

GT 5.0.0 Component Guide to Public Interfaces: GRAM5

GT 5.0.0 Component Guide to Public Interfaces: GRAM5

Table of Contents

1. APIs	1
1. Programming Model Overview	1
2. RSL Specification v1.1	2
1. RSL Syntax Overview	2
2. RSL Tokenization Overview	3
3. RSL Substitution Semantics	4
4. RSL Attribute Summary	4
5. Simple RSL Examples	7
6. RSL grammar and tokenization rules	8
I. GRAM5 Commands	10
globusrun	11
globus-job-cancel	15
globus-job-clean	16
globus-job-get-output	17
globus-job-run	19
globus-job-status	22
globus-job-submit	24
globus-personal-gatekeeper	27
globus-gram-audit	29
globus-gatekeeper	30
globus-job-manager	33
globus-job-manager-event-generator	38
globus-fork-starter	39
3. Job description	41
4. Semantics and syntax of protocols	42
1. GRAM5 Protocol	42
A. Errors	50
Glossary	60

List of Figures

4.1. GRAM State Transitions	49
-----------------------------------	----

List of Tables

1.1. GRAM Client APIs	1
4.1. GRAM Job States	49
A.1. GRAM5 Errors	51

List of Examples

2.1. Quoted Literal Examples	4
2.2. GRAM5 Job Request Examples	8

Chapter 1. APIs

1. Programming Model Overview

1.1. C API Documentation Links

Table 1.1. GRAM Client APIs

Name	Purpose
<u>GRAM Protocol</u> ¹	Low-level functions for processing GRAM protocol messages. Symbolic constants for RSL attributes, signals, and job states.
<u>GRAM Client</u> ²	Functions for submitting job requests, sending signals, and listening for job state updates.
<u>RSL</u> ³	Functions for parsing and manipulating job specifications in the RSL language.

¹ http://www.globus.org/api/c-globus-5.0.0/globus_gram_protocol/html/main.html

² http://www.globus.org/api/c-globus-5.0.0/globus_gram_client/html/main.html

³ http://www.globus.org/api/c-globus-5.0.0/globus_rsl/html/main.html

Chapter 2. RSL Specification v1.1

This is a document to specify the existing RSL v1.0 implementation and interfaces, as they are provided in the GT 5.0.0 release. This document serves as a reference, and more introductory text.

The Globus Resource Specification Language (RSL) provides a common interchange language to describe resources. The various components of the Globus Resource Management architecture manipulate RSL strings to perform their management functions in cooperation with the other components in the system. The RSL provides the skeletal syntax used to compose complicated resource descriptions, and the various resource management components introduce specific *ATTRIBUTE,VALUE*> pairings into this common structure. Each attribute in a resource description serves as a parameter to control the behavior of one or more components in the resource management system.

1. RSL Syntax Overview

The core syntax of the RSL syntax is the *relation*. Relations associate an attribute name with a value, eg the relation `executable=a.out` provides the name of an executable in a resource request. There are two generative syntactic structures in the RSL that are used to build more complicated resource descriptions out of the basic relations: *compound requests* and *value sequences*. In addition, the RSL syntax includes a facility to both introduce and dereference string *substitution variables*.

The simplest form of compound request, utilized by all resource management components, is the conjunct-request. The conjunct-request expresses a conjunction of simple relations or compound requests (like a boolean AND). The most common conjunct-request in Globus RSL strings is the combination of multiple relations such as executable name, node count, executable arguments, and output files for a basic GRAM job request. Similarly, the core RSL syntax includes a disjunct-request form to represent disjunctive relations (like a boolean OR). Currently, however, no resource management component utilizes the disjunct-request form.

The last form of compound request is the multi-request. The multi-request expresses multiple parallel resources that make up a resource description. The multi-request form differs from the conjunction and disjunction in two ways: multi-requests introduce new variable scope, meaning variables defined in one clause of a multi-request are not visible to the other clauses, and multi-requests introduce a non-reducible hierarchy to the resource description. Whereas relations within a conjunct-request can be thought of as *constraints* on the resource being described, the subclauses of a multi-request are best thought of as individual resource descriptions that together constitute an abstract resource collection; the same attributes may be *constrained* in different ways in each subclause without causing a logical contradiction. An example of a contradiction would be to constrain the `executable` attribute to be two conflicting values within a conjunction. Currently, however, no resource management component utilizes the disjunct-request form.

The simplest form of value in the RSL syntax is the string literal. When explicitly quoted, literals can contain any character, and many common literals that don't contain special characters can appear without quotes. Values can also be variable references, in which case the variable reference is in essence *replaced* with the string value defined for that variable. RSL descriptions can also express string-concatenation of values, especially useful to construct long strings out of several variable references. String concatenation is supported with both an explicit concatenation operator and implicit concatenation for many idiomatic constructions involving variable references and literals.

In addition to the simple value forms given above, the RSL syntax includes the value sequence to express ordered sets of values. The value sequence syntax is used primarily for defining variables and for providing the argument list for a program.

2. RSL Tokenization Overview

Each RSL string consists of a sequence of RSL tokens, whitespace, and comments. The RSL tokens are either special syntax or regular unquoted literals, where special syntax contains one or more of the following listed special characters and unquoted literals are made of sequences of characters excluding the special characters.

The complete set of special characters that cannot appear as part of an unquoted literal is:

- + (plus)
- & (ampersand)
- | (pipe)
- ((left paren)
-) (right paren)
- = (equal)
- < (left angle)
- > (right angle)
- ! (exclamation)
- " (double quote)
- ' (apostrophe)
- ^ (carat)
- # (pound)
- \$ (dollar)

These characters can only be used for the special syntactic forms described in the section and in the section or as within quoted literals.

Quoted literals are introduced with the " (double quote) or ' (single quote/apostrophe) and consist of all the characters up to (but not including) the next solo double or single quote, respectively. To escape a quote character within a quoted literal, the appearance of the quote character twice in a row is converted to a single instance of the character and the literal continues until the next solo quote character. For any quoted literal, there is only one possible escape sequence, eg within a literal delimited by the single quote character only the single quote character uses the escape notation and the double quote character can appear without escape.

Quoted literals can also be introduced with an alternate *user delimiter* notation. User delimited literals are introduced with the ^ (carat) character followed immediately by a user-provided delimiter; the literal consists of all the characters after the user's delimiter up to (but not including) the next solo instance of the delimiter. The delimiter itself may be escaped within the literal by providing two instances in a row, just as the regular quote delimiters are escaped in regular quoted literals.

RSL string comments use a notation similar to comments in the C programming language. Comments are introduced by the prefix (*. Comments continue to the first terminating suffix *) and cannot be nested. Comments are stripped from the RSL string during processing and are syntactically equivalent to whitespace.

Example 2.1. Quoted Literal Examples

Assign the value `Hello. Welcome to "The Grid"` to the attribute `arguments`, using double-quote as the delimiter and the escaping sequence.

```
arguments = "Hello. Welcome to \"The Grid\""
```

Assign the value `Hello. Welcome to "The Grid"` to the attribute `arguments` using the single-quote delimiter.

```
arguments = 'Hello. Welcome to "The Grid"'
```

Assign the value `Hello. Welcome to "The Grid"` to the attribute `arguments` using a user-defined quoting character `!`.

```
arguments = ^!Hello. Welcome to "The Grid"!
```

3. RSL Substitution Semantics

RSL strings can introduce and reference string variables. String substitution variables are defined in a special relation using the `rsl_substitution` attribute, and the definitions affect variable references made in the same conjunct-request (or disjunct-request), as well as references made within any multi-request nested inside one of the clauses of the conjunction (or disjunction). Each multi-request introduces a new variable scope for each subrequest, and variable definitions do not escape the closest enclosing scope.

Within any given scope, variable definitions are processed left-to-right in the resource description. Outermost scopes are processed before inner scopes, and the definitions in inner scopes augment the inherited definitions with new and/or updated variable definitions.

Variable definitions and variable references are processed in a single pass, with each definition updating the *environment* prior to processing the next definition. The value provided in a variable definition may include a reference to a previously-defined variable. References to variables that are not yet provided with definitions in the standard RSL variable processing order are replaced with an empty literal string.

4. RSL Attribute Summary

The RSL syntax is extensible because it defines structure without too many keywords. Each Globus resource management component introduces additional attributes to the set recognized by RSL-aware components, so it is difficult to provide a complete listing of attributes which might appear in a resource description. Resource management components are designed to utilize attributes they recognize and pass unrecognized relations through unchanged. This allows powerful compositions of different resource management functions.

The following listing summarizes the attribute names utilized by existing resource management components in the standard Globus release. Please see the individual component documentation for discussion of the attribute semantics.

4.1. GRAM5 Common RSL Attributes

<code>arguments</code>	The command line arguments for the executable. Use quotes, if a space is required in a single argument.
<code>count</code>	The number of executions of the executable.
<code>directory</code>	Specifies the path of the directory the jobmanager will use as the default directory for the requested job.

<code>dry_run</code>	If <code>dryrun = yes</code> then the jobmanager will not submit the job for execution and will return success.
<code>environment</code>	The environment variables that will be defined for the executable in addition to default set that is given to the job by the jobmanager.
<code>executable</code>	The name of the executable file to run on the remote machine. If the value is a GASS URL, the file is transferred to the remote gass cache before executing the job and removed after the job has terminated.
<code>file_clean_up</code>	Specifies a list of files which will be removed after the job is completed.
<code>file_stage_in</code>	Specifies a list of ("remote URL" "local file") pairs which indicate files to be staged to the nodes which will run the job.
<code>file_stage_in_shared</code>	Specifies a list of ("remote URL" "local file") pairs which indicate files to be staged into the cache. A symlink from the cache to the "local file" path will be made.
<code>file_stage_out</code>	Specifies a list of ("local file" "remote URL") pairs which indicate files to be staged from the job to a GASS-compatible file server.
<code>gass_cache</code>	Specifies location to override the GASS cache location.
<code>gram_my_job</code>	Obsolete and ignored.
<code>host_count</code>	Only applies to clusters of SMP computers, such as newer IBM SP systems. Defines the number of nodes ("pizza boxes") to distribute the "count" processes across.
<code>job_type</code>	This specifies how the jobmanager should start the job. Possible values are single (even if the count > 1, only start 1 process or thread), multiple (start count processes or threads), mpi (use the appropriate method (e.g. mpirun) to start a program compiled with a vendor-provided MPI library. Program is started with count nodes), and condor (starts condor jobs in the "condor" universe.)
<code>library_path</code>	Specifies a list of paths to be appended to the system-specific library path environment variables.
<code>max_cpu_time</code>	Explicitly set the maximum cputime for a single execution of the executable. The units is in minutes. The value will go through an <code>atoi()</code> conversion in order to get an integer. If the GRAM scheduler cannot set cputime, then an error will be returned.
<code>max_memory</code>	Explicitly set the maximum amount of memory for a single execution of the executable. The units is in Megabytes. The value will go through an <code>atoi()</code> conversion in order to get an integer. If the GRAM scheduler cannot set <code>maxMemory</code> , then an error will be returned.
<code>max_time</code>	The maximum walltime or cputime for a single execution of the executable. Walltime or cputime is selected by the GRAM scheduler being interfaced. The units is in minutes. The value will go through an <code>atoi()</code> conversion in order to get an integer.
<code>max_wall_time</code>	Explicitly set the maximum walltime for a single execution of the executable. The units is in minutes. The value will go through an <code>atoi()</code> conversion in order to get an integer. If the GRAM scheduler cannot set walltime, then an error will be returned.
<code>min_memory</code>	Explicitly set the minimum amount of memory for a single execution of the executable. The units is in Megabytes. The value will go through an <code>atoi()</code> conversion in order to

	get an integer. If the GRAM scheduler cannot set minMemory, then an error will be returned.
project	Target the job to be allocated to a project account as defined by the scheduler at the defined (remote) resource.
proxy_timeout	Obsolete and ignored. Now a job-manager-wide setting.
queue	Target the job to a queue (class) name as defined by the scheduler at the defined (remote) resource.
remote_io_url	Writes the given value (a URL base string) to a file, and adds the path to that file to the environment through the GLOBUS_REMOTE_IO_URL environment variable. If this is specified as part of a job restart RSL, the job manager will update the file's contents. This is intended for jobs that want to access files via GASS, but the URL of the GASS server has changed due to a GASS server restart.
restart	Start a new job manager, but instead of submitting a new job, start managing an existing job. The job manager will search for the job state file created by the original job manager. If it finds the file and successfully reads it, it will become the new manager of the job, sending callbacks on status and streaming stdout/err if appropriate. It will fail if it detects that the old jobmanager is still alive (via a timestamp in the state file). If stdout or stderr was being streamed over the network, new stdout and stderr attributes can be specified in the restart RSL and the jobmanager will stream to the new locations (useful when output is going to a GASS server started by the client that's listening on a dynamic port, and the client was restarted). The new job manager will return a new contact string that should be used to communicate with it. If a jobmanager is restarted multiple times, any of the previous contact strings can be given for the restart attribute.
rsl_substitution	Specifies a list of values which can be substituted into other rsl attributes' values through the \$(SUBSTITUTION) mechanism.
save_state	Causes the jobmanager to save it's job state information to a persistent file on disk. If the job manager exits or is suspended, the client can later start up a new job manager which can continue monitoring the job.
scratch_dir	Specifies the location to create a scratch subdirectory in. A SCRATCH_DIRECTORY RSL substitution will be filled with the name of the directory which is created.
stderr	The name of the remote file to store the standard error from the job. If the value is a GASS URL, the standard error from the job is transferred dynamically during the execution of the job.
stderr_position	Specifies where in the file remote standard error streaming should be restarted from. Must be 0.
stdin	The name of the file to be used as standard input for the executable on the remote machine. If the value is a GASS URL, the file is transferred to the remote gass cache before executing the job and removed after the job has terminated.
stdout	The name of the remote file to store the standard output from the job. If the value is a GASS URL, the standard output from the job is transferred dynamically during the execution of the job.
stdout_position	Specifies where in the file remote output streaming should be restarted from. Must be 0.

two_phase

Use a two-phase commit for job submission and completion. The job manager will respond to the initial job request with a `WAITING_FOR_COMMIT` error. It will then wait for a signal from the client before doing the actual job submission. The integer supplied is the number of seconds the job manager should wait before timing out. If the job manager times out before receiving the commit signal, or if a client issues a cancel signal, the job manager will clean up the job's files and exit, sending a callback with the job status as `GLOBUS_GRAM_PROTOCOL_JOB_STATE_FAILED`. After the job manager sends a `DONE` or `FAILED` callback, it will wait for a commit signal from the client. If it receives one, it cleans up and exits as usual. If it times out and `save_state` was enabled, it will leave all of the job's files in place and exit (assuming the client is down and will attempt a job restart later). The timeoutvalue can be extended via a signal. When one of the following errors occurs, the job manager does not delete the job state file when it exits: `GLOBUS_GRAM_PROTOCOL_ERROR_COMMIT_TIMED_OUT`, `GLOBUS_GRAM_PROTOCOL_ERROR_TTL_EXPIRED`, `GLOBUS_GRAM_PROTOCOL_ERROR_JM_STOPPED`, `GLOBUS_GRAM_PROTOCOL_ERROR_USER_PROXY_EXPIRED`. In these cases, it can not be restarted, so the job manager will not wait for the commit signal after sending the `FAILED` callback

username

Verify that the job is running as this user.

5. Simple RSL Examples

The following are some simple example RSL strings to illustrate idiomatic usage with existing tools and to make concrete some of the more interesting cases of tokenization, concatenation, and variable semantics. These are meant to illustrate the use of the RSL notation without much regard for the specific details of a particular resource management component.

Typical GRAM5 resource descriptions contain at least a few relations in a conjunction:

Example 2.2. GRAM5 Job Request Examples

This example shows a conjunct request containing values that are unquoted literals and ordered sequences of a mix of quoted and unquoted literals.

```
(* this is a comment *)
& (executable = a.out (* <-- that is an unquoted literal *))
  (directory = /home/nobody )
  (arguments = arg1 "arg 2")
  (count = 1)
```

This example demonstrates RSL substitutions, which can be used to make sure a string is used consistently multiple times in a resource description:

```
& (rsl_substitution = (TOPDIR "/home/nobody")
                        (DATADIR "${TOPDIR}/data")
                        (EXECDIR "${TOPDIR}/bin") )
  (executable = $(EXECDIR)/a.out
    (* ^-- implicit concatenation *))
  (directory = $(TOPDIR) )
  (arguments = $(DATADIR)/file1
    (* ^-- implicit concatenation *)
    ${DATADIR} # /file2
    (* ^-- explicit concatenation *)
    '$(FOO)' (* <-- a quoted literal *))
  (environment = (DATADIR $(DATADIR)))
  (count = 1)
```

Performing all variable substitution and removing comments yields an equivalent RSL string:

```
& (rsl_substitution = (TOPDIR "/home/nobody")
                        (DATADIR "/home/nobody/data")
                        (EXECDIR "/home/nobody/bin") )
  (executable = "/home/nobody/bin/a.out" )
  (directory = "/home/nobody" )
  (arguments = "/home/nobody/data/file1"
    "/home/nobody/data/file2"
    "$(FOO)" )
  (environment = (DATADIR "/home/nobody/data"))
  (count = 1)
```

Note in the above variable-substitution example, the variable substitution definitions are not automatically made a part of the job's environment. And explicit `environment` attribute must be used to add environment variables for the job. Also note that the third value in the arguments clause is not a variable reference but only quoted literal that happens to contain one of the special characters.

6. RSL grammar and tokenization rules

The following is a modified BNF grammar for the Resource Specification Language. Lexical rules are provided for the implicit concatenation sequences in the form of conventional regular expressions; for the *implicit-concat* non-terminal rules, whitespace is not allowed between juxtaposed non-terminals. Grammar comments are provided in square

brackets in a column to the right of the productions, eg [comment] to help relate productions in the grammar to the terminology used in the above discussion.

Regular expressions are provided for the terminal class `string-literal` and for RSL comments. These regular expressions make use of a common inverted character-class notation, as popularized by the various lex tools. Comments are syntactically equivalent to whitespace and can only appear where the comment prefix cannot be mistaken for the trailing part of a multi-character unquoted literal.

RSL Grammar

- [1] `specification ::= relation` /* relation */
| '+' `spec-list` /* multi-re-
| '&' `spec-list` quest */
| '|' `spec-list` /* conjunct-re-
| '&' `spec-list` quest */
| '|' `spec-list` /* disjunct-request */
- [2] `spec-list ::= '(' specification ')' spec-list`
| '(' `specification` ')'
- [3] `relation ::= 'rsl_substitution' '=' binding-sequence` /* Substitution variable
| `attribute op value-sequence` definition */
| `attribute op value-sequence` /* Attribute
| `attribute op value-sequence` binding relation */
- [4] `binding-sequence ::= binding binding-sequence`
| `binding`
- [5] `binding ::= '(' string-literal simple-value ')'` /* Substitution variable
| `string-literal` `simple-value` ')' definition */
| `string-literal` `simple-value` ')' /* attribute */
- [6] `attribute ::= string-literal`
- [7] `op ::= '=' | '!=' | '>' | '>=' | '<' | '<='`
- [8] `value-sequence ::= value value-sequence | value`
- [9] `value ::= '(' value-sequence ')' | simple-value`
- [10] `simple-value ::= string-literal` /* String */
| `simple-value` '#' `simple-value` /* Concatena-
| `implicit-concat` | `variable-reference` tion */
- [11] `variable-reference ::= '$(' string-literal ')'` /* Variable Reference */
- [12] `implicit-concat ::= (unquoted-literal)? (implicit-concat-core)+` /* Implicit concatenation */
- [13] `implicit-concat- ::= variable-reference`
core | (`variable-reference`) (`unquoted-literal`)
- [14] `string-literal ::= quoted-literal`
| `unquoted-literal`
- [15] `quoted-literal ::= ''' (([^\']) | ('''))* '''` /* Single-quote delimiter with
| ''' (([^\']) | ('''))* ''' escaping */
| '^' c(([^c]|(cc))* c /* Double-quote
| '^' c(([^c]|(cc))* c delimiter with escap-
| '^' c(([^c]|(cc))* c ing */
| '^' c(([^c]|(cc))* c /* User defined delim-
| '^' c(([^c]|(cc))* c iter c with escaping */
| '^' c(([^c]|(cc))* c /* Non-special characters */
- [16] `unquoted-literal ::= ([^\t\v\n+&|()=<>!'\"#$%]+)` /* Non-special characters */
- [17] `comment ::= '(* (([^*]) | ('* [^])) *)'` /* Comment */

GRAM5 Commands

Name

globusrun -- Execute and manage jobs via GRAM

```
globusrun [-help] [-usage] [-version] [-versions]
globusrun { -p | -parse }
{ -f RSL_FILENAME | -file RSL_FILENAME | RSL_SPECIFICATION }
globusrun [-n] [-no-interrupt]
{ -r RESOURCE_CONTACT | -resource RESOURCE_CONTACT }
{ -a | -authenticate-only }
globusrun [-n] [-no-interrupt]
{ -r RESOURCE_CONTACT | -resource RESOURCE_CONTACT }
{ -j | -jobmanager-version }
globusrun [-n] [-no-interrupt] { -k | -kill } { JOB_ID }
globusrun [-n] [-no-interrupt] [-full-proxy] [-D] { -y | -refresh-proxy } { JOB_ID }
globusrun { -status } { JOB_ID }
globusrun [-q] [-quiet] [-o] [-output-enable] [-s] [-server] [-w] [-write-allow] [-n] [-no-interrupt] [-b] [-batch] [-F]
[-fast-batch] [-full-proxy] [-D] [-d] [-dryrun]
{ -r RESOURCE_CONTACT | -resource RESOURCE_CONTACT }
{ -f RSL_FILENAME | -file RSL_FILENAME | RSL_SPECIFICATION }
```

Description

The **globusrun** program for submits and manages jobs run on a local or remote job host. The jobs are controlled by the **globus-job-manager** program which interfaces with a local resource manager that schedules and executes the job.

The **globusrun** program can be run in a number of different modes chosen by command-line options.

When `-help`, `-usage`, `-version`, or `-versions` command-line options are used, **globusrun** will print out diagnostic information and then exit.

When the `-p` or `-parse` command-line option is present, **globusrun** will verify the syntax of the RSL specification and then terminate. If the syntax is valid, **globusrun** will print out the string "RSL Parsed Successfully..." and exit with a zero exit code; otherwise, it will print an error message and terminate with a non-zero exit code.

When the `-a` or `-authenticate-only` command-line option is present, **globusrun** will verify that the service named by *RESOURCE_CONTACT* exists and the client's credentials are granted permission to access that service. If authentication is successful, **globusrun** will display the string "GRAM Authentication test successful" and exit with a zero exit code; otherwise it will print an explanation of the problem and will with a non-zero exit code.

When the `-j` or `-jobmanager-version` command-line option is present, **globusrun** will attempt to determine the software version that the service named by *RESOURCE_CONTACT* is running. If successful, it will display both the Toolkit version and the Job Manager package version and exit with a zero exit code; otherwise, it will print an explanation of the problem and exit with a non-zero exit code.

When the `-k` or `-kill` command-line option is present, **globusrun** will attempt to terminate the job named by *JOB_ID*. If successful, **globusrun** will exit with zero; otherwise it will display an explanation of the problem and exit with a non-zero exit code.

When the `-y` or `-refresh-proxy` command-line option is present, **globusrun** will attempt to delegate a new X.509 proxy to the job manager which is managing the job named by *JOB_ID*. If successful, **globusrun** will exit with zero; otherwise it will display an explanation of the problem and exit with a non-zero exit code. This behavior can be modified by the `-full-proxy` or `-D` command-line options to enable full proxy delegation. The default is limited proxy delegation.

When the `-status` command-line option is present, **globusrun** will attempt to determine the current state of the job. If successful, the state will be printed to standard output and **globusrun** will exit with a zero exit code; otherwise, a description of the error will be displayed and it will exit with a non-zero exit code.

Otherwise, **globusrun** will submit the job to a GRAM service. By default, **globusrun** waits until the job has terminated or failed before exiting, displaying information about job state changes and at exit time, the job exit code if it is provided by the GRAM service.

The **globusrun** program can also function as a GASS file server to allow the **globus-job-manager** program to stage files to and from the machine on which **globusrun** is executed to the GRAM service node. This behavior is controlled by the `-s`, `-o`, and `-w` command-line options.

Jobs submitted by **globusrun** can be monitored interactively or detached. To have **globusrun** detach from the GRAM service after submitting the job, use the `-b` or `-F` command-line options.

Options

The full set of options to **globusrun** consist of:

<code>-help</code>	Display a help message to standard error and exit.
<code>-usage</code>	Display a one-line usage summary to standard error and exit.
<code>-version</code>	Display the software version of globusrun to standard error and exit.
<code>-versions</code>	Display the software version of all modules used by globusrun (including DiRT information) to standard error and then exit.
<code>-p, -parse</code>	Do a parse check on the job specification and print diagnostics. If a parse error occurs, globusrun exits with a non-zero exit code.
<code>-f RSL_FILENAME, -file RSL_FILENAME</code>	Read job specification from the file named by <i>RSL_FILENAME</i> .
<code>-n, -no-interrupt</code>	Disable handling of the SIGINT signal, so that the interrupt character (typically Control-C) causes globusrun to terminate without canceling the job.
<code>-r RESOURCE_CONTACT, -resource RESOURCE_CONTACT</code>	Submit the request to the resource specified by <i>RESOURCE_CONTACT</i> . A resource may be specified in the following ways: <ul style="list-style-type: none"> • <i>HOST</i> • <i>HOST:PORT</i> • <i>HOST:PORT/SERVICE</i> • <i>HOST/SERVICE</i> • <i>HOST:/SERVICE</i> • <i>HOST::SUBJECT</i> • <i>HOST:PORT:SUBJECT</i> • <i>HOST/SERVICE:SUBJECT</i>

- *HOST:/SERVICE:SUBJECT*
- *HOST:PORT/SERVICE:SUBJECT*

If any of *PORT*, *SERVICE*, or *SUBJECT* is omitted, the defaults of 2811, jobmanager, and host@HOST are used respectively.

-j, -jobmanager-version	Print the software version being run by the service running at <i>RESOURCE_CONTACT</i> .
-k <i>JOB_ID</i> , -kill <i>JOB_ID</i>	Kill the job named by <i>JOB_ID</i>
-D, -full-proxy	Delegate a full impersonation proxy to the service. By default, a limited proxy is delegated when needed.
-y, -refresh-proxy	Delegate a new proxy to the service processing <i>JOB_ID</i> .
-status	Display the current status of the job named by <i>JOB_ID</i> .
-q, -quiet	Do not display job state change or exit code information.
-o, -output-enable	Start a GASS server within the globusrun application that allows access to its standard output and standard error streams only. Also, augment the <i>RSL_SPECIFICATION</i> with a definition of the GLOBUSRUN_GASS_URL RSL substitution and add <code>stdout</code> and <code>stderr</code> clauses which redirect the output and error streams of the job to the output and error streams of the interactive globusrun command. If this is specified, then globusrun acts as though the <code>-q</code> were also specified.
-s, -server	Start a GASS server within the globusrun application that allows access to its standard output and standard error streams for writing and any file local the the globusrun invocation for reading. Also, augment the <i>RSL_SPECIFICATION</i> with a definition of the GLOBUSRUN_GASS_URL RSL substitution and add <code>stdout</code> and <code>stderr</code> clauses which redirect the output and error streams of the job to the output and error streams of the interactive globusrun command. If this is specified, then globusrun acts as though the <code>-q</code> were also specified.
-w, -write-allow	Start a GASS server within the globusrun application that allows access to its standard output and standard error streams for writing and any file local the the globusrun invocation for reading or writing. Also, augment the <i>RSL_SPECIFICATION</i> with a definition of the GLOBUSRUN_GASS_URL RSL substitution and add <code>stdout</code> and <code>stderr</code> clauses which redirect the output and error streams of the job to the output and error streams of the interactive globusrun command. If this is specified, then globusrun acts as though the <code>-q</code> were also specified.
-b, -batch	Terminate after submitting the job to the GRAM service. The globusrun program will exit after the job hits any of the following states: PENDING, ACTIVE, FAILED, or DONE. The GASS-related options can be used to stage input files, but standard output, standard error, and file staging after the job completes will not be processed.
-F, -fast-batch	Terminate after submitting the job to the GRAM service. The globusrun program will exit after it receives a reply from the service. The <i>JOB_ID</i> will be displayed to standard output before terminating so that the job can be checked with the

`-status` command-line option or modified by the `-refresh-proxy` or `-kill` command-line options.

`-d, -dryrun`

Submit the job with the `dryrun` attribute set to true. When this is done, the job manager will prepare to start the job but start short of submitting it to the service. This can be used to detect problems with the `RSL_SPECIFICATION`.

Environment

If the following variables affect the execution of **globusrun**

`X509_USER_PROXY` Path to proxy credential.

`X509_CERT_DIR` Path to trusted certificate directory.

Bugs

The **globusrun** program assumes any failure to contact the job means the job has terminated. In fact, this may be due to the **globus-job-manager** program exiting after all jobs it is managing have reached the `DONE` or `FAILED` states. In order to reliably detect job termination, the `two_phase` RSL attribute should be used.

See Also

`globus-job-submit(1)`, `globus-job-run(1)`, `globus-job-clean(1)`, `globus-job-get-output(1)`, `globus-job-cancel(1)`

Name

`globus-job-cancel --` Cancel a GRAM batch job

```
globus-job-cancel [ -f | -force ] [ -q | -quiet ] JOBID  
globus-job-cancel [-help] [-usage] [-version] [-versions]
```

Description

The **globus-job-cancel** program cancels the job named by *JOBID*. Any cached files associated with the job will remain until **globus-job-clean** is executed for the job.

By default, **globus-job-cancel** prompts the user prior to canceling the job. This behavior can be overridden by specifying the `-f` or `-force` command-line options.

Options

The full set of options to **globus-job-cancel** are:

- `-help,` Display a help message to standard error and exit.
- `-usage`

- `-version` Display the software version of the **globus-job-cancel** program to standard output.

- `-version` Display the software version of the **globus-job-cancel** program including DiRT information to standard output.

- `-force,` Do not prompt to confirm job cancel and clean-up.
- `-f`

- `-quiet,` Do not print diagnostics for succesful cancel. Implies `-f`
- `-q`

ENVIRONMENT

If the following variables affect the execution of **globus-job-cancel**.

- `X509_USER_PROXY` Path to proxy credential.

- `X509_CERT_DIR` Path to trusted certificate directory.

Name

globus-job-clean -- Cancel and clean up a GRAM batch job

```
globus-job-clean [ -r RESOURCE | -resource RESOURCE ]  
[ -f | -force ] [ -q | -quiet ] JOBID  
globus-job-clean [-help] [-usage] [-version] [-versions]
```

Description

The **globus-job-clean** program cancels the job named by *JOBID* if it is still running, and then removes any cached files on the GRAM service node related to that job. In order to do the file clean up, it submits a job which removes the cache files. By default this cleanup job is submitted to the default GRAM resource running on the same host as the job. This behavior can be controlled by specifying a resource manager contact string as the parameter to the `-r` or `-resource` option.

By default, **globus-job-clean** prompts the user prior to canceling the job. This behavior can be overridden by specifying the `-f` or `-force` command-line options.

Options

The full set of options to **globus-job-clean** are:

<code>-help, -usage</code>	Display a help message to standard error and exit.
<code>-version</code>	Display the software version of the globus-job-clean program to standard output.
<code>-version</code>	Display the software version of the globus-job-clean program including DiRT information to standard output.
<code>-resource RESOURCE,</code> <code>-r RESOURCE</code>	Submit the clean-up job to the resource named by <i>RESOURCE</i> instead of the default GRAM service on the same host as the job contact.
<code>-force, -f</code>	Do not prompt to confirm job cancel and clean-up.
<code>-quiet, -q</code>	Do not print diagnostics for succesful clean-up. Implies <code>-f</code>

ENVIRONMENT

If the following variables affect the execution of **globus-job-clean**.

<code>X509_USER_PROXY</code>	Path to proxy credential.
<code>X509_CERT_DIR</code>	Path to trusted certificate directory.

Name

`globus-job-get-output --` Retrieve the output and error streams from a GRAM job

```
globus-job-get-output [ -r RESOURCE | -resource RESOURCE ]  
[ -out | -err ] [ -t LINES | -tail LINES ] [ -follow LINES | -f LINES ] JOBID  
globus-job-get-output [-help] [-usage] [-version] [-versions]
```

Description

The **globus-job-get-output** program retrieves the output and error streams of the job named by *JOBID*. By default, **globus-job-get-output** will retrieve all output and error data from the job and display them to its own output and error streams. Other behavior can be controlled by using command-line options. The data retrieval is implemented by submitting another job which simply displays the contents of the first job's output and error streams. By default this retrieval job is submitted to the default GRAM resource running on the same host as the job. This behavior can be controlled by specifying a particular resource manager contact string as the *RESOURCE* parameter to the `-r` or `-resource` option.

Options

The full set of options to **globus-job-get-output** are:

<code>-help, -usage</code>	Display a help message to standard error and exit.
<code>-version</code>	Display the software version of the globus-job-get-output program to standard output.
<code>-version</code>	Display the software version of the globus-job-get-output program including DiRT information to standard output.
<code>-resource <i>RESOURCE</i>,</code> <code>-r <i>RESOURCE</i></code>	Submit the retrieval job to the resource named by <i>RESOURCE</i> instead of the default GRAM service on the same host as the job contact.
<code>-out</code>	Retrieve only the standard output stream of the job. The default is to retrieve both standard output and standard error.
<code>-err</code>	Retrieve only the standard error stream of the job. The default is to retrieve both standard output and standard error.
<code>-tail <i>LINES</i>, -t</code> <code><i>LINES</i></code>	Print only the last <i>LINES</i> count lines of output from the data streams being retrieved. By default, the entire output and error file data is retrieved. This option can not be used along with the <code>-f</code> or <code>-follow</code> options.
<code>-follow <i>LINES</i>, -f</code> <code><i>LINES</i></code>	Print the last <i>LINES</i> count lines of output from the data streams being retrieved and then wait until canceled, printing any subsequent job output that occurs. By default, the entire output and error file data is retrieved. This option can not be used along with the <code>-t</code> or <code>-tail</code> options.

ENVIRONMENT

If the following variables affect the execution of **globus-job-get-output**.

`X509_USER_PROXY` Path to proxy credential.

X509_CERT_DIR Path to trusted certificate directory.

BUGS

The `-f` and `-follow` don't work in GRAM5.

Name

globus-job-run -- Execute a job using GRAM

```
globus-job-run [-dumprsl] [-dryrun] [-verify]
[-file ARGUMENT_FILE]
SERVICE_CONTACT
[ -np PROCESSES | -count PROCESSES ]
[ -m MAX_TIME | -maxtime MAX_TIME ]
[ -p PROJECT | -project PROJECT ]
[ -q QUEUE | -queue QUEUE ]
[ -d DIRECTORY | -directory DIRECTORY ] [-env NAME=VALUE]...
[-stdin [ -l | -s ] STDIN_FILE ] [-stdout [ -l | -s ] STDOUT_FILE ] [-stderr [ -l | -s ] STDERR_FILE ]
[-x RSL_CLAUSE]
[ -l | -s ] EXECUTABLE [ARGUMENT...]
globus-job-run [-help] [-usage] [-version] [-versions]
```

Description

The **globus-job-run** program constructs a job description from its command-line options and then submits the job to the GRAM service running at *SERVICE_CONTACT*. The executable and arguments to the executable are provided on the command-line after all other options. Note that the `-dumprsl`, `-dryrun`, `-verify`, and `-file` command-line options must occur before the first non-option argument, the *SERVICE_CONTACT*.

The **globus-job-run** provides similar functionality to **globusrun** in that it allows interactive start-up of GRAM jobs. However, unlike **globusrun**, it uses command-line parameters to define the job instead of RSL expressions.

Options

The full set of options to **globus-job-run** are:

<code>-help</code> , <code>-usage</code>	Display a help message to standard error and exit.
<code>-version</code>	Display the software version of the globus-job-run program to standard output.
<code>-version</code>	Display the software version of the globus-job-run program including DiRT information to standard output.
<code>-dumprsl</code>	Translate the command-line options to globus-job-run into an RSL expression that can be used with tools such as globusrun .
<code>-dryrun</code>	Submit the job request to the GRAM service with the <code>dryrun</code> option enabled. When this option is used, the GRAM service prepares to execute the job but stops before submitting the job to the LRM. This can be used to diagnose some problems such as missing files.
<code>-verify</code>	Submit the job request to the GRAM service with the <code>dryrun</code> option enabled and then without it enabled if the <code>dryrun</code> is successful.
<code>-file ARGUMENT_FILE</code>	Read additional command-line options from <i>ARGUMENT_FILE</i> .
<code>-np PROCESSES</code> , <code>-count PRO-</code> <code>CESSSES</code>	Start <i>PROCESSES</i> instances of the executable as a single job.

<code>-m MAX_TIME, -maxtime MAX_TIME</code>	Schedule the job to run for a maximum of <i>MAX_TIME</i> minutes.
<code>-p PROJECT, -project PRO- JECT</code>	Request that the job use the allocation <i>PROJECT</i> when submitting the job to the LRM.
<code>-q QUEUE, -queue QUEUE</code>	Request that the job be submitted to the LRM using the named <i>QUEUE</i> .
<code>-d DIRECTORY, -directory DIRECTORY</code>	Run the job in the directory named by <i>DIRECTORY</i> . Input and output files will be interpreted relative to this directory. This directory must exist on the file system on the LRM-managed resource. If not specified, the job will run in the home directory of the user the job is running as.
<code>-env NAME=VALUE</code>	Define an environment variable named by <i>NAME</i> with the value <i>VALUE</i> in the job environment. This option may be specified multiple times to define multiple environment variables.
<code>-stdin [-l -s] STDIN_FILE</code>	Use the file named by <i>STDIN_FILE</i> as the standard input of the job. If the <code>-l</code> option is specified, then this file is interpreted to be on a file system local to the LRM. If the <code>-s</code> option is specified, then this file is interpreted to be on the file system where globus-job-run is being executed, and the file will be staged via GASS. If neither is specified, the local behavior is assumed.
<code>-stdout [-l -s] STDOUT_FILE</code>	Use the file named by <i>STDOUT_FILE</i> as the destination for the standard output of the job. If the <code>-l</code> option is specified, then this file is interpreted to be on a file system local to the LRM. If the <code>-s</code> option is specified, then this file is interpreted to be on the file system where globus-job-run is being executed, and the file will be staged via GASS. If neither is specified, the local behavior is assumed.
<code>-stderr [-l -s] STDERR_FILE</code>	Use the file named by <i>STDERR_FILE</i> as the destination for the standard error of the job. If the <code>-l</code> option is specified, then this file is interpreted to be on a file system local to the LRM. If the <code>-s</code> option is specified, then this file is interpreted to be on the file system where globus-job-run is being executed, and the file will be staged via GASS. If neither is specified, the local behavior is assumed.
<code>-x RSL_CLAUSE</code>	Add a set of custom RSL attributes described by <i>RSL_CLAUSE</i> to the job description. The clause must be an RSL conjunction and may contain one or more attributes. This can be used to include attributes which can not be defined by other command-line options of globus-job-run .
<code>-l</code>	When included outside the context of <code>-stdin</code> , <code>-stdout</code> , or <code>-stderr</code> command-line options, <code>-l</code> option alters the interpretation of the executable path. If the <code>-l</code> option is specified, then the executable is interpreted to be on a file system local to the LRM.
<code>-s</code>	When included outside the context of <code>-stdin</code> , <code>-stdout</code> , or <code>-stderr</code> command-line options, <code>-s</code> option alters the interpretation of the executable path. If the <code>-s</code> option is specified, then the executable is interpreted to be on the file system where globus-job-run is being executed, and the file will be staged via GASS. If neither is specified, the local behavior is assumed.

ENVIRONMENT

If the following variables affect the execution of **globus-job-run**.

X509_USER_PROXY Path to proxy credential.

X509_CERT_DIR Path to trusted certificate directory.

See Also

globusrun(1), globus-job-submit(1), globus-job-clean(1), globus-job-get-output(1), globus-job-cancel(1)

Name

globus-job-status -- Check the status of a GRAM5 job

globus-job-status *JOBID*

globus-job-status [-help] [-usage] [-version] [-versions]

Description

The **globus-job-status** program checks the status of a GRAM job by sending a status request to the job manager contact for that job specified by the *JOBID* parameter. If successful, it will print the job status to standard output. The states supported by **globus-job-status** are:

PENDING	The job has been submitted to the LRM but has not yet begun execution.
ACTIVE	The job has begun execution.
FAILED	The job has failed.
SUSPENDED	The job is currently suspended by the LRM.
DONE	The job has completed.
UNSUBMITTED	The job has been accepted by GRAM, but not yet submitted to the LRM.
STAGE_IN	The job has been accepted by GRAM and is currently staging files prior to being submitted to the LRM.
STAGE_OUT	The job has completed execution and is currently staging files from the service node to other http, GASS, or GridFTP servers.

Options

The full set of options to **globus-job-status** are:

-help, -usage Display a help message to standard error and exit.

-version Display the software version of the **globus-job-status** program to standard output.

-versions Display the software version of the **globus-job-status** program including DiRT information to standard output.

ENVIRONMENT

If the following variables affect the execution of **globus-job-status**.

X509_USER_PROXY Path to proxy credential.

X509_CERT_DIR Path to trusted certificate directory.

Bugs

The **globus-job-status** program can not distinguish between the case of the job manager terminating for any reason and the job being in the DONE state.

See Also

globusrun(1)

Name

globus-job-submit -- Submit a batch job using GRAM

```
globus-job-submit [-dumprsl] [-dryrun] [-verify]
[-file ARGUMENT_FILE]
SERVICE_CONTACT
[ -np PROCESSES | -count PROCESSES ]
[ -m MAX_TIME | -maxtime MAX_TIME ]
[ -p PROJECT | -project PROJECT ]
[ -q QUEUE | -queue QUEUE ]
[ -d DIRECTORY | -directory DIRECTORY ] [-env NAME=VALUE]...
[-stdin [ -l | -s ] STDIN_FILE ] [-stdout [ -l | -s ] STDOUT_FILE ] [-stderr [ -l | -s ] STDERR_FILE ]
[-x RSL_CLAUSE]
[ -l | -s ] EXECUTABLE [ARGUMENT...]
globus-job-submit [-help] [-usage] [-version] [-versions]
```

Description

The **globus-job-submit** program constructs a job description from its command-line options and then submits the job to the GRAM service running at *SERVICE_CONTACT*. The executable and arguments to the executable are provided on the command-line after all other options. Note that the `-dumprsl`, `-dryrun`, `-verify`, and `-file` command-line options must occur before the first non-option argument, the *SERVICE_CONTACT*.

The **globus-job-submit** provides similar functionality to **globusrun** in that it allows batch submission of GRAM jobs. However, unlike **globusrun**, it uses command-line parameters to define the job instead of RSL expressions.

To retrieve the output and error streams of the job, use the program **globus-job-get-output**. To reclaim resources used by the job by deleting cached files and job state, use the program **globus-job-clean**. To cancel a batch job submitted by **globus-job-submit**, use the program **globus-job-cancel**.

Options

The full set of options to **globus-job-submit** are:

<code>-help, -usage</code>	Display a help message to standard error and exit.
<code>-version</code>	Display the software version of the globus-job-submit program to standard output.
<code>-versions</code>	Display the software version of the globus-job-submit program including DiRT information to standard output.
<code>-dumprsl</code>	Translate the command-line options to globus-job-submit into an RSL expression that can be used with tools such as globusrun .
<code>-dryrun</code>	Submit the job request to the GRAM service with the <code>dryrun</code> option enabled. When this option is used, the GRAM service prepares to execute the job but stops before submitting the job to the LRM. This can be used to diagnose some problems such as missing files.
<code>-verify</code>	Submit the job request to the GRAM service with the <code>dryrun</code> option enabled and then without it enabled if the <code>dryrun</code> is successful.

<code>-file ARGUMENT_FILE</code>	Read additional command-line options from <i>ARGUMENT_FILE</i> .
<code>-np PROCESSES, -count PROCESSES</code>	Start <i>PROCESSES</i> instances of the executable as a single job.
<code>-m MAX_TIME, -maxtime MAX_TIME</code>	Schedule the job to run for a maximum of <i>MAX_TIME</i> minutes.
<code>-p PROJECT, -project PROJECT</code>	Request that the job use the allocation <i>PROJECT</i> when submitting the job to the LRM.
<code>-q QUEUE, -queue QUEUE</code>	Request that the job be submitted to the LRM using the named <i>QUEUE</i> .
<code>-d DIRECTORY, -directory DIRECTORY</code>	Run the job in the directory named by <i>DIRECTORY</i> . Input and output files will be interpreted relative to this directory. This directory must exist on the file system on the LRM-managed resource. If not specified, the job will run in the home directory of the user the job is running as.
<code>-env NAME=VALUE</code>	Define an environment variable named by <i>NAME</i> with the value <i>VALUE</i> in the job environment. This option may be specified multiple times to define multiple environment variables.
<code>-stdin [-l -s] STDIN_FILE</code>	Use the file named by <i>STDIN_FILE</i> as the standard input of the job. If the <code>-l</code> option is specified, then this file is interpreted to be on a file system local to the LRM. If the <code>-s</code> option is specified, then this file is interpreted to be on the file system where globus-job-submit is being executed, and the file will be staged via GASS. If neither is specified, the local behavior is assumed.
<code>-stdout [-l -s] STDOUT_FILE</code>	Use the file named by <i>STDOUT_FILE</i> as the destination for the standard output of the job. If the <code>-l</code> option is specified, then this file is interpreted to be on a file system local to the LRM. If the <code>-s</code> option is specified, then this file is interpreted to be on the file system where globus-job-submit is being executed, and the file will be staged via GASS. If neither is specified, the local behavior is assumed.
<code>-stderr [-l -s] STDERR_FILE</code>	Use the file named by <i>STDERR_FILE</i> as the destination for the standard error of the job. If the <code>-l</code> option is specified, then this file is interpreted to be on a file system local to the LRM. If the <code>-s</code> option is specified, then this file is interpreted to be on the file system where globus-job-submit is being executed, and the file will be staged via GASS. If neither is specified, the local behavior is assumed.
<code>-x RSL_CLAUSE</code>	Add a set of custom RSL attributes described by <i>RSL_CLAUSE</i> to the job description. The clause must be an RSL conjunction and may contain one or more attributes. This can be used to include attributes which can not be defined by other command-line options of globus-job-submit .
<code>-l</code>	When included outside the context of <code>-stdin</code> , <code>-stdout</code> , or <code>-stderr</code> command-line options, <code>-l</code> option alters the interpretation of the executable path. If the <code>-l</code> option is specified, then the executable is interpreted to be on a file system local to the LRM.
<code>-s</code>	When included outside the context of <code>-stdin</code> , <code>-stdout</code> , or <code>-stderr</code> command-line options, <code>-s</code> option alters the interpretation of the executable path. If the <code>-s</code> option is specified, then the executable is interpreted to be on the file

system where **globus-job-run** is being executed, and the file will be staged via GASS. If neither is specified, the local behavior is assumed.

ENVIRONMENT

If the following variables affect the execution of **globus-job-submit**.

X509_USER_PROXY Path to proxy credential.

X509_CERT_DIR Path to trusted certificate directory.

See Also

globusrun(1), globus-job-run(1), globus-job-clean(1), globus-job-get-output(1), globus-job-cancel(1)

Name

`globus-personal-gatekeeper --` Manage a user's personal gatekeeper daemon

```
globus-personal-gatekeeper [-help] [-usage] [-version] [-versions] [-list] [-directory CONTACT]
globus-personal-gatekeeper [-debug] {-start} [-jmtime LRM] [-auditdir AUDIT_DIRECTORY] [-port PORT]
[-log [=DIRECTORY]] [-seg] [-acctfile ACCOUNTING_FILE]
globus-personal-gatekeeper [-killall] [-kill]
```

Description

The **globus-personal-gatekeeper** command is a utility which manages a gatekeeper and job manager service for a single user. Depending on the command-line arguments it will operate in one of several modes. In the first set of arguments indicated in the synopsis, the program provides information about the **globus-personal-gatekeeper** command or about instances of the **globus-personal-gatekeeper** that are running currently. The second set of arguments indicated in the synopsis provide control over starting a new **globus-personal-gatekeeper** instance. The final set of arguments provide control for terminating one or more **globus-personal-gatekeeper** instances.

The `-start` mode will create a new subdirectory of `$HOME/.globus` and write the configuration files needed to start a **globus-gatekeeper** daemon which will invoke the **globus-job-manager** service when new authenticated connections are made to its service port. The **globus-personal-gatekeeper** then exits, printing the contact string for the new gatekeeper prefixed by `GRAM contact:` to standard output. In addition to the arguments described above, any arguments described in **globus-job-manager(8)** can be appended to the command-line and will be added to the job manager configuration for the service started by the **globus-gatekeeper**.

The new **globus-gatekeeper** will continue to run in the background until killed by invoking **globus-personal-gatekeeper** with the `-kill` or `-killall` argument. When killed, it will kill the **globus-gatekeeper** and **globus-job-manager** processes, remove state files and configuration data, and then exit. Jobs which are running when the personal gatekeeper is killed will continue to run, but their job directory will be destroyed so they may fail in the LRM.

The full set of command-line options to **globus-personal-gatekeeper** consists of:

<code>-help, -usage</code>	Print command-line option summary and exit
<code>-version</code>	Print software version
<code>-versions</code>	Print software version including DiRT information
<code>-list</code>	Print a list of all currently running personal gatekeepers. These entries will be printed one per line.
<code>-directory CONTACT</code>	Print the configuration directory for the personal gatekeeper with the contact string <i>CONTACT</i> .
<code>-debug</code>	Print additional debugging information when starting a personal gatekeeper. This option is ignored in other modes.
<code>-start</code>	Start a new personal gatekeeper process.
<code>-jmtime LRM</code>	Use <i>LRM</i> as the local resource manager interface. If not provided when starting a personal gatekeeper, the job manager will use the default <code>fork</code> LRM.
<code>-auditdir AUDIT_DIRECTORY</code>	Write audit report files to <i>AUDIT_DIRECTORY</i> . If not provided, the job manager will not write any audit files.

- `-port PORT` Listen for gatekeeper TCP/IP connections on the port *PORT*. If not provided, the gatekeeper will let the operating system choose.
- `-log[=DIRECTORY]` Write job manager log files to *DIRECTORY*. If *DIRECTORY* is omitted, the default of `$HOME` will be used. If this option is not present, the job manager will not write any log files.
- `-seg` Try to use the SEG mechanism to receive job state change information, instead of polling for these. These require either the system administrator or the user to run an instance of the **globus-job-manager-event-generator** program for the LRM specified by the `-jmtime` option.
- `-acctfile ACCOUNTING_FILE` Write gatekeeper accounting entries to *ACCOUNTING_FILE*. If not provided, no accounting records are written.

Examples

This example shows the output when starting a new personal gatekeeper which will schedule jobs via the lsf LRM, with debugging enabled.

```
% globus-personal-gatekeeper -start -jmtime lsf
```

```
verifying setup...
```

```
done.
```

```
GRAM contact: personal-grid.example.org:57846:/DC=org/DC=example/CN=Joe User
```

This example shows the output when listing the current active personal gatekeepers.

```
% globus-personal-gatekeeper -list
```

```
personal-grid.example.org:57846:/DC=org/DC=example/CN=Joe User
```

This example shows the output when querying the configuration directory for the above personal gatekeeper. gatekeepers.

```
% globus-personal-gatekeeper -directory "personal-grid.example.org:57846:/DC=org/DC=example/CN=Joe User"
```

```
/home/juser/.globus/.personal-gatekeeper.personal-grid.example.org.1337
```

```
% globus-personal-gatekeeper -kill "personal-grid.example.org:57846:/DC=org/DC=example/CN=Joe User"
```

```
killing gatekeeper: "personal-grid.example.org:57846:/DC=org/DC=example/CN=Joe User"
```

See Also

globusrun(1), globus-job-manager(8), globus-gatekeeper(8)

Name

globus-gram-audit -- Load GRAM4 and GRAM5 audit records into a database

globus-gram-audit [--conf *CONFIG_FILE*] [--check] [--delete] [--audit-directory *AUDITDIR*]

Description

The **globus-gram-audit** program loads audit records to an SQL-based database. It reads `$GLOBUS_LOCATION/etc/globus-job-manager.conf` by default to determine the audit directory and then uploads all files in that directory that contain valid audit records to the database configured by the **globus_gram_job_manager_auditing_setup_scripts** package. If the upload completes successfully, the audit files will be removed.

The full set of command-line options to **globus-gram-audit** consist of:

<code>--conf <i>CONFIG_FILE</i></code>	Use <i>CONFIG_FILE</i> instead of the default from the configuration file for audit database configuration.
<code>--check</code>	Check whether the insertion of a record was successful by querying the database after inserting the records. This is used in tests.
<code>--delete</code>	Delete audit records from the database right after inserting them. This is used in tests to avoid filling the database with test records.
<code>--audit-directory <i>DIR</i></code>	Look for audit records in <i>DIR</i> , instead of looking in the directory specified in the job manager configuration. This is used in tests to control which records are loaded to the database and then deleted.
<code>--query <i>SQL</i></code>	Perform the given SQL query on the audit database. This uses the database information from the configuration file to determine how to contact the database.

FILES

The **globus-gram-audit** uses the following files (paths relative to `$GLOBUS_LOCATION`).

<code>etc/globus-gram-job-manager.conf</code>	GRAM5 job manager configuration. It includes the default path to the audit directory
<code>etc/globus-gram-audit.conf</code>	Audit configuration. It includes the information needed to contact the audit database.

Name

globus-gatekeeper -- Authorize and execute a grid service on behalf of a user

```
globus-gatekeeper [-help]
[-conf PARAMETER_FILE]
[-test] [ -d | -debug ]
{ -inetd | -f }
[ -p PORT | -port PORT ]
[-home PATH] [ -l LOGFILE | -logfile LOGFILE ]
[-acctfile ACCTFILE]
[-e LIBEXECDIR]
[-launch_method { fork_and_exit | fork_and_wait | dont_fork } ]
[-grid_services SERVICEDIR]
[-globusid GLOBUSID]
[-gridmap GRIDMAP]
[-x509_cert_dir TRUSTED_CERT_DIR]
[-x509_cert_file TRUSTED_CERT_FILE]
[-x509_user_cert CERT_PATH]
[-x509_user_key KEY_PATH]
[-x509_user_proxy PROXY_PATH]
[-k]
[-globuskmap KMAP]
```

Description

The **globus-gatekeeper** program is a meta-server similar to **inetd** or **xinetd** that starts other services after authenticating the TCP connection using GSSAPI.

The most common use for the **globus-gatekeeper** program is to start instances of the globus-job-manager(8) service. A single **globus-gatekeeper** deployment can handle multiple different service configurations by having entries in the grid-services directory.

Typically, users interact with the **globus-gatekeeper** program via client applications such as globusrun(1), **globus-job-submit**, or tools such as CoG jglobus or Condor-G.

The full set of command-line options to **globus-gatekeeper** consists of:

-help	Display a help message to standard error and exit
-conf <i>PARAMETER_FILE</i>	Load configuration parameters from <i>PARAMETER_FILE</i> . The parameters in that file are treated as additional command-line options.
-test	Parse the configuration file and print out the POSIX user id of the globus-gatekeeper process, service home directory, service execution directory, and X.509 subject name and then exits.
-d, -debug	Run the globus-gatekeeper process in the foreground.
-inetd	Flag to indicate that the globus-gatekeeper process was started via inetd or a similar super-server. If this flag is set and the globus-gatekeeper was not started via inetd, a warning will be printed in the gatekeeper log.

- f Flag to indicate that the **globus-gatekeeper** process should run in the foreground. This flag has no effect when the **globus-gatekeeper** is started via inetd.
- p *PORT*, -port *PORT* Listen for connections on the TCP/IP port *PORT*. This option has no effect if the **globus-gatekeeper** is started via inetd or a similar service. If not specified and the gatekeeper is running as root, the default of 754 is used. Otherwise, the gatekeeper defaults to an ephemeral port.
- home *PATH* Sets the gatekeeper deployment directory to *PATH*. This is used to interpret relative paths for accounting files, libexecdir, certificate paths, and also to set the GLOBUS_LOCATION environment variable in the service environment. If not specified, the gatekeeper uses its working directory.
- l *LOGFILE*, -logfile *LOGFILE* Write status log entries to *LOGFILE*
- acctfile *ACCTFILE* Set the path to write accounting records to *ACCTFILE*. If not set, no accounting records will be written.
- e *LIBEXECDIR* Look for service executables in *LIBEXECDIR*. If not specified, the default of *HOME/libexec* is used.
- launch_method *fork_and_exit*, *fork_and_wait*, *fork*, *exec*, *execv*, *execvp*, *execve* Determine how to launch services. The method may be either *fork_and_exit* (the service runs completely independently of the gatekeeper, which exits after creating the new service process), *fork_and_wait* (the service is run in a separate process from the gatekeeper but the gatekeeper does not exit until the service terminates), or *dont_fork*, where the gatekeeper process becomes the service process via the *exec()* system call.
- grid_services *SERVICEDIR* Look for service descriptions in *SERVICEDIR*. If this is a relative path, it is interpreted relative to the *HOME* value. If this is not specified, the default of *HOME/etc/grid-services* is used.
- globusid *GLOBUSID* Sets the GLOBUSID environment variable to *GLOBUSID*. This variable is used to construct the gatekeeper contact string if it can not be parsed from the service credential.
- gridmap *GRIDMAP* Use the file at *GRIDMAP* to map GSSAPI names to POSIX user names. If not specified, the default of *HOME/etc/grid-mapfile* is used.
- x509_cert_dir *TRUSTED_CERT_DIR* Use the directory *TRUSTED_CERT_DIR* to locate trusted CA X.509 certificates. The gatekeeper sets the environment variable *X509_CERT_DIR* to this value.
- x509_cert_file *TRUSTED_CERT_FILE* OBSOLETE GSI OPTION
- x509_user_cert *CERT_PATH* Read the service X.509 certificate from *CERT_PATH*. The gatekeeper sets the *X509_USER_CERT* environment variable to this value.
- x509_user_key *KEY_PATH* Read the private key for the service from *KEY_PATH*. The gatekeeper sets the *X509_USER_KEY* environment variable to this value.
- x509_user_proxy *PROXY_PATH* Read the X.509 proxy certificate from *PROXY_PATH*. The gatekeeper sets the *X509_USER_PROXY* environment variable to this value.
- k Assume authentication with Kerberos 5 GSSAPI instead of X.509 GSSAPI.

-globusmap *KMAP* Assume authentication with Kerberos 5 GSSAPI instead of X.509 GSSAPI and use *KMAP* as the path to the kerberos principal to POSIX user mapping file.

ENVIRONMENT

If the following variables affect the execution of **globus-gatekeeper**

X509_CERT_DIR Directory containing X.509 trust anchors and signing policy files.

X509_USER_PROXY Path to file containing an X.509 proxy.

X509_USER_CERT Path to file containing an X.509 user certificate.

X509_USER_KEY Path to file containing an X.509 user key.

Files

\$GLOBUS_LOCATION/etc/globus-gatekeeper.conf Default path to gatekeeper configuration file.

\$GLOBUS_LOCATION/etc/grid-services/*SERVICENAME* Service configuration for *SERVICENAME*.

See also

globusrun(1), globus-job-manager(8)

Name

globus-job-manager -- Execute and monitor jobs

```
globus-job-manager {-type LRM} [-conf CONFIG_PATH] [-help] [-globus-host-manufacturer MANUFACTURER]
[-globus-host-cputype CPUTYPE] [-globus-host-osname OSNAME] [-globus-host-osversion OSVERSION] [-globus-
gatekeeper-host HOST] [-globus-gatekeeper-port PORT] [-globus-gatekeeper-subject SUBJECT] [-home GLOBUS_LOC-
ATION] [-target-globus-location TARGET_GLOBUS_LOCATION] [-condor-arch ARCH] [-condor-os OS] [-history
HISTORY_DIRECTORY] [-scratch-dir-base SCRATCH_DIRECTORY] [-enable-syslog] [-stdio-log LOG_DIRECTORY]
[-log-levels LEVELS] [-state-file-dir STATE_DIRECTORY] [-globus-tcp-port-range PORT_RANGE] [-x509-cert-dir
TRUSTED_CERTIFICATE_DIRECTORY] [-cache-location GASS_CACHE_DIRECTORY] [-k] [-extra-envvars
VAR=VAL, . . .] [-seg-module SEG_MODULE] [-audit-directory AUDIT_DIRECTORY] [-globus-toolkit-version
TOOLKIT_VERSION] [-disable-streaming] [-disable-usagestats] [-usagestats-target TARGET] [-service-tag SER-
VICE_TAG]
```

Description

The **globus-job-manager** program is a service which starts and controls GRAM jobs which are executed by a local resource management system, such as LSF or Condor. The **globus-job-manager** program is typically started by the **globus-gatekeeper** program and not directly by a user. It runs until all jobs it is managing have terminated or its delegated credentials have expired.

Typically, users interact with the **globus-job-manager** program via client applications such as **globusrun**, **globus-job-submit**, or tools such as CoG jglobus or Condor-G.

The full set of command-line options to **globus-job-manager** consists of:

-help	Display a help message to standard error and exit
-type <i>LRM</i>	Execute jobs using the local resource manager named <i>LRM</i> .
-conf <i>CONFIG_PATH</i>	Read additional command-line arguments from the file <i>CONFIG_PATH</i> . If present, this must be the first command-line argument to the globus-job-manager program.
-globus-host-manufacturer <i>MANUFACTURER</i>	Indicate the manufacturer of the system which the jobs will execute on. This parameter sets the value of the \$(GLOBUS_HOST_MANUFACTURER) RSL substitution to <i>MANUFACTURER</i>
-globus-host-cputype <i>CPU-TYPE</i>	Indicate the CPU type of the system which the jobs will execute on. This parameter sets the value of the \$(GLOBUS_HOST_CPUTYPE) RSL substitution to <i>CPUTYPE</i>
-globus-host-osname <i>OS-NAME</i>	Indicate the operating system type of the system which the jobs will execute on. This parameter sets the value of the \$(GLOBUS_HOST_OSNAME) RSL substitution to <i>OSNAME</i>
-globus-host-osversion <i>OSVERSION</i>	Indicate the operating system version of the system which the jobs will execute on. This parameter sets the value of the \$(GLOBUS_HOST_OSVERSION) RSL substitution to <i>OSVERSION</i>
-globus-gatekeeper-host <i>HOST</i>	Indicate the host name of the machine which the job was submitted to. This parameter sets the value of the \$(GLOBUS_GATEKEEPER_HOST) RSL substitution to <i>HOST</i>

<code>-globus-gatekeeper-port</code> <i>PORT</i>	Indicate the TCP port number of gatekeeper to which jobs are submitted to. This parameter sets the value of the <code>\$(GLOBUS_GATEKEEPER_PORT)</code> RSL substitution to <i>PORT</i>
<code>-globus-gatekeeper-subject</code> <i>SUBJECT</i>	Indicate the X.509 identity of the gatekeeper to which jobs are submitted to. This parameter sets the value of the <code>\$(GLOBUS_GATEKEEPER_SUBJECT)</code> RSL substitution to <i>SUBJECT</i>
<code>-home</code> <i>GLOBUS_LOCATION</i>	Indicate the path where the Globus Toolkit(r) is installed on the service node. This is used by the job manager to locate its support and configuration files.
<code>-target-globus-location</code> <i>TARGET_GLOBUS_LOCATION</i>	Indicate the path where the Globus Toolkit(r) is installed on the execution host. If this is omitted, the value specified as a parameter to <code>-home</code> is used. This parameter sets the value of the <code>\$(GLOBUS_LOCATION)</code> RSL substitution to <i>TARGET_GLOBUS_LOCATION</i>
<code>-history</code> <i>HISTORY_DIRECTORY</i>	Configure the job manager to write job history files to <i>HISTORY_DIRECTORY</i> . These files are described in the FILES section below.
<code>-scratch-dir-base</code> <i>SCRATCH_DIRECTORY</i>	Configure the job manager to use <i>SCRATCH_DIRECTORY</i> as the default scratch directory root if a relative path is specified in the job RSL's <code>scratch_dir</code> attribute.
<code>-enable-syslog</code>	Configure the job manager to write log messages via syslog. Logging is further controlled by the argument to the <code>-log-levels</code> parameter described below.
<code>-stdio-log</code> <i>LOG_DIRECTORY</i>	Configure the job manager to write log messages to files in the <i>LOG_DIRECTORY</i> directory. Files will be named <i>LOG_DIRECTORY/gram_YYYYMM-DD.log</i> . Logging is further controlled by the argument to the <code>-log-levels</code> parameter described below. The <i>LOG_DIRECTORY</i> value can include variables derived from the job manager environment using the same syntax as RSL substitutions. For example, <code>-stdio-log \$(HOME)</code> would cause each user's logs to be stored in their individual home directories.
<code>-log-levels</code> <i>LEVELS</i>	Configure the job manager to write log messages of certain levels to syslog and/or log files. The available log levels are FATAL, ERROR, WARN, INFO, DEBUG, and TRACE. Multiple values can be combined with the <code> </code> character. The default value of logging when enabled is <code>FATAL ERROR</code> .
<code>-state-file-dir</code> <i>STATE_DIRECTORY</i>	Configure the job manager to write state files to <i>STATE_DIRECTORY</i> . If not specified, the job manager uses the default of <code>\$(GLOBUS_LOCATION)/tmp/gram_job_state/</code> . This directory must be writable by all users and be on a file system which supports POSIX advisory file locks.
<code>-globus-tcp-port-range</code> <i>PORT_RANGE</i>	Configure the job manager to restrict its TCP/IP communication to use ports in the range described by <i>PORT_RANGE</i> . This value is also made available in the job environment via the <code>GLOBUS_TCP_PORT_RANGE</code> environment variable.
<code>-x509-cert-dir</code> <i>TRUSTED_CERTIFICATE_DIRECTORY</i>	Configure the job manager to search <i>TRUSTED_CERTIFICATE_DIRECTORY</i> for its list of trusted CA certificates and their signing policies. This value is also made available in the job environment via the <code>X509_CERT_DIR</code> environment variable.

<code>-cache-location</code> <code>GASS_CACHE_DIRECTORY</code>	Configure the job manager to use the path <code>GASS_CACHE_DIRECTORY</code> for its temporary GASS-cache files. This value is also made available in the job environment via the <code>GLOBUS_GASS_CACHE_DEFAULT</code> environment variable.
<code>-k</code>	Configure the job manager to assume it is using Kerberos for authentication instead of X.509 certificates. This disables some certificate-specific processing in the job manager.
<code>-extra-envvars</code> <code>VAR=VAL,...</code>	Configure the job manager to define a set of environment variables in the job environment beyond those defined in the base job environment. The format of the parameter to this argument is a comma-separated sequence of <code>VAR=VAL</code> pairs, where <code>VAR</code> is the variable name and <code>VAL</code> is the variables value.
<code>-seg-module</code> <code>SEG_MODULE</code>	Configure the job manager to use the schedule event generator module named by <code>SEG_MODULE</code> to detect job state changes events from the local resource manager, in place of the less efficient polling operations used in GT2. To use this, one instance of the globus-job-manager-event-generator must be running to process events for the LRM into a generic format that the job manager can parse.
<code>-audit-directory</code> <code>AUDIT_DIRECTORY</code>	Configure the job manager to write audit records to the directory named by <code>AUDIT_DIRECTORY</code> . This records can be loaded into a database using the globus-gram-audit program.
<code>-globus-toolkit-version</code> <code>TOOLKIT_VERSION</code>	Configure the job manager to use <code>TOOLKIT_VERSION</code> as the version for audit and usage stats records.
<code>-service-tag</code> <code>SERVICE_TAG</code>	Configure the job manager to use <code>SERVICE_TAG</code> as a unique identifier to allow multiple GRAM instances to use the same job state directories without interfering with each other's jobs. If not set, the value <code>untagged</code> will be used.
<code>-disable-streaming</code>	Configure the job manager to disable file streaming. This is propagated to the LRM script interface but has no effect in GRAM5.
<code>-disable-usagestats</code>	Disable sending of any usage stats data, even if <code>-usagestats-target</code> is present in the configuration.
<code>-usagestats-target</code> <code>TARGET</code>	Send usage packets to a data collection service for analysis. The <code>TARGET</code> string consists of a comma-separated list of <code>HOST:PORT</code> combinations, each containing an optional list of data to send. See Usage Stats Packets ¹ for more information about the tags. Special tag strings of <code>all</code> (which enables all tags) and <code>default</code> may be used, or a sequence of characters for the various tags.
<code>-condor-arch</code> <code>ARCH</code>	Set the architecture specification for condor jobs to be <code>ARCH</code> in job classified ads generated by the GRAM5 condor LRM script. This is required for the condor LRM but ignored for all others.
<code>-condor-os</code> <code>OS</code>	Set the operating system specification for condor jobs to be <code>OS</code> in job classified ads generated by the GRAM5 condor LRM script. This is required for the condor LRM but ignored for all others.

¹ <http://confluence.globus.org/display/~bester/GRAM5+Usage+Stats+Packets>

Environment

If the following variables affect the execution of **globus-job-manager**

HOME	User's home directory.
LOGNAME	User's name.
JOBMANAGER_SYSLOG_ID	String to prepend to syslog audit messages.
JOBMANAGER_SYSLOG_FAC	Facility to log syslog audit messages as.
JOBMANAGER_SYSLOG_LVL	Priority level to use for syslog audit messages.
GATEKEEPER_JM_ID	Job manager ID to be used in syslog audit records.
GATEKEEPER_PEER	Peer information to be used in syslog audit records
GLOBUS_ID	Credential information to be used in syslog audit records
GLOBUS_JOB_MANAGER_SLEEP	Time (in seconds) to sleep when the job manager is started. [For debugging purposes only]
GRID_SECURITY_HOST- TP_BODY_FD	File descriptor of an open file which contains the initial job request and to which the initial job reply should be sent. This file descriptor is inherited from the globus-gatekeeper .
X509_USER_PROXY	Path to the X.509 user proxy which was delegated by the client to the globus-gatekeeper program to be used by the job manager.
GRID_SECURITY_CONTEXT_FD	File descriptor containing an exported security context that the job manager should use to reply to the client which submitted the job.

Files

<code>\$HOME/.globus/job/HOST- NAME/LRM.TAG.red</code>	Job manager delegated user credential.
<code>\$HOME/.globus/job/HOST- NAME/LRM.TAG.lock</code>	Job manager state lock file.
<code>\$HOME/.globus/job/HOST- NAME/LRM.TAG.pid</code>	Job manager pid file.
<code>\$HOME/.globus/job/HOST- NAME/LRM.TAG.sock</code>	Job manager socket for inter-job manager communications.
<code>\$HOME/.globus/job/HOST- NAME/JOB_ID/</code>	Job-specific state directory.
<code>\$HOME/.globus/job/HOST- NAME/JOB_ID/stdin</code>	Standard input which has been staged from a remote URL.
<code>\$HOME/.globus/job/HOST- NAME/JOB_ID/stdout</code>	Standard output which will be staged from a remote URL.

<code>\$HOME/.globus/job/HOST-NAME/JOB_ID/stderr</code>	Standard error which will be staged from a remote URL.
<code>\$HOME/.globus/job/HOST-NAME/JOB_ID/x509_user_proxy</code>	Job-specific delegated credential.
<code>\$GLOBUS_LOCATION/tmp/gram_job_state/job.HOST-NAME.JOB_ID</code>	Job state file.
<code>\$GLOBUS_LOCATION/tmp/gram_job_state/job.HOST-NAME.JOB_ID.lock</code>	Job state lock file. In most cases this will be a symlink to the job manager lock file.
<code>\$GLOBUS_LOCATION/etc/globus-job-manager.conf</code>	Default location of the global job manager configuration file.
<code>\$GLOBUS_LOCATION/etc/grid-services/jobmanager-LRM</code>	Default location of the LRM-specific gatekeeper configuration file.

See Also

globusrun(1), globus-gatekeeper(8), globus-personal-gatekeeper(1), globus-gram-audit(8)

Name

globus-job-manager-event-generator -- Create LRM-independent SEG files for the job manager to use

globus-job-manager-event-generator [-help] {-scheduler *LRM*} [-background] [-pidfile *PIDPATH*]

Description

The **globus-job-manager-event-generator** program is a utility which uses LRM-specific SEG parsers to generate a LRM-independent log file that a job manager instance can use to process job status change events. This program runs independently of all **globus-job-manager** instances so that only one process needs to deal with the LRM interface. The **globus-job-manager-event-generator** program can be run as a privileged user if required to interface with the LRM.

In order for **globus-job-manager-event-generator** to handle events for a particular LRM, the `globus_scheduler_event_generator_job_manager_setup` setup package must be configured after the LRM-specific setup package has been run. This can be forced by **gpt-postinstall -force** or running the command **cd \$GLOBUS_LOCATION/setup/globus; ./setup-seg-job-manager.pl**.

The full set of command-line options to **globus-job-manager-event-generator** consists of:

- help Print command-line option summary and exit.
- scheduler *LRM* Process events for the local resource manager named by *LRM*.
- background Run **globus-job-manager-event-generator** as a background process. It will fork a new process, print out its process ID and then the original process will terminate.
- pidfile *PIDPATH* Write the process ID of an instance of **globus-job-manager-event-generator** to the file named by *PIDPATH*. This file can be used to kill or monitor the **globus-job-manager-event-generator** process.

Files

globus-job-manager-seg.conf Configuration file for **globus-job-manager-event-generator**. Each line consists of a string of the form `LRM_log_path=PATH`, which indicates the directory containing LRM-independent format SEG log files for the LRM. This file is created by the running the `globus_scheduler_event_generator_job_manager_setup` setup package.

See Also

globus-scheduler-event-generator(8), globus-job-manager(8)

Name

globus-fork-starter -- Start and monitor a fork job

globus-fork-starter

Description

The **globus-fork-starter** program is executes jobs specified on its standard input stream, recording the job state changes to a file defined in the `$GLOBUS_LOCATION/etc/globus-fork.conf` configuration file. It runs until its standard input stream is closed and all jobs it is managing have terminated. The log generated by this program can be used by the SEG to provide job state changes and exit codes to the GRAM service. The **globus-fork-starter** program is typically started by the fork GRAM module.

The **globus-fork-starter** program expects its input to be a series of task definitions, separated by the newline character, each representing a separate job. Each task definition contains a number of fields, separated by the colon character. The first field is always the literal string `100` indicating the message format, the second field is a unique job tag that will be distinguish the reply from this program when multiple jobs are submitted. The rest of fields contain attribute bindings. The supported attributes are:

<code>directory</code>	Working directory of the job
<code>environment</code>	Comma-separated list of strings defining environment variables. The form of these strings is <code>var=value</code>
<code>count</code>	Number of processes to start
<code>executable</code>	Full path to the executable to run
<code>arguments</code>	Comma-separated list of command-line arguments for the job
<code>stdin</code>	Full path to a file containing the input of the job
<code>stdout</code>	Full path to a file to write the output of the job to
<code>stderr</code>	Full path to a file to write the error stream of the job

Within each field, the following characters may be escaped by preceding them with the backslash character:

- backslash (\)
- semicolon (;)
- comma (,)
- equal (=)

Additionally, newline can be represented within a field by using the escape sequence `\n`.

For each job the **globus-fork-starter** processes, it replies by writing a single line to standard output. The replies again consist of a number of fields separated by the semicolon character.

For a successful job start, the first field of the reply is the literal `101`, the second field is the tag from the input, and the third field is a comma-separated list of SEG job identifiers which consist the concatenation of a UUID and a process id. The **globus-fork-starter** program will write state changes to the SEG log using these job identifiers.

For a failure, the first field of the reply is the literal 102, the second field is the tag from the input, the third field is the integer representation of a GRAM error code, and the fourth field is a string explaining the error.

ENVIRONMENT

If the following variables affect the execution of **globus-fork-starter**

GLOBUS_LOCATION Path to Globus Toolkit installation. This is used to locate the `globus-fork.conf` configuration file.

Files

`$GLOBUS_LOCATION/etc/globus-fork.conf` Path to fork SEG configuration file.

Chapter 3. Job description

Jobs are described in GRAM5's job description language. For detailed schema information see the [GRAM5 RSL documentation](#). For more information and examples please check the [Globusrun section](#) in the [GRAM5 Users's guide](#).

Chapter 4. Semantics and syntax of protocols

1. GRAM5 Protocol

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 as used by GRAM5. This is compatible with with the GRAM Protocol parsers in GRAM2 with extensions.

1.1. 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-Line:

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

1.2. 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

```
rs1: &( executable = "/bin/echo" )
    ( arguments = "hello" )
```

becomes

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

In GRAM5, protocol extensions are supported in the status update messages. These extensions are implemented as extra attribute names *after* all of the attributes defined in the messages below. Older GRAM protocol parsers will ignore those extensions that occur after the attributes in the messages defined below. In GRAM5, the following extensions are used:

<code>exit-code</code>	Job exit code. Sent in job state callbacks and in job status replies when the job completes.
<code>gt3-failure-type</code>	Failure detail type for staging errors. Sent in job state callbacks and in job status replies when a job fails.
<code>gt3-failure-message</code>	Failure detail message for more context for errors. Sent in job state callbacks and in job status replies when a job fails.
<code>gt3-failure-source</code>	Failure detail message for the source of a failed file transfer. Sent in job state callbacks and in job status replies when a job fails.
<code>gt3-failure-destination</code>	Failure detail message for the destination of a failed file transfer. Sent in job state callbacks and in job status replies when a job fails.
<code>version</code>	Job manager package version. Sent in all messages from the job manager.
<code>toolkit-version</code>	Toolkit release that the job manager is running. Sent in all messages from the job manager.

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.

1.3. Message Types

1.3.1. 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

<code>job-manager-name</code>	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-LRM".
<code>host-name</code>	The name of the host on which the gatekeeper is running. This exists only for compatibility with the HTTP/1.1 protocol.
<code>message-size</code>	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".

1.3.2. 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-LRM*.

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.

1.3.3. 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.

1.3.4. 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 mask callback-contact"
```

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.

1.3.5. 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 callback-contact"
```

The values of the emphasized text items are as below:

<i>job-contact</i>	The job contact string returned in a response to a job request message, or determined by querying the MDS system.
<i>host-name</i>	The name of the host on which the job manager is running. This exists only for compatibility with the HTTP/1.1 protocol.
<i>message-size</i>	The length of the content of the message, not including the HTTP/1.1 header.
<i>version</i>	The version of the GRAM protocol which is being used. For the protocol defined in this document, the value must be the string "2".
<i>callback-contact</i>	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 @ref globus_gram_protocol_job_state_updates "below".

1.3.6. 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:

<i>job-contact</i>	The job contact string returned in a response to a job request message, or determined by querying the MDS system.
<i>host-name</i>	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.

1.3.7. 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 protocol library documentation. 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
"signal"
```

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 its parameters.

1.3.8. 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.

<i>host-name</i>	The host part of the callback-contact URL. This exists only for compatibility with the HTTP/1.1 protocol.
<i>message-size</i>	The length of the content of the message, not including the HTTP/1.1 header.
<i>version</i>	The version of the GRAM protocol which is being used. For the protocol defined in this document, the value must be the string 2.
<i>job-contact</i>	The job contact of the job which has changed states.

1.3.9. 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 proceed 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.

1.3.10. Security Attributes

The following security attributes are needed to communicate with the Gatekeeper:

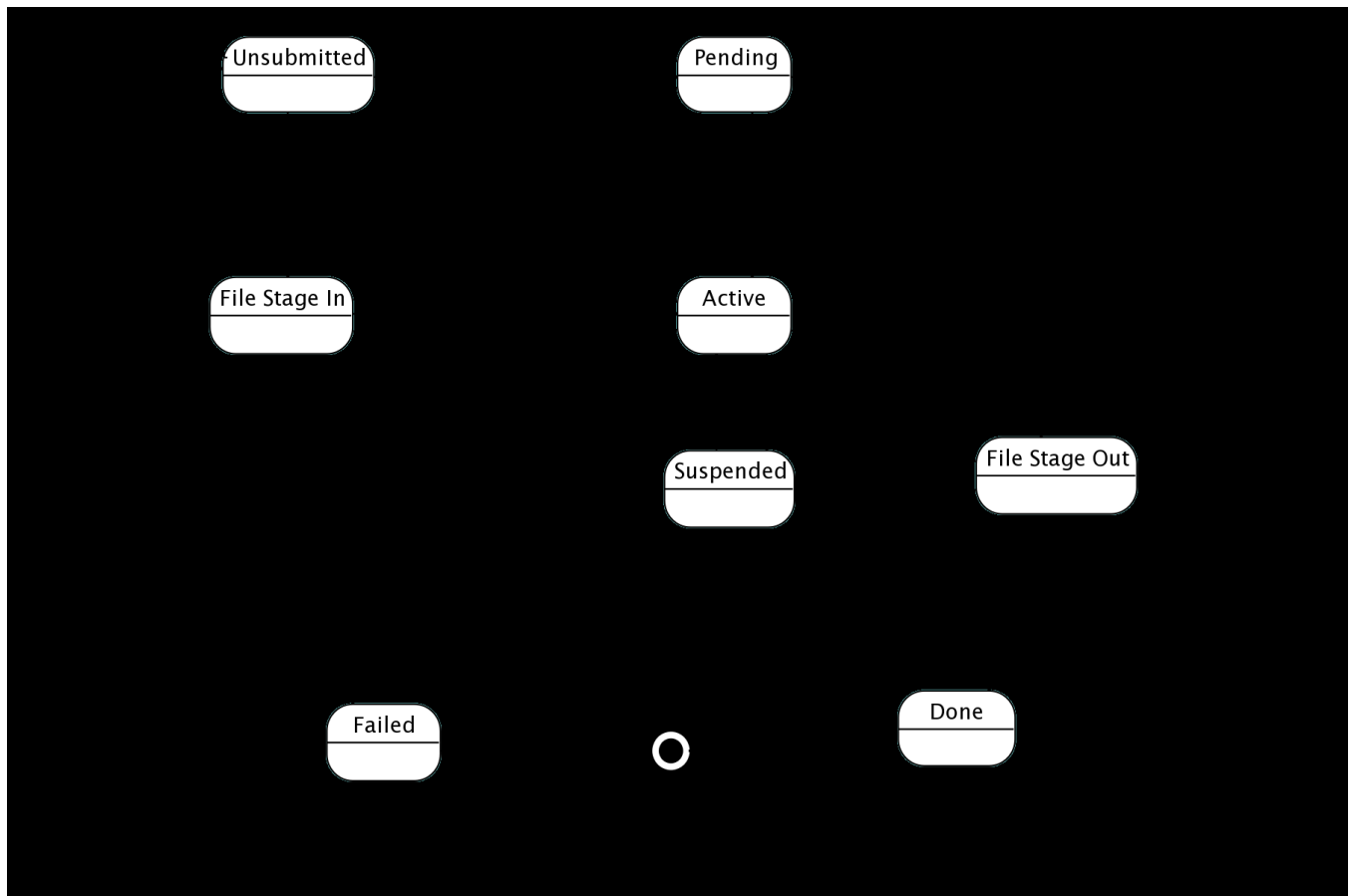
- Authentication must be done using GSSAPI mutual authentication
- Messages must be wrapped with support for the delegation message. When using Globus I/O, this is accomplished by using the the GLOBUS_IO_SECURE_CHANNEL_MODE_GSI_WRAP wrapping mode.

1.4. Job State Model

As the GRAM service processes a job, the job undergoes a series of state transitions. These states and their meanings follow:

Table 4.1. GRAM Job States

State	Meaning
GLOBUS_GRAM_PROTOCOL_JOB_STATE_UNSUBMITTED	Initial job state
GLOBUS_GRAM_PROTOCOL_JOB_STATE_STAGE_IN	Job staging in progress
GLOBUS_GRAM_PROTOCOL_JOB_STATE_PENDING	Job submitted to LRM, awaiting execution
GLOBUS_GRAM_PROTOCOL_JOB_STATE_ACTIVE	Job executing
GLOBUS_GRAM_PROTOCOL_JOB_STATE_SUSPENDED	Job made progress executing but is now suspended
GLOBUS_GRAM_PROTOCOL_JOB_STATE_STAGE_OUT	Job staging in progress after job completed
GLOBUS_GRAM_PROTOCOL_JOB_STATE_DONE	Job completed successfully
GLOBUS_GRAM_PROTOCOL_JOB_STATE_FAILED	Job was canceled or failed

Figure 4.1. GRAM State Transitions

Appendix A. Errors

Table A.1. GRAM5 Errors

Error Code	Reason	Possible Solutions
1	one of the RSL parameters is not supported	Check RSL documentation
2	the RSL length is greater than the maximum allowed	Use RSL substitutions to reduce length of RSL strings
3	an I/O operation failed	Enable trace logging and report to gram-dev@globus.org
4	jobmanager unable to set default to the directory requested	Check that RSL <code>directory</code> attribute refers to a directory that exists on the target system.
5	the executable does not exist	Check that the RSL <code>executable</code> attribute refers to an executable that exists on the target system.
6	of an unused <code>INSUFFICIENT_FUNDS</code>	Unimplemented feature.
7	authentication with the remote server failed	Check that the contact string contains the proper X.509 DN.
8	the user cancelled the job	Don't cancel jobs you want to complete.
9	the system cancelled the job	Check RSL requirements such as maximum time and memory are valid for the job.
10	data transfer to the server failed	Check gatekeeper and/or job manager logs to see why the process failed.
11	the stdin file does not exist	Check that the RSL <code>stdin</code> attribute refers to a file that exists on the target system or has a valid ftp, gsiftp, http, or https URL.
12	the connection to the server failed (check host and port)	Check that the service is running on the expected TCP/IP port. Check that no firewall prevents contacting that TCP/IP port. Check <code>\$GLOBUS_LOCATION/var/globus-gatekeeper.log</code> for runtime configuration errors.
13	the provided RSL 'maxtime' value is not an integer	Check that the RSL <code>maxtime</code> value evaluates to an integer.
14	the provided RSL 'count' value is not an integer	Check that the RSL <code>count</code> value evaluates to an integer.
15	the job manager received an invalid RSL	Check that the RSL string can be parsed by using <code>globusrun -p RSL</code> .
16	the job manager failed in allowing others to make contact	Check job manager log.
17	the job failed when the job manager attempted to run it	Verify that the LRM is configured properly.
18	an invalid paradyn was specified	OBSOLETE IN GRAM2
19	the provided RSL 'jobtype' value is invalid	The RSL <code>jobtype</code> attribute is not indicated as supported by the LRM. Valid <code>jobtype</code> values are <code>single</code> , <code>multiple</code> , <code>mpi</code> , and <code>condor</code> .
20	the provided RSL 'myjob' value is invalid	OBSOLETE IN GRAM5

Error Code	Reason	Possible Solutions
21	the job manager failed to locate an internal script argument file	Check that <code>\$GLOBUS_LOCATION/libexec/globus-job-manager-script.pl</code> exists and is executable. Check that the LRM-specific perl module is located in <code>\$GLOBUS_LOCATION/lib/perl/Globus/GRAM/JobManager/</code> directory and is valid. The command perl -I\$GLOBUS_LOCATION/lib/perl \$GLOBUS_LOCATION/lib/perl/Globus/GRAM/JobManager/LRM.pm can be used to check if there are any syntax errors in the script.
22	the job manager failed to create an internal script argument file	Check that your home directory is writable and not full.
23	the job manager detected an invalid job state	Check job manager logs.
24	the job manager detected an invalid script response	Check job manager logs. This is likely a bug in the LRM script.
25	the job manager detected an invalid script status	Check job manager logs. This is likely a bug in the LRM script.
26	the provided RSL 'jobtype' value is not supported by this job manager	Check that the RSL <code>jobtype</code> attribute is implemented by the LRM script. Note that some job types require configuration
27	unused ERROR_UNIMPLEMENTED	LRM does not support some feature included in the job request.
28	the job manager failed to create an internal script submission file	Check that the user's home file system is not full. Check job manager log
29	the job manager cannot find the user proxy	Check that client is delegating a proxy when authenticating with the gatekeeper. Check that the user's home filesystem and the <code>/tmp</code> file system are not full.
30	the job manager failed to open the user proxy	Check that the user's home filesystem and the <code>/tmp</code> file system are not full.
31	the job manager failed to cancel the job as requested	Check that the user's home filesystem and the <code>/tmp</code> file system are not full.
32	system memory allocation failed	Check job manager log for details.
33	the interprocess job communication initialization failed	OBSOLETE IN GRAM5
34	the interprocess job communication setup failed	OBSOLETE IN GRAM5
35	the provided RSL 'host count' value is invalid	Check that the RSL <code>host_count</code> attribute evaluates to an integer.
36	one of the provided RSL parameters is unsupported	Check job manager log for details about invalid parameter.
37	the provided RSL 'queue' parameter is invalid	Check that the RSL <code>queue</code> attribute evaluates to a string that corresponds to an LRM-specific queue name.
38	the provided RSL 'project' parameter is invalid	Check that the RSL <code>project</code> attribute evaluates to a string that corresponds to an LRM-specific project name.

Error Code	Reason	Possible Solutions
39	the provided RSL string includes variables that could not be identified	Check that all RSL substitutions are defined before being used in the job description.
40	the provided RSL 'environment' parameter is invalid	Check that the RSL <code>environment</code> attribute contains a sequence of <code>VARIABLE VALUE</code> pairs.
41	the provided RSL 'dryrun' parameter is invalid	Remove the RSL <code>dryrun</code> attribute from the job description.
42	the provided RSL is invalid (an empty string)	Include a non-empty RSL string in your job submission request.
43	the job manager failed to stage the executable	Check that the file service hosting the executable is reachable from the GRAM5 service node. Check that the executable exists on the file service node. Check that there is sufficient disk space in the user's home directory on the service node to store the executable.
44	the job manager failed to stage the stdin file	Check that the file service hosting the standard input file is reachable from the GRAM5 service node. Check that the standard input file exists on the file service node. Check that there is sufficient disk space in the user's home directory on the service node to store the standard input file.
45	the requested job manager type is invalid	OBSOLETE IN GRAM5
46	the provided RSL 'arguments' parameter is invalid	OBSOLETE IN GRAM2
47	the gatekeeper failed to run the job manager	Check the gatekeeper or job manager logs for more information.
48	the provided RSL could not be properly parsed	Check that the RSL string can be parsed by using globusrun -p RSL .
49	there is a version mismatch between GRAM components	Ask system administrator to upgrade GRAM service to GRAM2 or GRAM5
50	the provided RSL 'arguments' parameter is invalid	Check that the RSL <code>arguments</code> attribute evaluates to a sequence of strings.
51	the provided RSL 'count' parameter is invalid	Check that the RSL <code>count</code> attribute evaluates to a positive integer value.
52	the provided RSL 'directory' parameter is invalid	Check that the RSL <code>directory</code> attribute evaluates to a string.
53	the provided RSL 'dryrun' parameter is invalid	Check that the RSL <code>dryrun</code> attribute evaluates to either <code>yes</code> or <code>no</code> .
54	the provided RSL 'environment' parameter is invalid	Check that the RSL <code>environment</code> attribute evaluates to a sequence of <code>VARIABLE, VALUE</code> pairs.
55	the provided RSL 'executable' parameter is invalid	Check that the RSL <code>executable</code> attribute evaluates to a string value.
56	the provided RSL 'host_count' parameter is invalid	Check that the RSL <code>host_count</code> attribute evaluates to a positive integer value.
57	the provided RSL 'jobtype' parameter is invalid	Check that the RSL <code>jobtype</code> attribute evaluates to one of <code>single</code> , <code>multiple</code> , <code>mpi</code> , or <code>condor</code>

Error Code	Reason	Possible Solutions
58	the provided RSL 'maxtime' parameter is invalid	Check that the RSL <code>maxtime</code> attribute evaluates to a positive integer value.
59	the provided RSL 'myjob' parameter is invalid	OBSOLETE IN GRAM5.
60	the provided RSL 'paradyn' parameter is invalid	OBSOLETE IN GRAM2.
61	the provided RSL 'project' parameter is invalid	Check that the RSL <code>project</code> attribute evaluates to a string value.
62	the provided RSL 'queue' parameter is invalid	Check that the RSL <code>queue</code> attribute evaluates to a string value.
63	the provided RSL 'stderr' parameter is invalid	Check that the RSL <code>stderr</code> attribute evaluates to a string value or a sequence of <i>DESTINATION</i> URLs with optional <i>CACHE_TAG</i> string parameters.
64	the provided RSL 'stdin' parameter is invalid	Check that the RSL <code>stdin</code> attribute evaluates to a string value.
65	the provided RSL 'stdout' parameter is invalid	Check that the RSL <code>stdout</code> attribute evaluates to a string value or a sequence of <i>DESTINATION</i> URLs with optional <i>CACHE_TAG</i> string parameters.
66	the job manager failed to locate an internal script	Check job manager log for more details.
67	the job manager failed on the system call <code>pipe()</code>	OBSOLETE IN GRAM5
68	the job manager failed on the system call <code>fcntl()</code>	OBSOLETE IN GRAM2
69	the job manager failed to create the temporary stdout filename	OBSOLETE IN GRAM5
70	the job manager failed to create the temporary stderr filename	OBSOLETE IN GRAM5
71	the job manager failed on the system call <code>fork()</code>	OBSOLETE IN GRAM2
72	the executable file permissions do not allow execution	Check that the RSL <code>executable</code> attribute refers to an executable program or script.
73	the job manager failed to open stdout	Check that the RSL <code>stdout</code> attribute refers to one or more valid destination files or URLs.
74	the job manager failed to open stderr	Check that the RSL <code>stderr</code> attribute refers to one or more valid destination files or URLs.
75	the cache file could not be opened in order to relocate the user proxy	Check that the user's home directory is writable and not full on the GRAM5 service node.
76	cannot access cache files in <code>~/globus/.gass_cache</code> , check permissions, quota, and disk space	Check that the user's home directory is writable and not full on the GRAM5 service node.
77	the job manager failed to insert the contact in the client contact list	Check job manager log

Error Code	Reason	Possible Solutions
78	the contact was not found in the job manager's client contact list	Don't attempt to unregister callback contacts that are not registered
79	connecting to the job manager failed. Possible reasons: job terminated, invalid job contact, network problems, ...	Check that the job manager process is running. Check that the job manager credential has not expired. Check that the job manager contact refers to the correct TCP/IP host and port. Check that the job manager contact is not blocked by a firewall.
80	the syntax of the job contact is invalid	Check the syntax of job contact string.
81	the executable parameter in the RSL is undefined	Include the RSL <code>executable</code> in all job requests.
82	the job manager service is misconfigured. <code>condor arch</code> undefined	Add the <code>-condor-arch</code> to the command-line or configuration file for a job manager configured to use the <code>condor</code> LRM.
83	the job manager service is misconfigured. <code>condor os</code> undefined	Add the <code>-condor-os</code> to the command-line or configuration file for a job manager configured to use the <code>condor</code> LRM.
84	the provided RSL 'min_memory' parameter is invalid	Check that the RSL <code>min_memory</code> attribute evaluates to a positive integer value.
85	the provided RSL 'max_memory' parameter is invalid	Check that the RSL <code>max_memory</code> attribute evaluates to a positive integer value.
86	the RSL 'min_memory' value is not zero or greater	Check that the RSL <code>min_memory</code> attribute evaluates to a positive integer value.
87	the RSL 'max_memory' value is not zero or greater	Check that the RSL <code>max_memory</code> attribute evaluates to a positive integer value.
88	the creation of a HTTP message failed	Check job manager log.
89	parsing incoming HTTP message failed	Check job manager log.
90	the packing of information into a HTTP message failed	Check job manager log.
91	an incoming HTTP message did not contain the expected information	Check job manager log.
92	the job manager does not support the service that the client requested	Check that the client is talking to the correct service
93	the gatekeeper failed to find the requested service	OBSOLETE IN GRAM2
94	the jobmanager does not accept any new requests (shutting down)	Execute queries before the job has been cleaned up.
95	the client failed to close the listener associated with the callback URL	Call <code>globus_gram_client_callback_disallow()</code> with a valid the callback contact.
96	the gatekeeper contact cannot be parsed	Check the syntax of the gatekeeper contact string you are attempting to contact.
97	the job manager could not find the 'poe' command	OBSOLETE IN GRAM2
98	the job manager could not find the 'mpirun' command	Configure the LRM script with <code>mpirun</code> in your path.

Error Code	Reason	Possible Solutions
99	the provided RSL 'start_time' parameter is invalid	OBSOLETE IN GRAM2
100	the provided RSL 'reservation_handle' parameter is invalid	OBSOLETE IN GRAM2
101	the provided RSL 'max_wall_time' parameter is invalid	Check that the RSL <code>max_wall_time</code> attribute evaluates to a positive integer.
102	the RSL 'max_wall_time' value is not zero or greater	Check that the RSL <code>max_wall_time</code> attribute evaluates to a positive integer.
103	the provided RSL 'max_cpu_time' parameter is invalid	Check that the RSL <code>max_cpu_time</code> attribute evaluates to a positive integer.
104	the RSL 'max_cpu_time' value is not zero or greater	Check that the RSL <code>max_cpu_time</code> attribute evaluates to a positive integer.
105	the job manager is misconfigured, a scheduler script is missing	Check that the administrator has configured the LRM by running its setup script.
106	the job manager is misconfigured, a scheduler script has invalid permissions	Check that the administrator has installed the <code>GLLOBUS_LOCATION/libexec/globus-job-manager-script.pl</code> script. Check that the file system containing that script allows file execution.
107	the job manager failed to signal the job	OBSOLETE IN GRAM2
108	the job manager did not recognize/support the signal type	Check that your signal operation is using the correct signal constant.
109	the job manager failed to get the job id from the local scheduler	OBSOLETE IN GRAM2
110	the job manager is waiting for a commit signal	Send a two-phase commit signal to the job manager to acknowledge receiving the job contact from the job manager.
111	the job manager timed out while waiting for a commit signal	Send a two-phase commit signal to the job manager to acknowledge receiving the job contact from the job manager. Increase the two-phase commit time out for your job. Check that the job manager contact TCP/IP port is reachable from your client.
112	the provided RSL 'save_state' parameter is invalid	Check that the RSL <code>save_state</code> attribute is set to <code>yes</code> or <code>no</code> .
113	the provided RSL 'restart' parameter is invalid	Check that the RSL <code>restart</code> attribute evaluates to a string containing a job contact string.
114	the provided RSL 'two_phase' parameter is invalid	Check that the RSL <code>two_phase</code> attribute evaluates to a positive integer.
115	the RSL 'two_phase' value is not zero or greater	Check that the RSL <code>two_phase</code> attribute evaluates to a positive integer.
116	the provided RSL 'stdout_position' parameter is invalid	OBSOLETE IN GRAM5
117	the RSL 'stdout_position' value is not zero or greater	OBSOLETE IN GRAM5

Error Code	Reason	Possible Solutions
118	the provided RSL 'stderr_position' parameter is invalid	OBSOLETE IN GRAM5
119	the RSL 'stderr_position' value is not zero or greater	OBSOLETE IN GRAM5
120	the job manager restart attempt failed	OBSOLETE IN GRAM2
121	the job state file doesn't exist	Check that the job contact you are trying to restart matches one that the job manager returned to you.
122	could not read the job state file	Check that the state file directory is not full.
123	could not write the job state file	Check that the state file directory is not full.
124	old job manager is still alive	Contact the returned job manager contact to manage the job you are trying to restart.
125	job manager state file TTL expired	OBSOLETE in GRAM2
126	it is unknown if the job was submitted	Check job manager log.
127	the provided RSL 'remote_io_url' parameter is invalid	Check that the RSL <code>remote_io_url</code> attribute evaluates to a string value.
128	could not write the remote io url file	Check that the user's home file system on the job manager service node is writable and not full.
129	the standard output/error size is different	Send a stdio update signal to redirect the job manager output to a new URL
130	the job manager was sent a stop signal (job is still running)	Submit a restart request to monitor the job.
131	the user proxy expired (job is still running)	Generate a new proxy and then submit a restart request to monitor the job.
132	the job was not submitted by original job-manager	OBSOLETE IN GRAM2
133	the job manager is not waiting for that commit signal	Do not send a commit signal to a job that is not waiting for a commit signal.
134	the provided RSL scheduler specific parameter is invalid	Check the LRM-specific documentation to determine what values are legal for the RSL extensions implemented by the LRM.
135	the job manager could not stage in a file	Check that the file service hosting the file to stage is reachable from the GRAM5 service node. Check that the file to stage exists on the file service node. Check that there is sufficient disk space in the user's home directory on the service node to store the file to stage.
136	the scratch directory could not be created	Check that the directory named by the RSL <code>scratch_dir</code> attribute exists and is writable. Check that the directory named by the RSL <code>scratch_dir</code> attribute is not full.
137	the provided 'gass_cache' parameter is invalid	Check that the RSL <code>gass_cache</code> attribute evaluates to a string.
138	the RSL contains attributes which are not valid for job submission	Do not use restart- or signal-only RSL attributes when submitting a job.

Error Code	Reason	Possible Solutions
139	the RSL contains attributes which are not valid for stdio update	Do not use submit- or restart-only RSL attributes when sending a stdio update signal to a job.
140	the RSL contains attributes which are not valid for job restart	Do not use submit- or signal-only RSL attributes when restarting a job.
141	the provided RSL 'file_stage_in' parameter is invalid	Check that the RSL <code>file_stage_in</code> attribute evaluates to a sequence of <i>SOURCE DESTINATION</i> pairs.
142	the provided RSL 'file_stage_in_shared' parameter is invalid	Check that the RSL <code>file_stage_in_shared</code> attribute evaluates to a sequence of <i>SOURCE DESTINATION</i> pairs.
143	the provided RSL 'file_stage_out' parameter is invalid	Check that the RSL <code>file_stage_out</code> attribute evaluates to a sequence of <i>SOURCE DESTINATION</i> pairs.
144	the provided RSL 'gass_cache' parameter is invalid	Check that the RSL <code>gass_cache</code> attribute evaluates to a string.
145	the provided RSL 'file_cleanup' parameter is invalid	Check that the RSL <code>file_clean_up</code> attribute evaluates to a sequence of strings.
146	the provided RSL 'scratch_dir' parameter is invalid	Check that the RSL <code>scratch_dir</code> attribute evaluates to a string.
147	the provided scheduler-specific RSL parameter is invalid	Check the LRM-specific documentation to determine what values are legal for the RSL extensions implemented by the LRM.
148	a required RSL attribute was not defined in the RSL spec	Check that the RSL <code>executable</code> attribute is present in your job request RSL. Check that the RSL <code>restart</code> attributes is present in your restart RSL.
149	the <code>gass_cache</code> attribute points to an invalid cache directory	Check that the RSL <code>gass_cache</code> attributes evaluates to a directory that exists or can be created. Check that the user's home file system is writable and not full.
150	the provided RSL 'save_state' parameter has an invalid value	Check that the RSL <code>save_state</code> attribute has a value of <code>yes</code> or <code>no</code> .
151	the job manager could not open the RSL attribute validation file	Check that <code>\$GLOBUS_LOCATION/share/globus_gram_job_manager/globus-gram-job-manager.rvf</code> is present and readable on the job manager service node. Check that <code>\$GLOBUS_LOCATION/share/globus_gram_job_manager/LRM.rvf</code> is readable on the job manager service node if present.
152	the job manager could not read the RSL attribute validation file	Check that <code>\$GLOBUS_LOCATION/share/globus_gram_job_manager/globus-gram-job-manager.rvf</code> is valid. Check that <code>\$GLOBUS_LOCATION/share/globus_gram_job_manager/LRM.rvf</code> is valid if present.
153	the provided RSL 'proxy_timeout' is invalid	Check that RSL <code>proxy_timeout</code> attribute evaluates to a positive integer.
154	the RSL 'proxy_timeout' value is not greater than zero	Check that RSL <code>proxy_timeout</code> attribute evaluates to a positive integer.

Error Code	Reason	Possible Solutions
155	the job manager could not stage out a file	Check that the source file being staged exists on the job manager service node. Check that the directory of the destination file being staged exists on the file service node. Check that the directory of the destination file being staged is writable by the user. Check that the destination file service is reachable by the job manager service node.
156	the job contact string does not match any which the job manager is handling	Check that the job contact string matches one returned from a job request.
157	proxy delegation failed	Check that the job manager service node trusts the signer of your credential. Check that you trust the signer of the job manager service node's credential.
158	the job manager could not lock the state lock file	Check that the file system holding the job state directory supports POSIX advisory locking. Check that the job state directory is writable by the user on the service node. Check that the job state directory is not full.
159	an invalid globus_io_clientattr_t was used.	Check that you have initialized the <code>globus_io_clientattr_t</code> attribute prior to using it with the GRAM client API.
160	an null parameter was passed to the gram library	Check that you are passing legal values to all GRAM API calls.
161	the job manager is still streaming output	OBSOLETE IN GRAM5
162	the authorization system denied the request	Check with your GRAM system administrator to allow a particular certificate to be authorized.
163	the authorization system reported a failure	Check with your system administrator to verify that the authorization system is configured properly.
164	the authorization system denied the request - invalid job id	Check with your system administrator to verify that the authorization system is configured properly. Use a credential which is authorized to interact with a particular GRAM job.
165	the authorization system denied the request - not authorized to run the specified executable	Check with your system administrator to verify that the authorization system is configured properly. Use a credential which is authorized to interact with a particular GRAM job.
166	the provided RSL 'user_name' parameter is invalid.	Check that the RSL <code>user_name</code> attribute evaluates to a string.
167	the job is not running in the account named by the 'user_name' parameter.	Ask with the GRAM system administrator to add an authorization entry to allow your credential to run jobs as the specified user account.

Glossary