.

4.6.2.2 int globus_gram_protocol_frame_reply (int *code*, const globus_byte_t * *msg*, globus_size_t *msgsize*, globus_byte_t ** *framedmsg*, globus_size_t * *framedsize*)

Frame a GRAM reply

Adds an HTTP frame around a GRAM protocol reply. The frame is constructed from the message code passed as the first parameter. The framed reply is returned in a new string pointed to by *framedmsg* parameter and the length of the framed reply is returned in the *framedsize* parameter.

Parameters:

code The HTTP response code to associate with this reply.

msg The reply to be framed.

msgsize The length of the unframed reply.

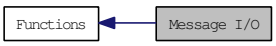
framedmsg A return parameter, which will contain the framed reply upon this function's return.

framedsize A return parameter, which will contain the length of the framed reply.

4.7 Message I/O

The functions in this section.

Collaboration diagram for Message I/O:



Functions

- int globus_gram_protocol_allow_attach (char **url, globus_gram_protocol_callback_t callback, void *callback_arg)
- int globus_gram_protocol_callback_disallow (char *url)
- int globus_gram_protocol_post (const char *url, globus_gram_protocol_handle_t *handle, globus_io_attr_t *attr, globus_byte_t *message, globus_size_t message_size, globus_gram_protocol_callback_t callback, void *callback_arg)
- int globus_gram_protocol_reply (globus_gram_protocol_handle_t handle, int code, globus_byte_t *message, globus_size_t message_size)
- int globus_gram_protocol_get_sec_context (globus_gram_protocol_handle_t handle, gss_ctx_id_t *context)

4.7.1 Detailed Description

The functions in this section.

4.7.2 Function Documentation

4.7.2.1 int globus_gram_protocol_allow_attach (char ** *url*, globus_gram_protocol_callback_t *callback*, void * *callback_arg*)

Create a GRAM Protocol listener. Creates a GRAM Protocol listener. The listener will automatically accept new connections on it's TCP/IP port and parse GRAM requests. The requests will be passed to the specified *callback* function to the user can unpack the request, handle it, and send a reply by calling globus_gram_protocol_reply().

Parameters:

url A pointer to a character array which will be allocated to hold the URL of the listener. This URL may be published or otherwise passed to applications which need to contact this protocol server. The URL will be of the form `https://<host>:<port>/`. It is the user's responsibility to free this memory.

callback The callback function to be called when a new request has been received by this listener. This function will be passed the request, which may be unpacked using one of the functions described in the message packing section of the documentation.

callback_arg A pointer to arbitrary user data which will be passed to the callback function as it's first parameter.

Return values:

GLOBUS_SUCCESS The listener was created. The *url* parameter points to a string containing the contact URL for the listener.

GLOBUS_GRAM_PROTOCOL_ERROR_INVALID_REQUEST The GRAM Protocol module was not properly activated.

GLOBUS_GRAM_PROTOCOL_ERROR_MALLOC_FAILED A memory allocation failed when trying to create the listener.

GLOBUS_GRAM_PROTOCOL_ERROR_NO_RESOURCES Some I/O error occurred when trying to create the listener.

See also:

globus_gram_protocol_callback_disallow()

4.7.2.2 int globus_gram_protocol_callback_disallow (char * *url*)

Disable a listener from handling any new requests. Disables a listener making it unable to receive any new requests, and freeing memory associated with the listener. Will block if a request is in progress, but once this function returns, no further request callbacks create by the listener will occur.

Parameters:

url The URL of the listener to disable.

Return values:

GLOBUS_SUCCESS The listener was closed. No further callbacks will be called on behalf of this listener.

GLOBUS_GRAM_PROTOCOL_ERROR_INVALID_JOB_CONTACT The *url* string could not be parsed.

GLOBUS_GRAM_PROTOCOL_ERROR_CALLBACK_NOT_FOUND The GRAM protocol library doesn't know of any listener associated with this URL.

See also:

globus_gram_protocol_allow_attach()

4.7.2.3 int globus_gram_protocol_post (const char * *url*, globus_gram_protocol_handle_t * *handle*, globus_io_attr_t * *attr*, globus_byte_t * *message*, globus_size_t *message_size*, globus_gram_protocol_callback_t *callback*, void * *callback_arg*)

Frame and send a GRAM protocol request. Connects to the GRAM Protocol server specified by the *url* parameter, frames the message with HTTP headers, and sends it. If *callback* is non-NULL, then the function pointed to by it will be called when a response is received from the server.

Parameters:

url The URL of the server to send the message to. The *url* may be freed once this function returns.

handle A pointer to a globus_gram_protocol_handle_t which will be initialized with a unique handle identifier. This identifier will be passed to the *callback* function to allow the caller to differentiate replies to multiple GRAM Protocol servers.

attr A pointer to a Globus I/O attribute set, which will be used as parameters when connecting to the GRAM server. The attribute set may be GLOBUS_NULL, in which case, the default GRAM Protocol attributes will be used (authentication to self, SSL-compatible transport, with message integrity).

message A pointer to a message array to be sent to the GRAM server. This is normally created by calling one of the GRAM Protocol pack functions. This message need not be NULL terminated. The memory associated with *message* may be freed as soon as this function returns.

message_size The length of the *message* string. Typically generated as one of the output parameters to one of the GRAM Protocol pack functions.

callback A pointer to a callback function to call when the response to this message is received. This may be GLOBUS_NULL, in which case no callback will be received, and the caller will be unable to verify whether the message was successfully received.

callback_arg A pointer to arbitrary user data which will be passed to the callback function as it's first parameter.

Return values:

GLOBUS_SUCCESS The message was successfully framed, and is in the process of being sent.

GLOBUS_GRAM_PROTOCOL_ERROR_INVALID_JOB_CONTACT The *url* parameter could not be parsed.

GLOBUS_GRAM_PROTOCOL_ERROR_MALLOC_FAILED A memory allocation failed when trying to frame or send the message.

GLOBUS_GRAM_PROTOCOL_ERROR_INVALID_REQUEST The GRAM Protocol module was not properly activated.

GLOBUS_GRAM_PROTOCOL_ERROR_NO_RESOURCES Some I/O error occurred when trying to send the message.

See also:

globus_gram_protocol_reply()

4.7.2.4 int globus_gram_protocol_reply (globus_gram_protocol_handle_t *handle*, int *code*, globus_byte_t * *message*, globus_size_t *message_size*)

Frame and send a GRAM protocol reply. On an existing handle, frame and send the reply. The reply consists of a response code and a message.

This function should only be called in response to a callback containing a GRAM Protocol request. It should not be called using the same handle as created by calling globus_gram_protocol_post().

Parameters:

handle The GRAM Protocol handle created when a connection arrives on a listener created by globus_gram_protocol_allow_attach(). The handle will be passed to the callback. The user must reply to all request callbacks which they receive.

code A response code. The code should be one from the standard HTTP response codes described in RFC XXX.

message A pointer to a message array to be sent to the GRAM client. This is normally created by calling one of the GRAM Protocol pack functions. This message need not be NULL terminated. The memory associated with *message* may be freed as soon as this function returns.

message_size The length of the *message* string. Typically generated as one of the output parameters to one of the GRAM Protocol pack functions.

Return values:

GLOBUS_SUCCESS The reply was successfully framed and is being sent.

GLOBUS_GRAM_PROTOCOL_ERROR_INVALID_REQUEST The GRAM Protocol module was not properly activated.

GLOBUS_GRAM_PROTOCOL_ERROR_NO_RESOURCES Some I/O error occurred when trying to send the reply.

See also:

globus_gram_protocol_post()

4.7.2.5 int globus_gram_protocol_get_sec_context (globus_gram_protocol_handle_t *handle*, gss_ctx_id_t * *context*)

Extract the GSS Context from a GRAM Connection

Extract the GSS Context from a existing, connected handle. This function should only be called after the GRAM protocol connection has been established.

Parameters:

handle The GRAM Protocol handle created when a connection arrives on a listener created by globus_gram_protocol_allow_attach().

context The GSS Context associated with the connection.

Return values:

GLOBUS_SUCCESS The reply was successfully framed and is being sent.

GLOBUS_GRAM_PROTOCOL_ERROR_INVALID_REQUEST The GRAM Protocol module was not properly activated.

4.8 Message Packing

Collaboration diagram for Message Packing:

Functions

- int globus_gram_protocol_pack_job_request (int job_state_mask, const char *callback_url, const char *rsl, globus_byte_t **query, globus_size_t *querysize)
- int globus_gram_protocol_pack_job_request_reply (int status, const char *job_contact, globus_byte_t **reply, globus_size_t *replysize)
- int globus_gram_protocol_pack_status_request (const char *status_request, globus_byte_t **query, globus_size_t *querysize)
- int globus_gram_protocol_pack_status_reply (int job_status, int failure_code, int job_failure_code, globus_byte_t **reply, globus_size_t *replysize)
- int globus_gram_protocol_pack_status_update_message (char *job_contact, int status, int failure_code, globus_byte_t **reply, globus_size_t *replysize)

4.8.1 Function Documentation

4.8.1.1 int globus_gram_protocol_pack_job_request (int *job_state_mask*, const char * *callback_url*, const char * *rsl*, globus_byte_t ** *query*, globus_size_t * *querysize*)

Pack a GRAM Job Request

Encodes the parameters of a job request in a GRAM protocol message. The resulting message may be sent with globus_gram_protocol_post() or framed with globus_gram_protocol_frame_request() and sent by the application.

Parameters:

job_state_mask The bitwise-or of the job states which the client would like to register for job state change callbacks.

callback_url A callback contact string which will be contacted when a job state change which matches the *job_state_mask* occurs. This may be NULL, if the client does not wish to register a callback contact with this job request.

rsl An RSL string which contains the job request. This will be parsed and validated on the server side.

query An output variable which will be populated with a new string containing the packed job request message. The caller must free this memory by calling globus_libc_free();

querysize An output variable which will be populated with the length of the job request message returned in *query*.

4.8.1.2 int globus_gram_protocol_pack_job_request_reply (int *status*, const char * *job_contact*, globus_byte_t ** *reply*, globus_size_t * *replysize*)

Pack a GRAM reply message

Encodes the parameters of a reply to a job request in a GRAM protocol message. The resulting message may be sent with globus_gram_protocol_post() or framed with globus_gram_protocol_frame_request() and sent by the application.

Parameters:

status The job's failure code if the job failed, or 0, if the job request was processed successfully.

job_contact A string containing the job's contact string, which may be used to contact the job manager to query or cancel the job. This may be NULL, if the job request was not successfull.

reply A pointer which will be set to point to a newly allocated reply string. The string must be freed by the caller with globus_libc_free()

replysize The length of the reply string.

Return values:

GLOBUS_SUCCESS The reply was successfully constructed.

GLOBUS_GRAM_PROTOCOL_MALLOC_FAILED Memory for the reply string could not be allocated.

4.8.1.3 int globus_gram_protocol_pack_status_request (const char * *status_request*, globus_byte_t ** *query*, globus_size_t * *querysize*)

Pack a GRAM Job Manager Query

Encodes the parameters of a job status request, or other GRAM query in a GRAM protocol message. The resulting message may be sent with globus_gram_protocol_post() or framed with globus_gram_protocol_frame_request() and sent by the application.

Parameters:

status_request A string containing the type of query. This may be "status", "register", "unregister", "signal", or "cancel".

query An output variable which will be populated with a new string containing the packed job query message.

querysize An output variable which will be populated with the length of the job query message returned in *query*.

4.8.1.4 int globus_gram_protocol_pack_status_reply (int *job_status*, int *failure_code*, int *job_failure_code*, globus_byte_t ** *reply*, globus_size_t * *replysize*)

Pack a GRAM reply message

Encodes the parameters of a reply to a job manager query in a GRAM protocol message. The resulting message may be sent with globus_gram_protocol_reply(). globus_gram_protocol_frame_reply() and sent by the application.

Parameters:

job_status The job's current job state.

failure_code The error code generated by the query. This may be GLOBUS_SUCCESS if the query succeeded.

job_failure_code The error code associated with the job if it has failed. This may be GLOBUS_SUCCESS if the job has not failed.

reply A pointer which will be set to point to a newly allocated reply string. The string must be freed by the caller with globus_libc_free()

replysize The length of the reply string.

Return values:

GLOBUS_SUCCESS The reply was successfully constructed.

GLOBUS_GRAM_PROTOCOL_MALLOC_FAILED Memory for the reply string could not be allocated.

4.8.1.5 int globus_gram_protocol_pack_status_update_message (char * *job_contact*, int *status*, int *failure_code*, globus_byte_t ** *reply*, globus_size_t * *replysize*)

Pack a status update message

Encodes the current status of a job in a GRAM protocol message. The resulting message may be sent with globus_gram_protocol_post() or framed with globus_gram_protocol_frame_request() and sent by the application. Status messages are sent by the job manager when the job's state changes.

Parameters:

job_contact The contact string associated with this job manager.

status The job's current job state.

failure_code The error associated with this job request, if the *status* value is GLOBUS_GRAM_-PROTOCOL_JOB_STATE_FAILED.

reply An output variable which will be populated with a new string containing the packed status message. The caller must free this memory by calling globus_libc_free();

replysize An output variable which will be populated with the length of the job request message returned in *reply*.

4.9 Message Unpacking

Collaboration diagram for Message Unpacking:

Functions

- int globus_gram_protocol_unpack_job_request (const globus_byte_t *query, globus_size_t querysize, int *job_state_mask, char **callback_url, char **description)
- int globus_gram_protocol_unpack_job_request_reply (const globus_byte_t *reply, globus_size_t replysize, int *status, char **job_contact)
- int globus_gram_protocol_unpack_status_request (const globus_byte_t *query, globus_size_t querysize, char **status_request)
- int globus_gram_protocol_unpack_status_reply (const globus_byte_t *reply, globus_size_t replysize, int *job_status, int *failure_code, int *job_failure_code)
- int globus_gram_protocol_unpack_status_update_message (const globus_byte_t *reply, globus_size_t replysize, char **job_contact, int *status, int *failure_code)

4.9.1 Function Documentation

4.9.1.1 int globus_gram_protocol_unpack_job_request (const globus_byte_t * *query*, globus_size_t *querysize*, int * *job_state_mask*, char ** *callback_url*, char ** *description*)

Unpack a job request

Extracts the parameters of a job request from a GRAM message. The parameters to this function mirror those of globus_gram_protocol_pack_job_request().

Parameters:

query The job request.

querysize The length of the job request string.

job_state_mask A pointer to an integer to be populated with the job state mask in the job request.

callback_url A pointer to be populated with a copy of the URL of the callback contact to be registered for this job request. The caller must free this memory by calling globus_libc_free().

description A pointer to be populated with a copy of the job description RSL for this job request. The caller must free this memory by calling globus_libc_free().

4.9.1.2 int globus_gram_protocol_unpack_job_request_reply (const globus_byte_t * *reply*, globus_size_t *replysize*, int * *status*, char ** *job_contact*)

Unpack a reply to a GRAM job request

Extracts the parameters of a reply to a job request from a GRAM message. The parameters to this function mirror those of globus_gram_protocol_pack_job_request_reply().

Parameters:

reply The job request reply.

replysize The length of the reply string.

status A pointer to an integer to be populated with the failure code associated with the job request. This may be GLOBUS_SUCCESS, if the job request was successful.

job_contact A pointer to a string to be populated with the job's contact string. This may set to NULL if the job request failed. If non-NULL upon return, the caller must free this string using globus_libc_free().

Return values:

GLOBUS_SUCCESS The reply was successfully unpacked.

GLOBUS_GRAM_PROTOCOL_ERROR_MALLOC_FAILED Memory for the *job_contact* string could not be allocated.

GLOBUS_GRAM_PROTOCOL_ERROR_HTTP_UNPACK_FAILED The reply message couldn't be parsed.

GLOBUS_GRAM_PROTOCOL_ERROR_VERSION_MISMATCH The reply message was in an incompatible version of the GRAM protocol.

4.9.1.3 int globus_gram_protocol_unpack_status_request (const globus_byte_t * *query*, globus_size_t *querysize*, char ** *status_request*)

Unpack a GRAM query

Extracts the parameters of a query from a GRAM message. The parameters to this function mirror those of globus_gram_protocol_pack_status_request().

Parameters:

query The GRAM query.

querysize The length of the query string.

status_request A pointer to a string to be populated with the query string. The caller must free this string using globus_libc_free().

Return values:

GLOBUS_SUCCESS The reply was successfully unpacked.

GLOBUS_GRAM_PROTOCOL_ERROR_MALLOC_FAILED Memory for the *job_contact* string could not be allocated.

GLOBUS_GRAM_PROTOCOL_ERROR_HTTP_UNPACK_FAILED The reply message couldn't be parsed.

GLOBUS_GRAM_PROTOCOL_ERROR_VERSION_MISMATCH The reply message was in an incompatible version of the GRAM protocol.

4.9.1.4 int globus_gram_protocol_unpack_status_reply (const globus_byte_t * *reply*, globus_size_t *replysize*, int * *job_status*, int * *failure_code*, int * *job_failure_code*)

Unpack a reply to a GRAM status request

Extracts the parameters of a reply to a status request from a GRAM message. The parameters to this function mirror those of globus_gram_protocol_pack_status_reply().

Parameters:

reply The job request reply.

replysize The length of the reply string.

job_status A pointer to an integer to be populated with thejob's current job state.

failure_code A pointer to an integer to be populated with the failure code associated with the status request. This may be GLOBUS_SUCCESS, if the job request was successful.

job_failure_code A pointer to an integer to be populated with the failure code for the job, if the *job_status* is GLOBUS_GRAM_PROTOCOL_JOB_STATE_FAILED.

Return values:

GLOBUS_SUCCESS The reply was successfully unpacked.

GLOBUS_GRAM_PROTOCOL_ERROR_HTTP_UNPACK_FAILED The reply message couldn't be parsed.

GLOBUS_GRAM_PROTOCOL_ERROR_VERSION_MISMATCH The reply message was in an incompatible version of the GRAM protocol.

4.9.1.5 int globus_gram_protocol_unpack_status_update_message (const globus_byte_t * *reply*, globus_size_t *replysize*, char ** *job_contact*, int * *status*, int * *failure_code*)

Unpack a status update message

Extracts the parameters of a status update from a GRAM message. The parameters to this function mirror those of globus_gram_protocol_pack_status_update_message().

Parameters:

reply The status update message.

replysize The length of the message.

job_contact An output variable which will be populated with a new string containing the job contact string. The caller must free this memory by calling globus_libc_free().

status A pointer to an integer to be populated with the job's current job state.

failure_code A pointer to an integer to be populated with the failure code for the job, if the *job_status* is GLOBUS_GRAM_PROTOCOL_JOB_STATE_FAILED.

Return values:

GLOBUS_SUCCESS The message was successfully unpacked.

GLOBUS_GRAM_PROTOCOL_ERROR_HTTP_UNPACK_FAILED The message couldn't be parsed.

GLOBUS_GRAM_PROTOCOL_ERROR_MALLOC_FAILED Memory for the *job_contact* string could not be allocated.

GLOBUS_GRAM_PROTOCOL_ERROR_VERSION_MISMATCH The status message was in an incompatible version of the GRAM protocol.

Index