GT 4.2.1 GRAM2: Admin Guide
Table of Contents

1. Building and Installing GRAM ................................................................. 1
2. Configuring GRAM .................................................................................... 2
   1. Configuration Files ........................................................................... 2
   2. Configure Inetd and Xinetd ............................................................... 3
   3. Advanced Configuration .................................................................... 5
3. Job Manager ............................................................................................. 6
   1. Job Manager Setup .......................................................................... 6
   2. Job Manager Configuration .............................................................. 6
   3. RSL Validation File Format ............................................................... 6
   4. Job Execution Environment ............................................................... 6
   5. RSL attributes ................................................................................ 7
   6. Adding job managers ...................................................................... 7
4. Scheduler Event Generator / Job Manager Integration .............................. 8
   1. Introduction ..................................................................................... 8
   2. globus-job-manager-event-generator ............................................ 8
   3. Job Manager Configuration ............................................................ 9
   4. globus-job-manager-event-generator Configuration ....................... 10
   5. Running the globus-job-manager-event-generator .......................... 10
   6. Troubleshooting the globus-job-manager-event-generator ............. 11
5. Audit Logging .......................................................................................... 12
6. Testing GRAM ........................................................................................ 13
7. Usage statistics collection by the Globus Alliance .................................. 14
Chapter 1. Building and Installing GRAM

Gram will be installed during the normal installation of the GT4.2 and needs no extra steps.
Chapter 2. Configuring GRAM

1. Configuration Files

GRAM uses the following configuration files and directories:

1. globus-gatekeeper.conf
2. globus-job-manager.conf
3. grid-services
4. /etc/grid-security/grid-mapfile

1. globus-gatekeeper.conf

Here is the default globus-gatekeeper.conf:

- x509_cert_dir /etc/grid-security/certificates
- x509_user_cert /etc/grid-security/hostcert.pem
- x509_user_key /etc/grid-security/hostkey.pem
- gridmap /etc/grid-security/grid-mapfile
- home /usr/local/globus
- e libexec
- logfile var/globus-gatekeeper.log
- port 2119
- grid_services etc/grid-services
- inetd

- x509_cert_dir specifies where to find the trusted CA certificates.
- x509_user_cert specifies where to find the gatekeeper cert.
- x509_user_key specifies where to find the gatekeeper key.
- gridmap specifies where to find the grid-mapfile.
- home specifies where the -e and -logfile variables are relative to. By default, this is your $GLOBUS_LOCATION.
- e specifies where to find scripts.
- logfile specifies where the gatekeeper should put its log.
- port specifies what port the gatekeeper will run on.
- grid_service specifies where the directory which contains the configured jobmanagers is.
- inetd specifies that the gatekeeper should exit after dealing with one request. That is because inetd will launch a copy of the gatekeeper for every request that comes in to the port in -port. If you are running a gatekeeper by hand, don't use this flag.

---

1 #gram2-admin-configfile-gatekeeper
2 #gram2-admin-configfile-jobmanager
3 #gram2-admin-configfile-gridservices
4 #gram2-admin-configfile-gridmapfile
2. globus-job-manager.conf

Here is an example globus-job-manager.conf:

-home "/home/bacon/pkggs/globus-2.4"
-globus-gatekeeper-host bacon.mcs.anl.gov
-globus-gatekeeper-port 2119
-globus-gatekeeper-subject "/O=Grid/O=Globus/CN=bacon.mcs.anl.gov"
-globus-host-cputype i686
-globus-host-manufacturer pc
-globus-host-osname Linux
-globus-host-osversion 2.2.19-4.7mdk
-save-logfile on_error
-state-file-dir /home/bacon/pkggs/globus-2.4/tmp
-machine-type unknown

See Job Manager Configuration for details. Note that the entries in this file are combined with the entries in $GLOBUS_LOCATION/etc/grid-services for any specific jobmanager.

3. grid-services/

$GLOBUS_LOCATION/etc/grid-services contains one file per configured jobmanager. The default jobmanager is contained in a file named "jobmanager". Actually this is a symbolic link to one of the jobmanager files located in the same directory that will be used as the default jobmanager. Here are the contents of an example file for a fork jobmanager:


To install additional jobmanagers, you need to download the scheduler-specific jobmanager package from the download page.

4. /etc/grid-security/grid-mapfile

The grid-mapfile specifies the list of authorized users of this resource. Each entry is a pairing of a subject name and a local user account. The location of this file is specified in globus-gatekeeper.conf

2. Configure Inetd and Xinetd

While running globus-personal-gatekeeper as a user is a good test, you will want to configure your machine to run globus-gatekeeper as root, so that other people will be able to use your gatekeeper. If you just run the personal gatekeeper, you won’t have authority to su to other user accounts. To setup a full gatekeeper, you will need to make the following modifications as root:

In /etc/services, add the service name "gsigatekeeper" to port 2119.

gsigatekeeper 2119/tcp # Globus Gatekeeper

Depending on whether your host is running inetd or xinetd, you will need to modify its configuration. If the directory /etc/xinetd.d/ exists, then your host is likely running xinetd. If the directory doesn't exist, your host is likely running inetd. Follow the appropriate instructions below according to what your host is running.

Inetd

gram2-admin-jobmanager-config
http://www.globus.org/toolkit/downloads/development/
For inetd, add the following entry, all on one line, to `/etc/inetd.conf`. Be sure to replace GLOBUS_LOCATION below with the actual value of $GLOBUS_LOCATION in your environment.

```
gsigatekeeper stream tcp nowait root
   /usr/bin/env env LD_LIBRARY_PATH=GLOBUS_LOCATION/lib
   GLOBUS_LOCATION/sbin/globus-gatekeeper
      -conf GLOBUS_LOCATION/etc/globus-gatekeeper.conf
```

This entry has changed from the entry provided for the gatekeeper in the Globus Toolkit 2.0 Administrator's Guide. The reason is that if you followed the instructions from the install section, you do not have a static gatekeeper. This requires you to set the LD_LIBRARY_PATH so that the gatekeeper can dynamically link against the libraries in $GLOBUS_LOCATION/lib. To accomplish the setting of the environment variable in inetd, we use /usr/bin/env (the location may vary on your system) to first set LD_LIBRARY_PATH, and then to call the gatekeeper itself.

The advantage of this setup is that when you apply a security update to your installation, the gatekeeper will pick it up dynamically without your having to rebuild it.

**Xinetd**

For xinetd, add a file called "globus-gatekeeper" to the `/etc/xinetd.d/` directory that has the following contents. Be sure to replace GLOBUS_LOCATION below with the actual value of $GLOBUS_LOCATION in your environment.

```
service gsigatekeeper
{
   socket_type  = stream
   protocol     = tcp
   wait         = no
   user         = root
   env          = LD_LIBRARY_PATH=GLOBUS_LOCATION/lib
   server       = GLOBUS_LOCATION/sbin/globus-gatekeeper
   server_args  = -conf GLOBUS_LOCATION/etc/globus-gatekeeper.conf
   disable      = no
}
```

This entry has changed from the entry provided for the gatekeeper in the Globus Toolkit 2.0 Administrator's Guide. The reason is that if you followed the instructions from the install section, you do not have a static gatekeeper. This requires you to set the LD_LIBRARY_PATH so that the gatekeeper can dynamically link against the libraries in $GLOBUS_LOCATION/lib. To accomplish the setting of the environment variable in xinetd, we use the "env =" option to set LD_LIBRARY_PATH in the gatekeeper's environment.

The advantage of this setup is that when you apply a security update to your installation, the gatekeeper will pick it up dynamically without your having to rebuild it.

After you have added the globus-gatekeeper service to either inetd or xinetd, you will need to notify inetd (or xinetd) that its configuration file has changed. To do this, follow the instructions for the server you are running below.

**Inetd**

On most Linux systems, you can simply run `killall -HUP inetd` On other systems, the following has the same effect: `ps aux | grep inetd | awk '{print $2;}' | xargs kill -HUP`

**Xinetd**

On most linux systems, you can simply run `/etc/rc.d/init.d/xinetd restart`. Your system may also support the "reload" option. On other systems (or if that doesn't work), see man xinetd.
At this point, your gatekeeper will start up when a connection comes in to port 2119, and will keep a log of its activity in $GLOBUS_LOCATION/var/globus-gatekeeper.log. However, it does not yet have any authorization mapping between certificate subjects and usernames. You will need to create a file named /etc/grid-security/grid-mapfile which consists of single line entries listing a certificate subject and a username, like this:

"/O=Grid/O=Globus/OU=your.domain/CN=Your Name" youruserid

You can check your subject name using grid-cert-info -subject. There are utility commands in $GLOBUS_LOCATION/sbin/grid-mapfile* for adding entries, removing entries, and checking consistency.

### 3. Advanced Configuration

Advanced configuration of GRAM consists of the following tasks:

1. **Adding jobmanagers**
2. **Adding trust to a new CA/removing trust from an old CA**
3. **Starting your own CA**

#### 1. Adding jobmanagers

For information about how to add a job manager for Condor, PBS, or LSF please look [here](#).

#### 2. Adding trust to a new CA/removing trust from an old CA

The set of trusted Certificate Authorities is contained in the /etc/grid-security/certificates directory. By default, that directory contains two entries. One, called 42864e48.0 is the public certificate of the Globus CA. The other, called 42864e48.signing_policy is the signing policy for the Globus CA certificate.

The name "42864e8" comes from the openssl -hash option. If you create your own Certificate Authority, you can use the command openssl x509 -in yourcert.pem -noout -hash to determine its hash value. You will need to place a copy of that public certificate, under the name hash.0 (where "hash" corresponds to the output of the openssl command) in the /etc/grid-security/certificates/ of every Toolkit installation which you want to trust certificates which your CA has signed. Additionally, you will have to create a hash.signing_policy file which contains the DN of your CA, as well as the namespace for which your CA signs.

Namespaces for CAs are designed to be unique. If you do establish your own CA, do not use the "/O=Grid/O=Globus" namespace. That is reserved for the Globus CA.

Removing trust for a particular CA is as easy as deleting the two files which correspond to the CA. First, look for the .signing_policy which corresponds to the CA you want to remove. Then remove both the .signing_policy and .0 file that correspond to that hash.

#### 3. Starting your own CA

There is a Globus package named Simple CA which is designed to help you establish a CA for your test Grid.
Chapter 3. Job Manager

The GRAM Job Manager program starts and monitors jobs on behalf of a GRAM client application. The job manager is typically started by the Gatekeeper program. It interfaces with a local scheduler to start jobs based on a job request RSL string.

1. Job Manager Setup

Job managers for Fork, PBS, LSF and Condor are included in the toolkit. But only the fork job manager is installed by default during a normal installation of the toolkit. The others must be installed separately if they are needed.

To install them from a source distribution, follow these steps:

1. go to the installer directory (e.g. gt4.2.1-all-source-installer)
2. make gt4-gram-[pbs|lsf|condor]
3. make install

Using PBS as the example, make sure the scheduler commands are in your path (qsub, qstat, pbsnodes). For PBS, another setup step is required to configure the remote shell for rsh access:

% cd $GLOBUS_LOCATION/setup/globus
% ./setup-globus-job-manager-pbs --remote-shell=rsh

The following links give extra information what parameters can be added to the setup scripts of the different scheduler adapters:

- **Condor Job Manager Setup**
- **PBS Job Manager Setup**
- **LSF Job Manager Setup**

2. Job Manager Configuration

Job Manager Configuration

3. RSL Validation File Format

RSL Validation File Format

4. Job Execution Environment

Job Execution Environment

---

5. RSL attributes

RSL Attributes

6. Adding job managers

The fork job manager scheduler will be installed during a normal installation of the toolkit and will be installed as the default job manager service (e.g. $GLOBUS_LOCATION/grid-services/jobmanager). Additional job manager scheduler packages installed will be installed using the convention "jobmanager-<scheduler-name>" (e.g. $GLOBUS_LOCATION/grid-services/jobmanager-pbs).

Information on how to install an additional job manager for Condor, PBS or LSF can be found here.

All job manager scheduler setup packages have the argument "-service-name <name>" in order to install a non-fork scheduler as the default job manager service. For example, this command will set the pbs scheduler as the default job manager service:

% setup-globus-job-manager-pbs -service-name jobmanager

If you need to alter the behavior of the job manager scheduler interface, or you want to create a new job manager scheduler interface for a scheduler that is not available, see this tutorial web page. The details of how to make a client submit to a non-default gatekeeper is covered in the user's guide section.

Note: If you wish to have your job manager report into your MDS, you need to install the appropriate GRAM Reporter setup package for your scheduler. The GRAM Reporter setup packages for each scheduler can be found on the download page. The details of how to make a client submit to a non-default gatekeeper is covered in the user's guide section.

Note: If you wish to have your job manager report into your MDS, you need to install the appropriate GRAM Reporter setup package for your scheduler. The GRAM Reporter setup packages for each scheduler can be found on the download page.

---

8 #gram2-admin-jobmanager-setup
9 http://www.globus.org/toolkit/downloads/development/
10 http://www.globus.org/toolkit/downloads/development/
Chapter 4. Scheduler Event Generator / Job Manager Integration

1. Introduction

This option is a method for the GRAM2 Job Manager to monitor the jobs it submits to the local scheduler. After installing, you can configure a job manager to use the new event based method for monitoring jobs, instead of the script-based polling implementation.

This change consists of a few parts

• A new script `globus-job-manager-event-generator` which translates scheduler-specific log information to a general form which the job manager can parse. This script may need to be run as a privileged account in order to parse the log files, depending on the log permissions. This script MUST be running in order for Job Manager processes to receive job state change notifications from the scheduler.

• A new SEG module `globus_scheduler_event_generator_job_manager` which parses a log file to determine which job state changes occur for jobs being managed by a pre-WS GRAM Job Manager.

• Changes to the `globus-gram-job-manager` program to use the Scheduler Event Generator API to look for job state change events in a log file instead using scripts to query the scheduler state.

2. globus-job-manager-event-generator

The `globus-job-manager-event-generator` script creates a log of all scheduler events related to a particular scheduler instance. This script was created for two purposes

• To avoid requiring that all GRAM user’s have the privileges to read the scheduler’s log file. Users may not be allowed read access to the scheduler’s log files on all sites. The Job Manager processes is run under the user’s local account (as mapped in the gridmap file), it is this processes that will be updated for job status via the SEG log file instead of directly from the scheduler’s log file.

• To provide a simple format for the scheduler event generator logs so that the job manager will be able to quickly recover state information if the job manager is terminated and restarted. Some scheduler logs are difficult to parse, or inefficient for seeking to a particular timestamp (as is necessary for recovering job state change information). The data written by this script is easily locatable by date, and it is simple to remove old job information without compromising current job manager execution.

One instance of the `globus-job-manager-event-generator` must be running for each scheduler type to be implemented using the Scheduler Event Generator interface to receive job state changes. This program is located in the `sbin` subdirectory of the GLOBUS_LOCATION. The typical command line for this program is `$GLOBUS_LOCATION/sbin/globus-job-manager-event-generator -s SCHEDULER_TYPE`, where `SCHEDULER_TYPE` is the scheduler name of the Scheduler Event Generator module which should be used to generate events (lsf, condor, pbs).

For example, to start the event generator program to monitor an LSF batch system:

```
$GLOBUS_LOCATION/sbin/globus-job-manager-event-generator -s lsf
```

NOTE: if the `globus-job-manager-event-generator` is not running, no job state changes will be sent from any job manager program which is configured to use the Scheduler Event Generator.
3. Job Manager Configuration

By default, the job manager is configured to use the pre-WS GRAM script-based polling method. A new command line option (-seg) was added to the globus-job-manager program to enable using the Scheduler Event Generator-driven job state change notifications.

There are two ways to configure the job manager to use the scheduler event generator: globally, in the $GLOBUS_LOCATION/etc/globus-job-manager.conf file, or on a per-service basis in the service entry file in the $GLOBUS_LOCATION/etc/grid-services directory.

3.1. Global Job Manager Configuration

To enable using the Scheduler Event Generator interface for all Job Managers started from a particular GLOBUS_LOCATION, add a line containing the string

-seg

to the file $GLOBUS_LOCATION/etc/globus-job-manager.conf.

EXAMPLE $GLOBUS_LOCATION/etc/globus-job-manager.conf:

- home "/opt/globus"
- globus-gatekeeper-host globus.yourdomain.org
- globus-gatekeeper-port 2119
- globus-gatekeeper-subject "/O=Grid/OU=Your Organization/CN=host/globus.yourdomain.org"
- globus-host-cputype i686
- globus-host-manufacturer pc
- globus-host-osname Linux
- globus-host-osversion 2.6.10
- save-logfile on_error
- state-file-dir /opt/globus/tmp/gram_job_state
- machine-type unknown
- seg

3.2. Scheduler-specific Job Manager Configuration

To enable using the Scheduler Event Generator interface for a particular Job Manager, add the string -seg to the end of the line in the service's file in the $GLOBUS_LOCATION/etc/grid-services directory.

EXAMPLE $GLOBUS_LOCATION/etc/grid-services/jobmanager-lsf:


⚠️ No SEG with Job Manager fork

The Job Manager fork does not support using the Scheduler Event Generator. If the -seg option is passed to a fork Job Manager, it will be ignored.
4. globus-job-manager-event-generator Configuration

The globus-job-manager-event-generator program requires that the globus_job_manager_event_generator setup package be installed and run. This setup package creates the $GLOBUS_LOCATION/etc/globus-job-manager-seg.conf file and initializes a directory to use for the scheduler logs.

By default, this setup script will create a configuration entry and directory for each scheduler installed on the system. For each scheduler to be handled by the globus-job-manager-event-generator program, there must be an entry in the file in the pattern:

<SCHEDULER_TYPE>_log_path=<PATH>

The two variable substitutions for this pattern are

SCHEDULER_TYPE

Must match the name of the scheduler-event-generator module for the scheduler (supported with GT 4.2 are lsf, condor, and pbs).

PATH

A path to a directory which must be writable by the account which will run the globus-job-manager-event-generator program for the SCHEDULER_TYPE, and world-readable (or readable for a group which contains all users which will run jobs via GRAM on that system). Each directory specified in the configuration file must be unique, or behavior is undefined.

EXAMPLE $GLOBUS_LOCATION/etc/globus-job-manager-seg.conf:

lsf_log_path=/opt/globus/var/globus-job-manager-seg-lsf
pbs_log_path=/opt/globus/var/globus-job-manager-seg-pbs

In this example, pbs and lsf schedulers are configured to use distinct subdirectories of the /opt/globus/var/ directory.

NOTE: For best performance, the log paths should be persistent across system reboots and mounted locally (non-net-worked).

NOTE: If a scheduler is added after the configuration step is done, administrator must rerun the setup package's script ($GLOBUS_LOCATION/setup/globus/setup-seg-job-manager.pl) or modify the configuration file and create the required directory with appropriate permissions.

5. Running the globus-job-manager-event-generator

The globus-job-manager-event-generator must be running when jobs are submitted to the Job Manager if job state changes are to be detected. One instance of the globus-job-manager-event-generator program must be running for each scheduler type which is handled by a Job Manager and configured to use the Scheduler Event Generator interface.
The command line for the globus-job-manager-event-generator program is `globus-job-manager-event-generator -s SCHEDULER_TYPE`. The SCHEDULER_TYPE should match the pattern of a log_path entry in the `$GLOBUS_LOCATION/etc/globus-job-manager-seg.conf` as described above.

NOTE: Remember, if your scheduler logs have restrictive permissions, then this script must be run by an account which has privileges to read those files.

NOTE: Old log files created by the globus-job-manager-event-generator script may be deleted if the administrator is certain that there are no jobs which will restart and require the old information. The names of the log files correspond to the dates when the events occurred. If there is at least one log file in the directory, then when the globus-job-manager-event-generator is restarted, it will resume logging from the timestamp of the newest event in that log file.

### 6. Troubleshooting the globus-job-manager-event-generator

**PROBLEM:** The globus-job-manager-event-generator program terminates immediately with the output:

```
Error: SCHEDULER not configured
```

**SOLUTION 1:** Make sure that you specified the correct name for the SCHEDULER module on the command line to the `globus-job-manager-event-generator` program

**SOLUTION 2:** There is no entry for lsf in the `$GLOBUS_LOCATION/etc/globus-job-manager-seg.conf` file. See the section on globus-job-manager-event-generator Configuration.

**PROBLEM:** The globus-job-manager-event-generator program terminates immediately with the output:

```
Fault: globus_xio: Operation was canceled
```

**SOLUTION:** The scheduler module selected on the command line could not be loaded by the Globus Scheduler Event Generator. Check that the name is correct, the module is installed, and the setup script for that module has been run.

**PROBLEM:** The Job Manager never receives any events from the scheduler.

**SOLUTION 1:** Verify that the directory specified in the `$GLOBUS_LOCATION/etc/globus-job-manager-seg.conf` for the scheduler exists, is writable by the account running the `globus-job-manager-event-generator` and is readable by the user account running the job manager.

**SOLUTION 2:** Verify that the globus-job-manager-event-generator program is running.

**SOLUTION 3:** Verify that the globus-job-manager-event-generator program has permissions to read the scheduler logs. To help diagnose this, run (as the account you wish to run the globus-job-manager-event-generator as) the command

```
$GLOBUS_LOCATION/libexec/globus-scheduler-event-generator -s <SCHEDULER_TYPE> -t 1
```

You should see events printed to the stdout of that process if it is working correctly.
Chapter 5. Audit Logging

Note

For more information, click here.
Chapter 6. Testing GRAM

First launch a gatekeeper by running the following (as yourself, not root):

```
% grid-proxy-init -debug -verify
  % globus-personal-gatekeeper -start
```

This command will output a contact string like `hostname:4589:/O=Grid/O=Globus/CN=Your Name`. Substitute that contact string for `<contact>` in the following command:

```
% globus-job-run <contact> /bin/date
```

You should see the current date and time. At this point you can stop the personal gatekeeper and destroy your proxy with:

```
% globus-personal-gatekeeper -killall
  % grid-proxy-destroy
```

Please note that the above instructions are just for testing, and do not install a fully functioning gatekeeper on your machine for everyone to use. Installing a system-level gatekeeper for everyone to use will be covered in the configuration section of this guide.
Chapter 7. Usage statistics collection by the Globus Alliance

No usage statistic package is sent after the completion of a job like it's done in WS-GRAM (see here\(^1\)).

\(^1\) ../wsgram/admin-index.html#s-wsgram-admin-usage
# Table of Contents

1. Introduction .................................................................................................................................... 1  
2. Commandline Tools ......................................................................................................................... 2  
3. Resource Specification Language (RSL) .......................................................................................... 3  
4. RSL Attributes .................................................................................................................................. 4  
5. Job Execution Environment ....................................................................................................... 5  
6. GRAM Error Codes ....................................................................................................................... 6  
7. Usage Scenarios ............................................................................................................................. 15  
8. Troubleshooting ............................................................................................................................ 16
List of Tables

6.1. Gram Error Codes .......................................................................................................................... 7
Chapter 1. Introduction

GRAM services provide secure job submission to many types of job schedulers for users who have the right to access a job hosting resource in a Grid environment. The existence of a valid proxy is in fact required for job submission. All GRAM job submission options are supported transparently through the embedded request document input. In fact, the job startup is done by submitting a client-side provided job description to the GRAM services. This submission can be made by end-users with the GRAM command-line tools.
Chapter 2. Commandline Tools

Gram Clients\(^1\)

\(^1\) http://www.globus.org/toolkit/docs/2.4/admin/guide-user.html#gram
Chapter 3. Resource Specification Language (RSL)

Resource Specification Language (RSL)¹

¹ http://www.globus.org/toolkit/docs/2.4/gram/rsl_spec1.html
Chapter 4. RSL Attributes

RSL Attributes¹

¹ http://www.globus.org/api/c-globus-4.2.1/globus_gram_job_manager/html/globus_job_manager_rsl.html
Chapter 5. Job Execution Environment

Job Execution Environment\(^1\)

\(^1\) http://www.globus.org/api/c-globus-4.2.1/globus_gram_job_manager/html/globus_gram_job_manager_job_execution_environment.html
Chapter 6. GRAM Error Codes
## Table 6.1. Gram Error Codes

<table>
<thead>
<tr>
<th>#</th>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>Success</td>
<td>Success</td>
</tr>
<tr>
<td>1</td>
<td>GLOBUS_GRAM_PROTOCOL_ERROR_PARAMETER_NOT_SUPPORTED</td>
<td>one of the RSL parameters is not supported</td>
</tr>
<tr>
<td>2</td>
<td>GLOBUS_GRAM_PROTOCOL_ERROR_INVALID_REQUEST</td>
<td>the RSL length is greater than the maximum allowed</td>
</tr>
<tr>
<td>3</td>
<td>GLOBUS_GRAM_PROTOCOL_ERROR_NO_RESOURCES</td>
<td>an I/O operation failed</td>
</tr>
<tr>
<td>4</td>
<td>GLOBUS_GRAM_PROTOCOL_ERROR_BAD_DIRECTORY</td>
<td>jobmanager unable to set default to the directory requested</td>
</tr>
<tr>
<td>5</td>
<td>GLOBUS_GRAM_PROTOCOL_ERROR_EXECUTABLE_NOT_FOUND</td>
<td>the executable does not exist</td>
</tr>
<tr>
<td>6</td>
<td>GLOBUS_GRAM_PROTOCOL_ERROR_INSUFFICIENT_FUNDS</td>
<td>of an unused INSUFFICIENT_FUNDS</td>
</tr>
<tr>
<td>7</td>
<td>GLOBUS_GRAM_PROTOCOL_ERROR_AUTHORIZATION</td>
<td>authentication with the remote server failed</td>
</tr>
<tr>
<td>8</td>
<td>GLOBUS_GRAM_PROTOCOL_ERROR_USER_CANCELLED</td>
<td>the user cancelled the job</td>
</tr>
<tr>
<td>9</td>
<td>GLOBUS_GRAM_PROTOCOL_ERROR_SYSTEM_CANCELLED</td>
<td>the system cancelled the job</td>
</tr>
<tr>
<td>10</td>
<td>GLOBUS_GRAM_PROTOCOL_ERROR_PROTOCOL_FAILED</td>
<td>data transfer to the server failed</td>
</tr>
<tr>
<td>11</td>
<td>GLOBUS_GRAM_PROTOCOL_ERROR_STDIN_NOT_FOUND</td>
<td>the stdin file does not exist</td>
</tr>
<tr>
<td>12</td>
<td>GLOBUS_GRAM_PROTOCOL_ERROR_CONNECTION_FAILED</td>
<td>the connection to the server failed (check host and port)</td>
</tr>
<tr>
<td>13</td>
<td>GLOBUS_GRAM_PROTOCOL_ERROR_INVALID_MAXTIME</td>
<td>the provided RSL 'maxtime' value is not an integer</td>
</tr>
<tr>
<td>14</td>
<td>GLOBUS_GRAM_PROTOCOL_ERROR_INVALID_COUNT</td>
<td>the provided RSL 'count' value is not an integer</td>
</tr>
<tr>
<td>15</td>
<td>GLOBUS_GRAM_PROTOCOL_ERROR_NULL_SPECIFICATION_TREE</td>
<td>the job manager received an invalid RSL</td>
</tr>
<tr>
<td>16</td>
<td>GLOBUS_GRAM_PROTOCOL_ERROR_JM_FAILED_ALLOW_ATTACH</td>
<td>the job manager failed in allowing others to make contact</td>
</tr>
<tr>
<td>17</td>
<td>GLOBUS_GRAM_PROTOCOL_ERROR_JOB_EXECUTION_FAILED</td>
<td>the job failed when the job manager attempted to run it</td>
</tr>
<tr>
<td>18</td>
<td>GLOBUS_GRAM_PROTOCOL_ERROR_INVALID_PARADYN</td>
<td>an invalid paradyn was specified</td>
</tr>
<tr>
<td>19</td>
<td>GLOBUS_GRAM_PROTOCOL_ERROR_INVALID_JOBTYPE</td>
<td>the provided RSL 'jobtype' value is invalid</td>
</tr>
<tr>
<td>20</td>
<td>GLOBUS_GRAM_PROTOCOL_ERROR_INVALID_GRAM_MYJOB</td>
<td>the provided RSL 'myjob' value is invalid</td>
</tr>
<tr>
<td>21</td>
<td>GLOBUS_GRAM_PROTOCOL_ERROR_BAD_SCRIPT_ARG_FILE</td>
<td>the job manager failed to locate an internal script argument file</td>
</tr>
<tr>
<td>#</td>
<td>Name</td>
<td>Description</td>
</tr>
<tr>
<td>----</td>
<td>----------------------------------------------------------------------</td>
<td>----------------------------------------------------------------------------</td>
</tr>
<tr>
<td>22</td>
<td>GLOBUS_GRAM_PROTOCOL_ERROR_ARG_FILE_CREATION_FAILED</td>
<td>the job manager failed to create an internal script argument file</td>
</tr>
<tr>
<td>23</td>
<td>GLOBUS_GRAM_PROTOCOL_ERROR_INVALID_JOBSTATE</td>
<td>the job manager detected an invalid job state</td>
</tr>
<tr>
<td>24</td>
<td>GLOBUS_GRAM_PROTOCOL_ERROR_INVALID_SCRIPT_REPLY</td>
<td>the job manager detected an invalid script response</td>
</tr>
<tr>
<td>25</td>
<td>GLOBUS_GRAM_PROTOCOL_ERROR_INVALID_SCRIPT_STATUS</td>
<td>the job manager detected an invalid script status</td>
</tr>
<tr>
<td>26</td>
<td>GLOBUS_GRAM_PROTOCOL_ERROR_JOBTYPE_NOT_SUPPORTED</td>
<td>the provided RSL 'jobtype' value is not supported by this job manager</td>
</tr>
<tr>
<td>27</td>
<td>GLOBUS_GRAM_PROTOCOL_ERROR_UNIMPLEMENTED</td>
<td>unused ERROR_UNIMPLEMENTED</td>
</tr>
<tr>
<td>28</td>
<td>GLOBUS_GRAM_PROTOCOL_ERROR_TEMP_SCRIPT_FILE_FAILED</td>
<td>the job manager failed to create an internal script submission file</td>
</tr>
<tr>
<td>29</td>
<td>GLOBUS_GRAM_PROTOCOL_ERROR_USER_PROXY_NOT_FOUND</td>
<td>the job manager cannot find the user proxy</td>
</tr>
<tr>
<td>30</td>
<td>GLOBUS_GRAM_PROTOCOL_ERROR_OPENING_USER_PROXY</td>
<td>the job manager failed to open the user proxy</td>
</tr>
<tr>
<td>31</td>
<td>GLOBUS_GRAM_PROTOCOL_ERROR_JOB_CANCEL_FAILED</td>
<td>the job manager failed to cancel the job as requested</td>
</tr>
<tr>
<td>32</td>
<td>GLOBUS_GRAM_PROTOCOL_ERROR_MALLOC_FAILED</td>
<td>system memory allocation failed</td>
</tr>
<tr>
<td>33</td>
<td>GLOBUS_GRAM_PROTOCOL_ERROR_DUCT_INIT_FAILED</td>
<td>the interprocess job communication initialization failed</td>
</tr>
<tr>
<td>34</td>
<td>GLOBUS_GRAM_PROTOCOL_ERROR_DUCT_LSP_FAILED</td>
<td>the interprocess job communication setup failed</td>
</tr>
<tr>
<td>35</td>
<td>GLOBUS_GRAM_PROTOCOL_ERROR_INVALID_HOST_COUNT</td>
<td>the provided RSL 'host count' value is invalid</td>
</tr>
<tr>
<td>36</td>
<td>GLOBUS_GRAM_PROTOCOL_ERROR_UNSUPPORTED_PARAMETER</td>
<td>one of the provided RSL parameters is unsupported</td>
</tr>
<tr>
<td>37</td>
<td>GLOBUS_GRAM_PROTOCOL_ERROR_INVALID_QUEUE</td>
<td>the provided RSL 'queue' parameter is invalid</td>
</tr>
<tr>
<td>38</td>
<td>GLOBUS_GRAM_PROTOCOL_ERROR_INVALID_PROJECT</td>
<td>the provided RSL 'project' parameter is invalid</td>
</tr>
<tr>
<td>39</td>
<td>GLOBUS_GRAM_PROTOCOL_ERROR_RSL_EVALUATION_FAILED</td>
<td>the provided RSL string includes variables that could not be identified</td>
</tr>
<tr>
<td>40</td>
<td>GLOBUS_GRAM_PROTOCOL_ERROR_BAD_RSL_ENVIRONMENT</td>
<td>the provided RSL 'environment' parameter is invalid</td>
</tr>
<tr>
<td>41</td>
<td>GLOBUS_GRAM_PROTOCOL_ERROR_DRYRUN</td>
<td>the provided RSL 'dryrun' parameter is invalid</td>
</tr>
<tr>
<td>42</td>
<td>GLOBUS_GRAM_PROTOCOL_ERROR_ZERO_LENGTH_RSL</td>
<td>the provided RSL is invalid (an empty string)</td>
</tr>
<tr>
<td>43</td>
<td>GLOBUS_GRAM_PROTOCOL_ERROR_STAGING_EXECUTABLE</td>
<td>the job manager failed to stage the executable</td>
</tr>
<tr>
<td>#</td>
<td>Name</td>
<td>Description</td>
</tr>
<tr>
<td>----</td>
<td>----------------------------------------------------------------------</td>
<td>--------------------------------------------------</td>
</tr>
<tr>
<td>44</td>
<td>GLOBUS_GRAM_PROTOCOL_ERROR_STAGING_STDIN</td>
<td>the job manager failed to stage the stdin file</td>
</tr>
<tr>
<td>45</td>
<td>GLOBUS_GRAM_PROTOCOL_ERROR_INVALID_JOB_MANAGER_TYPE</td>
<td>the requested job manager type is invalid</td>
</tr>
<tr>
<td>46</td>
<td>GLOBUS_GRAM_PROTOCOL_ERROR_BAD_ARGUMENTS</td>
<td>the provided RSL 'arguments' parameter is invalid</td>
</tr>
<tr>
<td>47</td>
<td>GLOBUS_GRAM_PROTOCOL_ERROR_GATEKEEPER_MISCONFIGURED</td>
<td>the gatekeeper failed to run the job manager</td>
</tr>
<tr>
<td>48</td>
<td>GLOBUS_GRAM_PROTOCOL_ERROR_BAD_RSL</td>
<td>the provided RSL could not be properly parsed</td>
</tr>
<tr>
<td>49</td>
<td>GLOBUS_GRAM_PROTOCOL_ERROR_VERSION_MISMATCH</td>
<td>there is a version mismatch between GRAM</td>
</tr>
<tr>
<td>50</td>
<td>GLOBUS_GRAM_PROTOCOL_ERROR_RSL_ARGUMENTS</td>
<td>the provided RSL 'arguments' parameter is invalid</td>
</tr>
<tr>
<td>51</td>
<td>GLOBUS_GRAM_PROTOCOL_ERROR_RSL_COUNT</td>
<td>the provided RSL 'count' parameter is invalid</td>
</tr>
<tr>
<td>52</td>
<td>GLOBUS_GRAM_PROTOCOL_ERROR_RSL_DIRECTORY</td>
<td>the provided RSL 'directory' parameter is invalid</td>
</tr>
<tr>
<td>53</td>
<td>GLOBUS_GRAM_PROTOCOL_ERROR_RSL_DRYRUN</td>
<td>the provided RSL 'dryrun' parameter is invalid</td>
</tr>
<tr>
<td>54</td>
<td>GLOBUS_GRAM_PROTOCOL_ERROR_RSL_ENVIRONMENT</td>
<td>the provided RSL 'environment' parameter is invalid</td>
</tr>
<tr>
<td>55</td>
<td>GLOBUS_GRAM_PROTOCOL_ERROR_RSL_EXECUTEABLE</td>
<td>the provided RSL 'executable' parameter is invalid</td>
</tr>
<tr>
<td>56</td>
<td>GLOBUS_GRAM_PROTOCOL_ERROR_RSL_HOST_COUNT</td>
<td>the provided RSL 'host_count' parameter is invalid</td>
</tr>
<tr>
<td>57</td>
<td>GLOBUS_GRAM_PROTOCOL_ERROR_RSL_JOBTYPE</td>
<td>the provided RSL 'jobtype' parameter is invalid</td>
</tr>
<tr>
<td>58</td>
<td>GLOBUS_GRAM_PROTOCOL_ERROR_RSL_MAXTIME</td>
<td>the provided RSL 'maxtime' parameter is invalid</td>
</tr>
<tr>
<td>59</td>
<td>GLOBUS_GRAM_PROTOCOL_ERROR_RSL_MYJOB</td>
<td>the provided RSL 'myjob' parameter is invalid</td>
</tr>
<tr>
<td>60</td>
<td>GLOBUS_GRAM_PROTOCOL_ERROR_RSL_PARADYN</td>
<td>the provided RSL 'paradyn' parameter is invalid</td>
</tr>
<tr>
<td>61</td>
<td>GLOBUS_GRAM_PROTOCOL_ERROR_RSL_PROJECT</td>
<td>the provided RSL 'project' parameter is invalid</td>
</tr>
<tr>
<td>62</td>
<td>GLOBUS_GRAM_PROTOCOL_ERROR_RSL_QUEUE</td>
<td>the provided RSL 'queue' parameter is invalid</td>
</tr>
<tr>
<td>63</td>
<td>GLOBUS_GRAM_PROTOCOL_ERROR_RSL_STDERR</td>
<td>the provided RSL 'stderr' parameter is invalid</td>
</tr>
<tr>
<td>64</td>
<td>GLOBUS_GRAM_PROTOCOL_ERROR_RSL_STDIN</td>
<td>the provided RSL 'stdin' parameter is invalid</td>
</tr>
<tr>
<td>65</td>
<td>GLOBUS_GRAM_PROTOCOL_ERROR_RSL.Stdout</td>
<td>the provided RSL 'stdout' parameter is invalid</td>
</tr>
<tr>
<td>#</td>
<td>Name</td>
<td>Description</td>
</tr>
<tr>
<td>----</td>
<td>-----------------------------------------------------------</td>
<td>--------------------------------------------------------------------</td>
</tr>
<tr>
<td>66</td>
<td>GLOBUS_GRAM_PROTOCOL_ERROR_OPENING_JOBMANAGER_SCRIPT</td>
<td>the job manager failed to locate an internal script</td>
</tr>
<tr>
<td>67</td>
<td>GLOBUS_GRAM_PROTOCOL_ERROR_CREATINGPIPE</td>
<td>the job manager failed on the system call pipe()</td>
</tr>
<tr>
<td>68</td>
<td>GLOBUS_GRAM_PROTOCOL_ERROR_FCNTL_FAILED</td>
<td>the job manager failed on the system call fcntl()</td>
</tr>
<tr>
<td>69</td>
<td>GLOBUS_GRAM_PROTOCOL_ERROR_STDOUT_FILENAME FAILED</td>
<td>the job manager failed to create the temporary stdout filename</td>
</tr>
<tr>
<td>70</td>
<td>GLOBUS_GRAM_PROTOCOL_ERROR_STDERR_FILENAME FAILED</td>
<td>the job manager failed to create the temporary stderr filename</td>
</tr>
<tr>
<td>71</td>
<td>GLOBUS_GRAM_PROTOCOL_ERROR_FORCING_EXECUTABLE</td>
<td>the job manager failed on the system call fork()</td>
</tr>
<tr>
<td>72</td>
<td>GLOBUS_GRAM_PROTOCOL_ERROR_EXECUTABLE_PERMISSIONS</td>
<td>the executable file permissions do not allow execution</td>
</tr>
<tr>
<td>73</td>
<td>GLOBUS_GRAM_PROTOCOL_ERROR_OPENING_STDOUT</td>
<td>the job manager failed to open stdout</td>
</tr>
<tr>
<td>74</td>
<td>GLOBUS_GRAM_PROTOCOL_ERROR_OPENING_STDERR</td>
<td>the job manager failed to open stderr</td>
</tr>
<tr>
<td>75</td>
<td>GLOBUS_GRAM_PROTOCOL_ERROR_OPENING_CACHE_USER_PROXY</td>
<td>the cache file could not be opened in order to relocate the user proxy</td>
</tr>
<tr>
<td>76</td>
<td>GLOBUS_GRAM_PROTOCOL_ERROR_OPENING_CACHE</td>
<td>cannot access cache files in ~/.globus/gass_cache, check permissions, quota, and disk space</td>
</tr>
<tr>
<td>77</td>
<td>GLOBUS_GRAM_PROTOCOL_ERROR_INSERTING_CLIENT_CONTACT</td>
<td>the job manager failed to insert the contact in the client contact list</td>
</tr>
<tr>
<td>78</td>
<td>GLOBUS_GRAM_PROTOCOL_ERROR_CLIENT_CONTACT_NOT_FOUND</td>
<td>the contact was not found in the job manager's client contact list</td>
</tr>
<tr>
<td>79</td>
<td>GLOBUS_GRAM_PROTOCOL_ERROR_CONTACTING_JOB_MANAGER</td>
<td>connecting to the job manager failed. Possible reasons: job terminated, invalid job contact, network problems, ...</td>
</tr>
<tr>
<td>80</td>
<td>GLOBUS_GRAM_PROTOCOL_ERROR_INVALID_ID_JOB_CONTACT</td>
<td>the syntax of the job contact is invalid</td>
</tr>
<tr>
<td>81</td>
<td>GLOBUS_GRAM_PROTOCOL_ERROR_UNDEFINED_EXE</td>
<td>the executable parameter in the RSL is undefined</td>
</tr>
<tr>
<td>82</td>
<td>GLOBUS_GRAM_PROTOCOL_ERROR_CONDOR_ARCH</td>
<td>the job manager service is misconfigured. condor arch undefined</td>
</tr>
<tr>
<td>83</td>
<td>GLOBUS_GRAM_PROTOCOL_ERROR_CONDOR_OS</td>
<td>the job manager service is misconfigured. condor os undefined</td>
</tr>
<tr>
<td>84</td>
<td>GLOBUS_GRAM_PROTOCOL_ERROR_RSL_MIN_MEMORY</td>
<td>the provided RSL 'min_memory' parameter is invalid</td>
</tr>
<tr>
<td>85</td>
<td>GLOBUS_GRAM_PROTOCOL_ERROR_RSL_MAX_MEMORY</td>
<td>the provided RSL 'max_memory' parameter is invalid</td>
</tr>
<tr>
<td>86</td>
<td>GLOBUS_GRAM_PROTOCOL_ERROR_INVALID_MIN_MEMORY</td>
<td>the RSL 'min_memory' value is not zero or greater</td>
</tr>
<tr>
<td>#</td>
<td>Name</td>
<td>Description</td>
</tr>
<tr>
<td>----</td>
<td>-----------------------------------------------------------</td>
<td>-----------------------------------------------------------------------------</td>
</tr>
<tr>
<td>87</td>
<td>GLOBUS_GRAM_PROTOCOL_ERROR_INVALID_MAX_MEMORY</td>
<td>the RSL 'max_memory' value is not zero or greater</td>
</tr>
<tr>
<td>88</td>
<td>GLOBUS_GRAM_PROTOCOL_ERROR_HTTP_FRAME_FAILED</td>
<td>the creation of a HTTP message failed</td>
</tr>
<tr>
<td>89</td>
<td>GLOBUS_GRAM_PROTOCOL_ERROR_HTTP_UNFRAME_FAILED</td>
<td>parsing incoming HTTP message failed</td>
</tr>
<tr>
<td>90</td>
<td>GLOBUS_GRAM_PROTOCOL_ERROR_HTTP_PACK_FAILED</td>
<td>the packing of information into a HTTP message failed</td>
</tr>
<tr>
<td>91</td>
<td>GLOBUS_GRAM_PROTOCOL_ERROR_HTTP_UNPACK_FAILED</td>
<td>an incoming HTTP message did not contain the expected information</td>
</tr>
<tr>
<td>92</td>
<td>GLOBUS_GRAM_PROTOCOL_ERROR_INVALID_ID_JOB_QUERY</td>
<td>the job manager does not support the service that the client requested</td>
</tr>
<tr>
<td>93</td>
<td>GLOBUS_GRAM_PROTOCOL_ERROR_SERVICE_NOT_FOUND</td>
<td>the gatekeeper failed to find the requested service</td>
</tr>
<tr>
<td>94</td>
<td>GLOBUS_GRAM_PROTOCOL_ERROR_JOB_QUERY_DENIAL</td>
<td>the job manager does not accept any new requests (shutting down)</td>
</tr>
<tr>
<td>95</td>
<td>GLOBUS_GRAM_PROTOCOL_ERROR_CALLBACK_NOT_FOUND</td>
<td>the client failed to close the listener associated with the callback URL</td>
</tr>
<tr>
<td>96</td>
<td>GLOBUS_GRAM_PROTOCOL_ERROR_BAD_GATEKEEPER_CONTACT</td>
<td>the gatekeeper contact cannot be parsed</td>
</tr>
<tr>
<td>97</td>
<td>GLOBUS_GRAM_PROTOCOL_ERROR_POE_NOT_FOUND</td>
<td>the job manager could not find the 'poe' command</td>
</tr>
<tr>
<td>98</td>
<td>GLOBUS_GRAM_PROTOCOL_ERROR_MPIRUN_NOT_FOUND</td>
<td>the job manager could not find the 'mpirun' command</td>
</tr>
<tr>
<td>99</td>
<td>GLOBUS_GRAM_PROTOCOL_ERROR_RSL_START_TIME</td>
<td>the provided RSL 'start_time' parameter is invalid</td>
</tr>
<tr>
<td>100</td>
<td>GLOBUS_GRAM_PROTOCOL_ERROR_RSL_RESERVATION_HANDLE</td>
<td>the provided RSL 'reservation_handle' parameter is invalid</td>
</tr>
<tr>
<td>101</td>
<td>GLOBUS_GRAM_PROTOCOL_ERROR_RSL_MAX_WALL_TIME</td>
<td>the provided RSL 'max_wall_time' parameter is invalid</td>
</tr>
<tr>
<td>102</td>
<td>GLOBUS_GRAM_PROTOCOL_ERROR_INVALID_MAX_WALL_TIME</td>
<td>the RSL 'max_wall_time' value is not zero or greater</td>
</tr>
<tr>
<td>103</td>
<td>GLOBUS_GRAM_PROTOCOL_ERROR_RSL_MAX_CPU_TIME</td>
<td>the provided RSL 'max_cpu_time' parameter is invalid</td>
</tr>
<tr>
<td>104</td>
<td>GLOBUS_GRAM_PROTOCOL_ERROR_INVALID_MAX_CPU_TIME</td>
<td>the RSL 'max_cpu_time' value is not zero or greater</td>
</tr>
<tr>
<td>105</td>
<td>GLOBUS_GRAM_PROTOCOL_ERROR_JM_SCRIPT_NOT_FOUND</td>
<td>the job manager is misconfigured, a scheduler script is missing</td>
</tr>
<tr>
<td>106</td>
<td>GLOBUS_GRAM_PROTOCOL_ERROR_JM_SCRIPT_PERMISSIONS</td>
<td>the job manager is misconfigured, a scheduler script has invalid permissions</td>
</tr>
<tr>
<td>107</td>
<td>GLOBUS_GRAM_PROTOCOL_ERROR_SIGNALING_JOB</td>
<td>the job manager failed to signal the job</td>
</tr>
<tr>
<td>108</td>
<td>GLOBUS_GRAM_PROTOCOL_ERROR_UNKNOWN_SIGNAL_TYPE</td>
<td>the job manager did not recognize/support the signal type</td>
</tr>
<tr>
<td>#</td>
<td>Name</td>
<td>Description</td>
</tr>
<tr>
<td>----</td>
<td>--------------------------------------------------------</td>
<td>------------------------------------------------------------------------------</td>
</tr>
<tr>
<td>109</td>
<td>GLOBUS_GRAM_PROTOCOL_ERROR_GETTING_JOBID</td>
<td>the job manager failed to get the job id from the local scheduler</td>
</tr>
<tr>
<td>110</td>
<td>GLOBUS_GRAM_PROTOCOL_ERROR_WAITING_FOR_COMMIT</td>
<td>the job manager is waiting for a commit signal</td>
</tr>
<tr>
<td>111</td>
<td>GLOBUS_GRAM_PROTOCOL_ERROR_COMMIT_TIMED_OUT</td>
<td>the job manager timed out while waiting for a commit signal</td>
</tr>
<tr>
<td>112</td>
<td>GLOBUS_GRAM_PROTOCOL_ERROR_RSL_SAVE_STATE</td>
<td>the provided RSL 'save_state' parameter is invalid</td>
</tr>
<tr>
<td>113</td>
<td>GLOBUS_GRAM_PROTOCOL_ERROR_RSL_RESTART</td>
<td>the provided RSL 'restart' parameter is invalid</td>
</tr>
<tr>
<td>114</td>
<td>GLOBUS_GRAM_PROTOCOL_ERROR_RSL_TWO_PHASE_COMMIT</td>
<td>the provided RSL 'two_phase' parameter is invalid</td>
</tr>
<tr>
<td>115</td>
<td>GLOBUS_GRAM_PROTOCOL_ERROR_INVALID_ID_TWO_PHASE_COMMIT</td>
<td>the RSL 'two_phase' value is not zero or greater</td>
</tr>
<tr>
<td>116</td>
<td>GLOBUS_GRAM_PROTOCOL_ERROR_RSL_STDOUT_POSITION</td>
<td>the provided RSL 'stdout_position' parameter is invalid</td>
</tr>
<tr>
<td>117</td>
<td>GLOBUS_GRAM_PROTOCOL_ERROR_INVALID_ID_STDOUT_POSITION</td>
<td>the RSL 'stdout_position' value is not zero or greater</td>
</tr>
<tr>
<td>118</td>
<td>GLOBUS_GRAM_PROTOCOL_ERROR_RSL_STDERR_POSITION</td>
<td>the provided RSL 'stderr_position' parameter is invalid</td>
</tr>
<tr>
<td>119</td>
<td>GLOBUS_GRAM_PROTOCOL_ERROR_INVALID_ID_STDERR_POSITION</td>
<td>the RSL 'stderr_position' value is not zero or greater</td>
</tr>
<tr>
<td>120</td>
<td>GLOBUS_GRAM_PROTOCOL_ERROR_RESTART_FAILED</td>
<td>the job manager restart attempt failed</td>
</tr>
<tr>
<td>121</td>
<td>GLOBUS_GRAM_PROTOCOL_ERROR_NO_STATE_FILE</td>
<td>the job state file doesn't exist</td>
</tr>
<tr>
<td>122</td>
<td>GLOBUS_GRAM_PROTOCOL_ERROR_READING_STATE_FILE</td>
<td>could not read the job state file</td>
</tr>
<tr>
<td>123</td>
<td>GLOBUS_GRAM_PROTOCOL_ERROR_WRITING_STATE_FILE</td>
<td>could not write the job state file</td>
</tr>
<tr>
<td>124</td>
<td>GLOBUS_GRAM_PROTOCOL_ERROR_OLD_JM_ALIVE</td>
<td>old job manager is still alive</td>
</tr>
<tr>
<td>125</td>
<td>GLOBUS_GRAM_PROTOCOL_ERROR_TTL_EXPIRED</td>
<td>job manager state file TTL expired</td>
</tr>
<tr>
<td>126</td>
<td>GLOBUS_GRAM_PROTOCOL_ERROR_SUBMIT_UNKNOWN</td>
<td>it is unknown if the job was submitted</td>
</tr>
<tr>
<td>127</td>
<td>GLOBUS_GRAM_PROTOCOL_ERROR_RSL_REMOTE_IO_URL</td>
<td>the provided RSL 'remote_io_url' parameter is invalid</td>
</tr>
<tr>
<td>128</td>
<td>GLOBUS_GRAM_PROTOCOL_ERROR_WRITING_REMOTE_IO_URL</td>
<td>could not write the remote io url file</td>
</tr>
<tr>
<td>129</td>
<td>GLOBUS_GRAM_PROTOCOL_ERROR_STDIO_SIZE</td>
<td>the standard output/error size is different</td>
</tr>
<tr>
<td>130</td>
<td>GLOBUS_GRAM_PROTOCOL_ERROR_JM_STOPPED</td>
<td>the job manager was sent a stop signal (job is still running)</td>
</tr>
<tr>
<td>#</td>
<td>Name</td>
<td>Description</td>
</tr>
<tr>
<td>-----</td>
<td>--------------------------------------------------</td>
<td>----------------------------------------------------------</td>
</tr>
<tr>
<td>131</td>
<td>GLOBUS_GRAM_PROTOCOL_ERROR_USER_PROXY_EXPIRED</td>
<td>the user proxy expired (job is still running)</td>
</tr>
<tr>
<td>132</td>
<td>GLOBUS_GRAM_PROTOCOL_ERROR_JOB_UNSUBMITTED</td>
<td>the job was not submitted by original jobmanager</td>
</tr>
<tr>
<td>133</td>
<td>GLOBUS_GRAM_PROTOCOL_ERROR_INVALID_COMMIT</td>
<td>the job manager is not waiting for that commit signal</td>
</tr>
<tr>
<td>134</td>
<td>GLOBUS_GRAM_PROTOCOL_ERROR_RSL_SCHEDULER_SPECIFIC</td>
<td>the provided RSL scheduler specific parameter is invalid</td>
</tr>
<tr>
<td>135</td>
<td>GLOBUS_GRAM_PROTOCOL_ERROR_STAGE_IN_FAILED</td>
<td>the job manager could not stage in a file</td>
</tr>
<tr>
<td>136</td>
<td>GLOBUS_GRAM_PROTOCOL_ERROR_INVALID_SCRATCH</td>
<td>the scratch directory could not be created</td>
</tr>
<tr>
<td>137</td>
<td>GLOBUS_GRAM_PROTOCOL_ERROR_RSL_CACHE</td>
<td>the provided 'gass_cache' parameter is invalid</td>
</tr>
<tr>
<td>138</td>
<td>GLOBUS_GRAM_PROTOCOL_ERROR_INVALID_SUBMIT_ATTRIBUTE</td>
<td>the RSL contains attributes which are not valid for job submission</td>
</tr>
<tr>
<td>139</td>
<td>GLOBUS_GRAM_PROTOCOL_ERROR_INVALID_STDIO_UPDATE_ATTRIBUTE</td>
<td>the RSL contains attributes which are not valid for stdio update</td>
</tr>
<tr>
<td>140</td>
<td>GLOBUS_GRAM_PROTOCOL_ERROR_INVALID_RESTART_ATTRIBUTE</td>
<td>the RSL contains attributes which are not valid for job restart</td>
</tr>
<tr>
<td>141</td>
<td>GLOBUS_GRAM_PROTOCOL_ERROR_RSL_FILE_STAGE_IN</td>
<td>the provided RSL 'file_stage_in' parameter is invalid</td>
</tr>
<tr>
<td>142</td>
<td>GLOBUS_GRAM_PROTOCOL_ERROR_RSL_FILE_STAGE_IN_SHARED</td>
<td>the provided RSL 'file_stage_in_shared' parameter is invalid</td>
</tr>
<tr>
<td>143</td>
<td>GLOBUS_GRAM_PROTOCOL_ERROR_RSL_FILE_STAGE_OUT</td>
<td>the provided RSL 'file_stage_out' parameter is invalid</td>
</tr>
<tr>
<td>144</td>
<td>GLOBUS_GRAM_PROTOCOL_ERROR_RSL_GASS_CACHE</td>
<td>the provided RSL 'gass_cache' parameter is invalid</td>
</tr>
<tr>
<td>145</td>
<td>GLOBUS_GRAM_PROTOCOL_ERROR_RSL_FILE_CLEANUP</td>
<td>the provided RSL 'file_cleanup' parameter is invalid</td>
</tr>
<tr>
<td>146</td>
<td>GLOBUS_GRAM_PROTOCOL_ERROR_RSL_SCRATCH</td>
<td>the provided RSL 'scratch_dir' parameter is invalid</td>
</tr>
<tr>
<td>147</td>
<td>GLOBUS_GRAM_PROTOCOL_ERROR_INVALID_SCHEDULER_SPECIFIC</td>
<td>the provided scheduler-specific RSL parameter is invalid</td>
</tr>
<tr>
<td>148</td>
<td>GLOBUS_GRAM_PROTOCOL_ERROR_UNDEFINED_ATTRIBUTE</td>
<td>a required RSL attribute was not defined in the RSL spec</td>
</tr>
<tr>
<td>149</td>
<td>GLOBUS_GRAM_PROTOCOL_ERROR_INVALID_CACHE</td>
<td>the gass_cache attribute points to an invalid cache directory</td>
</tr>
<tr>
<td>150</td>
<td>GLOBUS_GRAM_PROTOCOL_ERROR_INVALID_SAVE_STATE</td>
<td>the provided RSL 'save_state' parameter has an invalid value</td>
</tr>
<tr>
<td>151</td>
<td>GLOBUS_GRAM_PROTOCOL_ERROR_OPENING_VALIDATION_FILE</td>
<td>the job manager could not open the RSL attribute validation file</td>
</tr>
<tr>
<td>152</td>
<td>GLOBUS_GRAM_PROTOCOL_ERROR_READING_VALIDATION_FILE</td>
<td>the job manager could not read the RSL attribute validation file</td>
</tr>
<tr>
<td>#</td>
<td>Name</td>
<td>Description</td>
</tr>
<tr>
<td>-----</td>
<td>-----------------------------------------------------------</td>
<td>------------------------------------------------------------</td>
</tr>
<tr>
<td>153</td>
<td>GLOBUS_GRAM_PROTOCOL_ERROR_RSL_PROXY_TIMEOUT</td>
<td>the provided RSL 'proxy_timeout' is invalid</td>
</tr>
<tr>
<td>154</td>
<td>GLOBUS_GRAM_PROTOCOL_ERROR_INVALID_ID_PROXY_TIMEOUT</td>
<td>the RSL 'proxy_timeout' value is not greater than zero</td>
</tr>
<tr>
<td>155</td>
<td>GLOBUS_GRAM_PROTOCOL_ERROR_STAGE_OUT_FAILED</td>
<td>the job manager could not stage out a file</td>
</tr>
<tr>
<td>156</td>
<td>GLOBUS_GRAM_PROTOCOL_ERROR_JOB&gt;Contact_NOT_FOUND</td>
<td>the job contact string does not match any which the job manager is handling</td>
</tr>
<tr>
<td>157</td>
<td>GLOBUS_GRAM_PROTOCOL_ERROR_DELEGATION_FAILED</td>
<td>proxy delegation failed</td>
</tr>
<tr>
<td>158</td>
<td>GLOBUS_GRAM_PROTOCOL_ERROR_LOCKING_STATE_LOCK_FILE</td>
<td>the job manager could not lock the state lock file</td>
</tr>
<tr>
<td>159</td>
<td>GLOBUS_GRAM_PROTOCOL_ERROR_INVALID_ID_ATTR</td>
<td>an invalid globus_io_clientattr_t was used.</td>
</tr>
<tr>
<td>160</td>
<td>GLOBUS_GRAM_PROTOCOL_ERROR_NULL_PARAMETER</td>
<td>an null parameter was passed to the gram library</td>
</tr>
<tr>
<td>161</td>
<td>GLOBUS_GRAM_PROTOCOL_ERROR_STILL_STREAMING</td>
<td>the job manager is still streaming output</td>
</tr>
<tr>
<td>162</td>
<td>GLOBUS_GRAM_PROTOCOL_ERROR_AUTHORIZATION_DENIED</td>
<td>the authorization system denied the request</td>
</tr>
<tr>
<td>163</td>
<td>GLOBUS_GRAM_PROTOCOL_ERROR_AUTHORIZATION_SYSTEM_FAILURE</td>
<td>the authorization system reported a failure</td>
</tr>
<tr>
<td>164</td>
<td>GLOBUS_GRAM_PROTOCOL_ERROR_AUTHORIZATION_DENIED_JOB_ID</td>
<td>the authorization system denied the request - invalid job id</td>
</tr>
<tr>
<td>165</td>
<td>GLOBUS_GRAM_PROTOCOL_ERROR_AUTHORIZATION_DENIED_EXECUTABLE</td>
<td>the authorization system denied the request - not authorized to run the specified executable</td>
</tr>
<tr>
<td>166</td>
<td>GLOBUS_GRAM_PROTOCOL_ERROR_RSL_USER_NAME</td>
<td>the provided RSL 'user_name' parameter is invalid.</td>
</tr>
<tr>
<td>167</td>
<td>GLOBUS_GRAM_PROTOCOL_ERROR_INVALID_ID_USER_NAME</td>
<td>the job is not running in the account named by the 'user_name' parameter</td>
</tr>
</tbody>
</table>
Chapter 7. Usage Scenarios

There is no content available at this time.
Chapter 8. Troubleshooting

Globus Toolkit 2.x Error FAQ¹

¹ http://www.globus.org/toolkit/docs/2.4/faq_errors.html
# Table of Contents

1. Introduction .................................................................................................................................... 1
2. Public interface ................................................................................................................................ 2
   1. Scheduler Event Generator ......................................................................................................... 2
   2. Client ..................................................................................................................................... 2
   3. GRAM Protocol ............................................................................................................................ 2
3. Tutorials ......................................................................................................................................... 3
   1. GRAM Job Manager Scheduler Tutorial ....................................................................................... 3
4. Usage scenarios .............................................................................................................................. 17
5. Debugging .................................................................................................................................... 18
6. Troubleshooting ............................................................................................................................. 19
7. Related Documentation .................................................................................................................... 20
Chapter 1. Introduction

There is no content available at this time.
Chapter 2. Public interface

1. Scheduler Event Generator

- General information¹
- SEG protocol²
- SEG API³
- SEG Tests⁴

2. Client

Client API⁵

3. GRAM Protocol

- General information⁶
- GRAM Protocol Functions⁷
- Job States⁸
- Signals⁹
- GRAM Errors¹⁰
- GRAM Protocol Message Format¹¹

³ http://www.globus.org/api/c-globus-4.2.1/globus_scheduler_event_generator/html/group__seg__api.html
⁷ http://www.globus.org/api/c-globus-4.2.1/globus_gram_protocol/html/group__globus__gram__protocol__functions.html
⁸ http://www.globus.org/api/c-globus-4.2.1/globus_gram_protocol/html/group__globus__gram__protocol__job__state.html
⁹ http://www.globus.org/api/c-globus-4.2.1/globus_gram_protocol/html/group__globus__gram__protocol__job__signal.html
¹⁰ http://www.globus.org/api/c-globus-4.2.1/globus_gram_protocol/html/group__globus__gram__protocol__error.html
Chapter 3. Tutorials

1. GRAM Job Manager Scheduler Tutorial

This tutorial describes the steps needed to build a GRAM Job Manager Scheduler interface package. The audience for this tutorial is a person interested in adding support for a new scheduler interface to GRAM. This tutorial will assume some familiarity with GTP, autoconf, automake, and Perl. As a reference point, this tutorial will refer to the code in the LSF Job Manager package.

1.1. Writing a Scheduler Interface

This section deals with writing the perl module which implements the interface between the GRAM job manager and the local scheduler. Consult the Job Manager Scheduler Interface section\(^1\) of this manual for a more detailed reference on the Perl modules which are used here.

The scheduler interface is implemented as a Perl module which is a subclass of the Globus::GRAM::JobManager module. Its name must match the scheduler type string used when the service is installed. For the LSF scheduler, the name is *lsf*, so the module name is *Globus::GRAM::JobManager::lsf* and it is stored in the file *lsf.pm*. Though there are several methods in the JobManager interface, they only ones which absolutely need to be implemented in a scheduler module are submit, poll, cancel.

We'll begin by looking at the start of the lsf source module, lsf.in (the transformation to lsf.pm happens when the setup script is run. To begin the script, we import the GRAM support modules into the scheduler module's namespace, declare the module's namespace, and declare this module as a subclass of the Globus::GRAM::JobManager module. All scheduler packages will need to do this, substituting the name of the scheduler type being implemented where we see *lsf* below.

```perl
use Globus::GRAM::Error;
use Globus::GRAM::JobState;
use Globus::GRAM::JobManager;
use Globus::Core::Paths;
...

package Globus::GRAM::JobManager::lsf;

@ISA = qw(Globus::GRAM::JobManager);
```

Next, we declare any system-specific values which will be substituted when the setup package scripts are run. In the LSF case, we need the know the paths to a few programs which interact with the scheduler:

```perl
BEGIN
{
    $mpirun = '@MPIRUN@';
    $bsub   = '@BSUB@';
    $bjobs  = '@BJOBS@';
    $bkill  = '@BKILL@';

1#
1.1.1. Writing a constructor

For scheduler interfaces which need to setup some data before calling their other methods, they can overload the new method which acts as a constructor. Scheduler scripts which don't need any per-instance initialization will not need to provide a constructor, the Globus::GRAM::JobManager constructor will do the job.

If you do need to overloaded this method, be sure to call the JobManager module's constructor to allow it to do its initialization, as in this example:

```perl
sub new {
    my $proto = shift;
    my $class = ref($proto) || $proto;
    my $self = $class->SUPER::new(@_);
    ## Insert scheduler-specific startup code here
    return $self;
}
```

The job interface methods are called with only one argument, the scheduler object itself. That object contains the a Globus::GRAM::JobDescription object ($self->{JobDescription}) which includes the values from the RSL string associated with the request, as well as a few extra values:

**job_id**

The string returned as the value of JOB_ID in the return hash from submit. This won't be present for methods called before the job is submitted.

**uniq_id**

A string associated with this job request by the job manager program. It will be unique for all jobs on a host for all time.

**cache_tag**

The GASS cache tag related to this job submission. Files in the cache with this tag will be cleaned by the cleanup_cache() method.

Now, let's look at the methods which will interface to the scheduler.

1.1.2. Submitting Jobs

All scheduler modules must implement the submit method. This method is called when the job manager wishes to submit the job to the scheduler. The information in the original job request RSL string is available to the scheduler interface through the JobDescription data member of it's hash.
For most schedulers, this is the longest method to be implemented, as it must decide what to do with the job description, and convert them to something which the scheduler can understand.

We'll look at some of the steps in the LSF manager code to see how the scheduler interface is implemented.

In the beginning of the submit method, we'll get our parameters and look up the job description in the manager-specific object:

```perl
sub submit
{
    my $self = shift;
    my $description = $self->{JobDescription};
}
```

Then we will check for values of the job parameters that we will be handling. For example, this is how we check for a valid job type in the LSF scheduler interface:

```perl
if(defined($description->jobtype()))
{
    if($description->jobtype !~ /^(mpi|single|multiple)$/)
    {
        return Globus::GRAM::Error::JOBTYPE_NOT_SUPPORTED;
    }
    elsif($description->jobtype() eq 'mpi' && $mpirun eq "no")
    {
        return Globus::GRAM::Error::JOBTYPE_NOT_SUPPORTED;
    }
}
```

The lsf module supports most of the core RSL attributes, so it does more processing to determine what to do with the values in the job description.

Once we’ve inspected the JobDescription we’ll know what we need to tell the scheduler about so that it’ll start the job properly. For LSF, we will construct a job description script and pass that to the `bsub` command. This script is a bourne shell script with some special comments which LSF uses to decide what constraints to use when scheduling the job.

First, we'll open the new file, and write the file header:

```perl
$lsf_job_script = new IO::File($lsf_job_script_name, '>>');
$lsf_job_script->print<<EOF;
#! /bin/sh
#
# LSF batch job script built by Globus Job Manager
#
EOF
```

Then, we'll add some special comments to pass job constraints to LSF:
if(defined($queue))
{
    $lsf_job_script->print("#BSUB -q $queue\n");
}
if(defined($description->project()))
{
    $lsf_job_script->print("#BSUB -P " . $description->project() . "\n");
}

Before we start the executable in the LSF job description script, we will quote and escape the job's arguments so that they will be passed to the application as they were in the job submission RSL string:

At the end of the job description script, we actually run the executable named in the JobDescription. For LSF, we support a few different job types which require different startup commands. Here, we will quote and escape the strings in the argument list so that the values of the arguments will be identical to those in the initial job request string. For this Bourne-shell syntax script, we will double-quote each argument, and escaping the backslash (\), dollar-sign ($), double-quote ("), and single-quote (') characters. We will use this new string later in the script.

@arguments = $description->arguments();

foreach(@arguments)
{
    if(ref($_))
    {
        return Globus::GRAM::Error::RSL_ARGUMENTS;
    }
}
if($arguments[0])
{
    foreach(@arguments)
    {
        $_ =~ s/\%/\\%/g;
        $_ =~ s/$/$\$/g;
        $_ =~ s"/\\"/g;
        $_ =~ s`/\\`/g;
        $args .= '" . $_ . '" ';
    }
    else
    {
        $args = "";
    }
}

To end the LSF job description script, we will write the command line of the executable to the script. Depending on the job type of this submission, we will need to start either one or more instances of the executable, or the mpirun program which will start the job with the executable count in the JobDescription:

if($description->jobtype() eq "mpi")
Next, we submit the job to the scheduler. Be sure to close the script file before trying to redirect it into the submit command, or some of the script file may be buffered and things will fail in strange ways!

When the submission command returns, we check its output for the scheduler-specific job identifier. We will use this value to be able to poll or cancel the job.

The return value of the script should be either a GRAM error object or a reference to a hash of values. The Globus::GRAM::JobManager documentation lists the valid keys to that hash. For the submit method, we'll return the job identifier as the value of JOB_ID in the hash. If the scheduler returned a job status result, we could return that as well. LSF does not, so we'll just check for the job ID and return it, or if the job fails, we'll return an error object:

```perl
$lsf_job_script->close();
$job_id = (grep(/is submitted/, 
    split(/\n/,'$bsub < $lsf_job_script_name')))[0];
if($? == 0)
{
    $job_id =~ m/<([^>]*)>/;
    $job_id = $1;
    return { JOB_ID => $job_id };}
return Globus::GRAM::Error::INVALID_SCRIPT_REPLY;
}
```

That finishes the submit method. Most of the functionality for the scheduler interface is now written. We just have a few more (much shorter) methods to implement.
### 1.1.3. Polling Jobs

All scheduler modules must also implement the poll method. The purpose of this method is to check for updates of a job's status, for example, to see if a job has finished.

When this method is called, we'll get the job ID (which we returned from the submit method above) as well as the original job request information in the object's JobDescription. In the LSF script, we'll pass the job ID to the `bjobs` program, and that will return the job's status information. We'll compare the status field from the `bjobs` output to see what job state we should return.

If the job fails, and there is a way to determine that from the scheduler, then the script should return in its hash both

\[
\text{JOB\_STATE} \to \text{Globus::GRAM::JobState::FAILED}
\]

and

\[
\text{ERROR} \to \text{Globus::GRAM::Error::<ERROR\_TYPE>::value}
\]

Here's an excerpt from the LSF scheduler module implementation:

```perl
sub poll
{
    my $self = shift;
    my $description = $self->{JobDescription};
    my $job_id = $description->jobid();
    my $state;
    my $status_line;
    $self->log("polling job $job_id");
    # Get first line matching job id
    $_ = (grep(/$job_id/, `$bjobs $job_id 2>/dev/null`))[0];
    # Get 3th field (status)
    $state = (split(/\s+/))[2];
    if(/PEND/)
    {
        $state = Globus::GRAM::JobState::PENDING;
    }
    elsif(/USUSP|SSUSP|PSUSP/)
    {
        $state = Globus::GRAM::JobState::SUSPENDED
    }
    ...
    return {JOB\_STATE => $state};
}
```
1.1.4. Cancelling Jobs

All scheduler modules must also implement the cancel method. The purpose of this method is to cancel a running job.

As with the poll method described above, this method will be given the job ID as part of the JobDescription object held by the manager object. If the scheduler interface provides feedback that the job was cancelled successfully, then we can return a JOB_STATE change to the FAILED state. Otherwise we can return an empty hash reference, and let the poll method return the state change next time it is called.

To process a cancel in the LSF case, we will run the bkill command with the job ID.

```perl
sub cancel {
    my $self = shift;
    my $description = $self->{JobDescription};
    my $job_id = $description->jobid();

    $self->log("cancel job $job_id");
    system("$bkill $job_id >/dev/null 2>/dev/null");

    if($? == 0) {
        return { JOB_STATE => Globus::GRAM::JobState::FAILED }
    } else {
        return Globus::GRAM::Error::JOB_CANCEL_FAILED;
    }
}
```

1.1.5. End of the script

It is required that all perl modules return a non-zero value when they are parsed. To do this, make sure the last line of your module consists of:

```
1;
```

1.2. Setting up a Scheduler

Once we've written the job manager script, we need to get it installed so that the gatekeeper will be able to run our new service. We do this by writing a setup script. For LSF, we will write the script `setup-globus-job-manager-lsf.pl`, which we will list in the LSF package as the **Post_Install_Program**.

To set up the Gatekeeper service, our LSF setup script does the following:

1. Perform system-specific configuration.
2. Install the GRAM scheduler Perl module and register as a gatekeeper service.
3. **(Optional)** Install an RSL validation file defining extra scheduler-specific RSL attributes which the scheduler interface will support.
4. Update the GPT metadata to indicate that the job manager service has been set up.

1.2.1. System-Specific Configuration

First, our scheduler setup script probes for any system-specific information needed to interface with the local scheduler. For example, the LSF scheduler uses the `mpirun`, `bsub`, `bqueue`, `bjobs`, and `bkill` commands to submit, poll, and cancel jobs. We’ll assume that the administrator who is installing the package has these commands in their path. We’ll use an autoconf script to locate the executable paths for these commands and substitute them into our scheduler Perl module. In the LSF package, we have the `find-lsf-tools` script, which is generated during bootstrap by autoconf from the `find-lsf-tools.in` file:

```
## Required Prolog

AC_REVISION($Revision: 1.2 $)
AC_INIT(1sf.in)

# checking for the GLOBUS_LOCATION

if test "x$GLOBUS_LOCATION" = "x"; then
  echo "ERROR Please specify GLOBUS_LOCATION" >&2
  exit 1
fi

...  

## Check for optional tools, warn if not found

AC_PATH_PROG(MPIRUN, mpirun, no)
if test "$MPIRUN" = "no" ; then
  AC_MSG_WARN([Cannot locate mpirun])
fi

...  

## Check for required tools, error if not found

AC_PATH_PROG(BSUB, bsub, no)
if test "$BSUB" = "no" ; then
  AC_MSG_ERROR([Cannot locate bsub])
fi

...  

## Required epilog - update scheduler specific module

prefix='$(GLOBUS_LOCATION)'  
exec_prefix='$(GLOBUS_LOCATION)'  
libexecdir=${prefix}/libexec

AC_OUTPUT(
  lsf.pm:lsf.in
)
```
If this script exits with a non-zero error code, then the setup script propagates the error to the caller and exits without installing the service.

### 1.2.2. Registering as a Gatekeeper Service

Next, the setup script installs it's perl module into the perl library directory and registers an entry in the Globus Gatekeeper's service directory. The program `globus-job-manager-service`\(^2\) (distributed in the job manager program setup package) performs both of these tasks. When run, it expects the scheduler perl module to be located in the `$GLOBUS_LOCATION/setup/globus` directory.

```
$libexecdir/globus-job-manager-service -add -m lsf -s jobmanager-lsf;
```

### 1.2.3. Installing an RSL Validation File

If the scheduler script implements RSL attributes which are not part of the core set supported by the job manager, it must publish them in the job manager's data directory. If the scheduler script wants to set some default values of RSL attributes, it may also set those as the default values in the validation file.

The format of the validation file is described in the [RSL Validation File Format] section of the documentation. The validation file must be named `scheduler-type.rvf` and installed in the `$GLOBUS_LOCATION/share/globus_gram_job_manager` directory.

In the LSF setup script, we check the list of queues supported by the local LSF installation, and add a section of acceptable values for the `queue` RSL attribute:

```
open(VALIDATION_FILE, ">$ENV{GLOBUS_LOCATION}/share/globus_gram_job_manager/lsf.rvf");

# Customize validation file with queue info
open(BQUEUES, "bqueues -w | ");

# discard header
$_ = <BQUEUES>;
my @queues = ();

while(<BQUEUES>)
{
    chomp;
    $1 =~ m/\^\$(S+)/;
    push(@queues, $1);
}
close(BQUEUES);

if(@queues)
{
    print VALIDATION_FILE "Attribute: queue\n";
}
```

---


print VALIDATION_FILE join(" ", "Values:", @queues);
}
close VALIDATION_FILE;

1.2.4. Updating GPT Metadata

Finally, the setup package should create and finalize a Grid::GPT::Setup. The value of $package must be the same value as the gpt_package_metadata Name attribute in the package's metadata file. If either the new() or finish() methods fail, then it is considered good practice to clean up any files created by the setup script. From setup-globus-job-manager-lsf.pl:

my $metadata =
new Grid::GPT::Setup(
   package_name => "globus_gram_job_manager_setup_lsf");
...
$metadata->finish();

1.3. Packaging

Now that we've written a job manager scheduler interface, we'll package it using GPT to make it easy for our users to build and install. We'll start by gathering the different files we've written above into a single directory lsf.

• lsf.in
• find-lsf-tools.in
• setup-globus-job-manager.pl

1.3.1. Package Documentation

If there are any scheduler-specific options defined for this scheduler module, or if there any any optional setup items, then it is good to provide a documentation page which describes these. For LSF, we describe the changes since the last version of this package in the file globus_gram_job_manager_lsf.dox. This file consists of a doxygen mainpage. See www.doxygen.org for information on how to write documentation with that tool.

1.3.2. configure.in

Now, we'll write our configure.in script. This file is converted to the configure shell script by the bootstrap script below. Since we don't do any probes for compile-time tools or system characteristics, we just call the various initialization macros used by GPT, declare that we may provide doxygen documentation, and then output the files we need substitutions done on.

AC_REVISION($Revision: 1.2 $)
AC_INIT(Makefile.am)
GLOBUS_INIT
AM_PROG_LIBTOOL
Initialize the automake rules the last argument
AM_INIT_AUTOMAKE($GPT_NAME, $GPT_VERSION)

LAC_DOXYGEN("../","*.dox")

GLOBUS_FINALIZE

AC_OUTPUT(
    Makefile
    pkgdata/Makefile
    pkgdata/pkg_data_src.gpt
    doxygen/Doxyfile
    doxygen/Doxyfile-internal
    doxygen/Makefile
)

### 1.3.3. Package Metadata

Now we'll write our metadata file, and put it in the pkgdata subdirectory of our package. The important things to note in this file are the package name and version, the post_install_program, and the setup sections. These define how the package distribution will be named, what command will be run by `gpt-postinstall` when this package is installed, and what the setup dependencies will be written when the `Grid::GPT::Setup` object is **finalized**.

```xml
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE gpt_package_metadata SYSTEM "package.dtd">
<gpt_package_metadata Format_Version="0.02" Name="globus_gram_job_manager_setup_lsf">
    <Aging_Version Age="0" Major="1" Minor="0" />
    <Description>LSF Job Manager Setup</Description>
    <Functional_Group>ResourceManagement</Functional_Group>
    <Version_Stability Release="Beta" />
    <src_pkg>
        <With_Flavors build="no" />
        <Source_Setup_Dependency PkgType="pgm">
            <Setup_Dependency Name="globus_gram_job_manager_setup">
                <Version>
                    <Simple_Version Major="3" />
                </Version>
            </Setup_Dependency>
            <Setup_Dependency Name="globus_common_setup">
                <Version>
                    <Simple_Version Major="2" />
                </Version>
            </Setup_Dependency>
        </Source_Setup_Dependency>
    </src_pkg>
</gpt_package_metadata>
```

---

1.3.4. Automake Makefile.am

The automake Makefile.am for this package is short because there isn’t any compilation needed for this package. We just need to define what needs to be installed into which directory, and what source files need to be put into our source distribution. For the LSF package, we need to list the lsf.in, find-lsf-tools, and code>setup-globus-job-manager-lsf.pl scripts as files to be installed into the setup directory. We need to add those files plus our documentation source file to the EXTRA_LIST variable so that they will be included in source distributions. The rest of the lines in the file are needed for proper interaction with GPT.

```makefile
include $(top_srcdir)/globus_automake_pre
include $(top_srcdir)/globus_automake_pre_top

SUBDIRS = pkgdata doxygen

setup_SCRIPTS = \
    lsf.in \
    find-lsf-tools \
    setup-globus-job-manager-lsf.pl

EXTRA_DIST = $(setup_SCRIPTS) globus_gram_job_manager_lsf.dox
```

include $(top_srcdir)/globus_automake_post
include $(top_srcdir)/globus_automake_post_top
1.3.5. Bootstrap

The final piece we need to write for our package is the bootstrap script. This script is the standard bootstrap script for a globus package, with an extra line to generate the find-lsf-tools script using autoconf.

```sh
#!/bin/sh

# checking for the GLOBUS_LOCATION

if test "x$GLOBUS_LOCATION" = "x"; then
    echo "ERROR: Please specify GLOBUS_LOCATION" >&2
    exit 1
fi

if [ ! -f ${GLOBUS_LOCATION}/libexec/globus-bootstrap.sh ]; then
    echo "ERROR: Unable to locate ${GLOBUS_LOCATION}/libexec/globus-bootstrap.sh"
    echo "Please ensure that you have installed the globus-core package and"
    echo "that GLOBUS_LOCATION is set to the proper directory"
    exit
fi

. ${GLOBUS_LOCATION}/libexec/globus-bootstrap.sh

autoconf find-lsf-tools.in > find-lsf-tools
chmod 755 find-lsf-tools

exit 0
```

1.4. Building, Testing, and Debugging

With this all done, we can now try to build our new package. To do so, we'll need to run

```
% ./bootstrap
% ./gpt-build
```

If all of the files are written correctly, this should result in our package being installed into $GLOBUS_LOCATION. Once that is done, we should be able to run gpt-postinstall to configure our new job manager.

Now, we should be able to run the command

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

to start a gatekeeper configured to run a job manager using our new scripts. Running this will output a contact string (referred to as `contact-string` below), which we can use to connect to this new service. To do so, we'll run globus-job-run to submit a test job:

```
% globus-job-run <contact-string> /bin/echo Hello, LSF
```
Hello, LSF

1.4.1. When Things Go Wrong

If the test above fails, or more complicated job failures are occurring, then you'll have to debug your scheduler interface. Here are a few tips to help you out.

Make sure that your script is valid Perl. If you run

```perl
perl -I$GLOBUS_LOCATION/lib/perl \
   $GLOBUS_LOCATION/lib/perl/Globus/GRAM/JobManager/lsf.pm
```

You should get no output. If there are any diagnostics, correct them (in the lsf.in file), reinstall your package, and rerun the setup script.

Look at the Globus Toolkit Error FAQ\(^5\) and see if the failure is perhaps not related to your scheduler script at all.

Enable logging for the job manager. By default, the job manager is configured to log only when it notices a job failure. However, if your problem is that your script is not returning a failure code when you expect, you might want to enable logging always. To do this, modify the job manager configuration file to contain ”-save-logfile always” in place of ”-save-log on_error”.

Adding logging messages to your script: the JobManager object implements a log method, which allows you to write messages to the job manager log file. Do this as your methods are called to pinpoint where the error occurs.

Save the job description file when your script is run. This will allow you to run the globus-job-manager-script.pl interactively (or in the Perl debugger). To save the job description file, you can do

```perl
$self->{JobDescription}->save("/tmp/job_description.$$");
```

in any of the methods you've implemented.

\(^5\) http://www.globus.org/toolkit/docs/2.4/faq_errors.html
Chapter 4. Usage scenarios

There is no content available at this time.
Chapter 5. Debugging

There is no content available at this time.
Chapter 6. Troubleshooting

There is no content available at this time.
Chapter 7. Related Documentation

Information about other C-APIs of the GT can be found here\textsuperscript{1}

\textsuperscript{1}http://www.globus.org/api/c-globus-4.2.1/
GT 4.2.1 Release Notes: GRAM2

Table of Contents

1. Component Overview .................................................................................................................................. 1
2. Feature Summary ......................................................................................................................................... 1
3. Summary of Changes in GRAM2 .............................................................................................................. 2
4. Bug Fixes .................................................................................................................................................. 2
5. Known Problems ....................................................................................................................................... 2
6. Technology Dependencies ........................................................................................................................ 4
7. Tested Platforms ........................................................................................................................................ 4
8. Backward Compatibility Summary .......................................................................................................... 4
9. For More Information ................................................................................................................................ 4

<titleabbrev>Release Notes</titleabbrev>

1. Component Overview

The Grid Resource Allocation and Management (GRAM) service provides a single interface for requesting and using remote system resources for the execution of "jobs". The most common use of GRAM is remote application execution and control. It is designed to provide a uniform, flexible interface to job scheduling systems.

2. Feature Summary

Features new in release 4.2.1

- None

Other Supported Features

- Remote job execution and management
- Uniform and flexible interface to batch scheduling systems
- File staging before and after job execution
- Data streaming of stdout/err during jobs execution
- Recording of audit records to a directory, with a tool for uploading audit records to a database.
- An alternate job polling mechanism which uses the Scheduler Event Generator from GRAM4 in place of polling the scheduler directly.

Deprecated Features

- None
3. Summary of Changes in GRAM2

A number of patches from VDT have been applied in this release. Details about these patches is here\(^1\)

4. Bug Fixes

- **Bug 1538:** Gatekeeper log rotation and logging job accounting info
- **Bug 1550:** Fixes for race condition in job manager
- **Bug 1551:** Race condition in job manager
- **Bug 4213:** Patch to disable streaming
- **Bug 4771:** Date bug in job manager log file
- **Bug 6196:** GRAM2 Test Failures
- **Bug 6240:** Gatekeeper crash
- **Bug 6241:** GRAM fails to load large state fails
- **Bug 6302:** New gatekeeper feature to control service fork
- **Bug 6303:** Remove obsolete GRAM reporter in GT 4.2
- **Bug 6307:** Update path for fast polling file for condor
- **Bug 6358:** Changes in current Gram2 audit logging
- **Bug 6391:** Fix leaks, null pointer dereferences, and uninitialized memory reads in GRAM2

5. Known Problems

The following problems and limitations are known to exist for GRAM2 at the time of the 4.2.1 release:

5.1. Limitations

- [list limitations]

\(^1\) http://bugzilla.globus.org/bugzilla/show_bug.cgi?id=6192
\(^2\) http://bugzilla.globus.org/bugzilla/show_bug.cgi?id=1538
\(^3\) http://bugzilla.globus.org/bugzilla/show_bug.cgi?id=1550
\(^4\) http://bugzilla.globus.org/bugzilla/show_bug.cgi?id=1551
\(^5\) http://bugzilla.globus.org/bugzilla/show_bug.cgi?id=4213
\(^6\) http://bugzilla.globus.org/bugzilla/show_bug.cgi?id=4771
\(^7\) http://bugzilla.globus.org/bugzilla/show_bug.cgi?id=6196
\(^8\) http://bugzilla.globus.org/bugzilla/show_bug.cgi?id=6240
\(^9\) http://bugzilla.globus.org/bugzilla/show_bug.cgi?id=6241
\(^10\) http://bugzilla.globus.org/bugzilla/show_bug.cgi?id=6302
\(^11\) http://bugzilla.globus.org/bugzilla/show_bug.cgi?id=6303
\(^12\) http://bugzilla.globus.org/bugzilla/show_bug.cgi?id=6307
\(^13\) http://bugzilla.globus.org/bugzilla/show_bug.cgi?id=6358
\(^14\) http://bugzilla.globus.org/bugzilla/show_bug.cgi?id=6391
5.2. Known Bugs

- **Bug 1934:** Gatekeeper's syslog output cannot be controlled
- **Bug 2739:** Gatekeeper AuthZ/Gridmap Callout result logging
- **Bug 2741:** Catching SIGSEGV if dynamic loading of authorization modules fails
- **Bug 3373:** globus removes the temporary job directory before pbs writes back into it
- **Bug 4235:** globus-job-manager doesn't exit if the job fails.
- **Bug 4360:** globus-job-get-output bug prevents output delivery, PBS jobmanager affected. See also globus-job-clean, globus-job-cancel
- **Bug 4730:** MPI Jobs using Globus LSF in HP XC Cluster....
- **Bug 4747:** Need evaluation of patch to JobManager.pm
- **Bug 5143:** DONE state never reported for Condor jobs when using Condor-G grid monitor
- **Bug 5200:** GRAM (pre-webservices) from OSG 0.6.0 (VDT 1.6.1) has bad syslog format
- **Bug 5207:** GRAM SoftEnv extension bug
- **Bug 5272:** Invalid parsing of RSL file
- **Bug 5429:** stdin is lost when jobtype=multiple with jobmanager-lsf
- **Bug 5536:** Missing dependency in package globus_gram_job_manager_auditing
- **Bug 5537:** Missing dependency in package globus_gram_job_manager_auditing
- **Bug 5554:** GRAM2 4.0.5 setup-globus-job-manager-fork.pl silent failure
- **Bug 5556:** Audit directory setup instructions are insecure
- **Bug 5621:** gram2 credential refresh problems in 4.0.5
- **Bug 5775:** Gram status of old jobs incorrect on some lsf systems

15 http://bugzilla.globus.org/bugzilla/show_bug.cgi?id=1934
16 http://bugzilla.globus.org/bugzilla/show_bug.cgi?id=2739
17 http://bugzilla.globus.org/bugzilla/show_bug.cgi?id=2741
18 http://bugzilla.globus.org/bugzilla/show_bug.cgi?id=3373
19 http://bugzilla.globus.org/bugzilla/show_bug.cgi?id=4235
20 http://bugzilla.globus.org/bugzilla/show_bug.cgi?id=4360
21 http://bugzilla.globus.org/bugzilla/show_bug.cgi?id=4730
22 http://bugzilla.globus.org/bugzilla/show_bug.cgi?id=4747
23 http://bugzilla.globus.org/bugzilla/show_bug.cgi?id=5143
24 http://bugzilla.globus.org/bugzilla/show_bug.cgi?id=5200
25 http://bugzilla.globus.org/bugzilla/show_bug.cgi?id=5207
26 http://bugzilla.globus.org/bugzilla/show_bug.cgi?id=5272
27 http://bugzilla.globus.org/bugzilla/show_bug.cgi?id=5429
28 http://bugzilla.globus.org/bugzilla/show_bug.cgi?id=5536
29 http://bugzilla.globus.org/bugzilla/show_bug.cgi?id=5537
30 http://bugzilla.globus.org/bugzilla/show_bug.cgi?id=5554
31 http://bugzilla.globus.org/bugzilla/show_bug.cgi?id=5556
32 http://bugzilla.globus.org/bugzilla/show_bug.cgi?id=5621
33 http://bugzilla.globus.org/bugzilla/show_bug.cgi?id=5775
• **Bug 6184:** pbs.pm jobmanager fails jobs on qstat failure

• **Bug 6337:** Cannot configure globus to use different certificate path than default

### 6. Technology Dependencies

GRAM2 depends on the following GT components:

- C Common Libraries
- Non-WS Authentication and Authorization
- XIO
- GridFTP

GRAM2 depends on the following 3rd party software. The dependency exists only for the batch schedulers configured, thus making job submissions possible to the batch scheduling service:

- PBS
- Condor
- LSF
- other batch schedulers... (where the GRAM scheduler interface has been implemented)

### 7. Tested Platforms

Tested Platforms for GRAM2

- Linux

### 8. Backward Compatibility Summary

There have been no protocol changes since GT version 4.0.x

There have been no API changes since GT version 4.0.x

There have been no exception/error changes since GT version 4.0.x

There have been no schema changes since GT version 4.0.x

### 9. For More Information

See [GRAM2](http://bugzilla.globus.org/bugzilla/show_bug.cgi?id=6184) for more information about this component.

---

34 http://bugzilla.globus.org/bugzilla/show_bug.cgi?id=6184
35 http://bugzilla.globus.org/bugzilla/show_bug.cgi?id=6337
Audit Logging in GRAM2

Table of Contents

1. Overview ........................................................................................................................................ 1
2. Audit and Accounting Records ............................................................................................................ 2
3. GJID .............................................................................................................................................. 2
4. Accessing Audit and Accounting Information ........................................................................................ 2
5. For More Information ....................................................................................................................... 3
6. Functionality ................................................................................................................................... 3
7. Configuration to enable audit logging ................................................................................................. 3

1. Overview

GRAM2 includes mechanisms to provide access to audit and accounting information associated with jobs that GRAM2 submits to a local resource manager (LRM) such as PBS, LSF, or Condor.

Note

Remember, GRAM is not a local resource manager but rather a protocol engine for communicating with a range of different local resource managers using a standard message format.

In some scenarios, it is desirable to get general information about the usage of the underlying LRM, such as:

- What kinds of jobs were submitted via GRAM?
- How long did the processing of a job take?
- How many jobs were submitted by user X?

The following three use cases give a better overview of the meaning and purpose of auditing and accounting:

1. **Group Access.** A grid resource provider allows a remote service (e.g., a gateway or portal) to submit jobs on behalf of multiple users. The grid resource provider only obtains information about the identity of the remote submitting service and thus does not know the identity of the users for which the grid jobs are submitted. This group access is allowed under the condition that the remote service stores audit information so that, if and when needed, the grid resource provider can request and obtain information to track a specific job back to an individual user.

2. **Query Job Accounting.** A client that submits a job needs to be able to obtain, after the job has completed, information about the resources consumed by that job. In portal and gateway environments where many users submit many jobs against a single allocation, this per-job accounting information is needed soon after the job completes so that client-side accounting can be updated. Accounting information is sensitive and thus should only be released to authorized parties.

3. **Auditing.** In a distributed multi-site environment, it can be necessary to investigate various forms of suspected intrusion and abuse. In such cases, we may need to access an audit trail of the actions performed by a service. When accessing this audit trail, it will frequently be important to be able to relate specific actions to the user.

The audit record of a job is stored at the end of the processing cycle of a job - either when it is completely processed or failed.
2. Audit and Accounting Records

While audit and accounting records may be generated and stored by different entities in different contexts, we make the following assumptions in this chapter:

<table>
<thead>
<tr>
<th>Generated by:</th>
<th>Audit Records</th>
<th>Accounting Records</th>
</tr>
</thead>
<tbody>
<tr>
<td>LRM to which the GRAM service submits jobs</td>
<td>GRAM service</td>
<td>LRM to which the GRAM service submits jobs</td>
</tr>
<tr>
<td>Stored in:</td>
<td>Database, indexed by GJID</td>
<td>LRM, indexed by JID</td>
</tr>
<tr>
<td>Data that is stored:</td>
<td>See list below.</td>
<td>May include all information about the duration and resource-usage of a job</td>
</tr>
</tbody>
</table>

The audit record of each job contains the following data:

- **job_grid_id**: String representation of the resource EPR
- **local_job_id**: Job/process id generated by the scheduler
- **subject_name**: Distinguished name (DN) of the user
- **username**: Local username
- **idempotence_id**: Job id generated on the client-side
- **creation_time**: Date when the job resource is created
- **queued_time**: Date when the job is submitted to the scheduler
- **stage_in_grid_id**: String representation of the stageIn-EPR (RFT)
- **stage_out_grid_id**: String representation of the stageOut-EPR (RFT)
- **clean_up_grid_id**: String representation of the cleanUp-EPR (RFT)
- **globus_toolkit_version**: Version of the server-side GT
- **resource_manager_type**: Type of the resource manager (Fork, Condor, ...)
- **job_description**: Complete job description document
- **success_flag**: Flag that shows whether the job failed or finished successfully
- **finished_flag**: Flag that shows whether the job is already fully processed or still in progress

3. GJID

The GRAM2 service returns a "job contact" that is used to control the job. The job contact, by default, is already in an acceptable GJID format; therefore, the GRAM2 client and service do not need to convert it in any way.

4. Accessing Audit and Accounting Information

To connect the two sets of records, both audit and accounting, we require that GRAM records the JID in each audit record that it generates. It is then straightforward for an audit service to respond to requests such as "Give me the charge of the job with JID x" by:
1. first selecting matching record(s) from the audit table,

2. then using the local JID(s) to join to the accounting table of the LRM and access relevant accounting record(s).

We propose a Web Service interface for accessing audit and accounting information. OGSA-DAI$^1$ is a WSRF service that can create a single virtual database from two or more remote databases. In the future, other per-job information such as job performance data could be stored using the GJID or local JID as an index, and then made available in the same virtual database.

5. For More Information

The rest of this chapter focuses on how to configure GRAM2 to enable Audit Logging. A case study for TeraGrid can be read [here]$^2$, which also includes more information about how to use this data to get accounting information of a job, query the audit database for information via a Web Services interface, etc.

6. Functionality

Audit logging in GRAM2 is realized in the following way:

1. The job manager writes a file to disk for each job. This file contains the audit record. The format of an audit record file that is logged to the database is a comma-separated list of double-quoted strings.

2. The audit records are not inserted into the GRAM audit database directly. The job manager will, at final job termination (FAILED or DONE state), write a record to a unique file in a directory specified by a configuration file. These audit files must be uploaded by a program which can be called manually or be run periodically as a cron job. The program is a perl script and is located in ${GLOBUS_LOCATION}/libexec/globus-gram-audit and creates audit records in the configured database from the user audit files. Once the record is uploaded, the program will remove the audit file.

Here's an example on how a crontab entry must look like in order to run the script every 15 minutes:

0,15,30,45 * * * * ${GLOBUS_LOCATION}/libexec/globus-gram-audit

The script gets the necessary parameters to connect to the database from the configuration file ${GLOBUS_LOCATION}/etc/globus-job-manager-audit.conf, which is described below.

You may notice that this method is different than that used for GRAM4. GRAM4 writes audit records directly to the audit database. This is done because only a single account (the container account) may be given access to the DB. The container account is already trusted, so this is reasonable for GRAM4.

In GRAM2, the Job Manager process writes the audit records. However, this process runs under the user's account; opening up access to the audit database for each user is not acceptable.

Therefore, the Job Manager writes the audit record to a file. Then a single database upload program, running under the same GRAM4 container account (e.g. globus), can first verify that the owner of the GRAM2 audit file and the username field in the audit record match. This prevents users from interfering with other users' audit records.

7. Configuration to enable audit logging

Audit logging is turned off by default. To turn on Audit Logging, follow these steps:

---

$^1$ [http://www.globus.org/toolkit/docs/4.0/techpreview/ogsadaif/](http://www.globus.org/toolkit/docs/4.0/techpreview/ogsadaif/)

7.1. Configure Audit Record Directory

Add the following line to $GLOBUS_LOCATION/etc/globus-job-manager.conf:

```
-audit-directory <your desired audit record directory>
```

7.2. Create Audit Record Directory

Create the audit record directory specified in the above configuration file with the following permissions:

```
rwx-wsrwx
```

7.3. Database configuration

Edit $GLOBUS_LOCATION/etc/globus-job-manager-audit.conf to include the correct database connection parameters. For example:

```
DRIVERCLASS:com.mysql.jdbc.Driver
USERNAME:john
PASSWORD:foo
URL:jdbc:mysql://myhost/auditDatabase
AUDITVERSION:1
```

We support 3 database systems: MySQL, PostgreSQL, Derby. The following table gives an overview which values must be used for the parameters url and driverClassName in the above JNDI configuration for the various db systems. Derby is configured as the default DB system.

<table>
<thead>
<tr>
<th>DB system</th>
<th>driverClassName</th>
<th>url</th>
</tr>
</thead>
<tbody>
<tr>
<td>MySQL</td>
<td>com.mysql.jdbc.Driver</td>
<td>jdbc:mysql://HOST[:PORT]/auditDatabase</td>
</tr>
<tr>
<td>PostgreSQL</td>
<td>org.postgresql.Driver</td>
<td>jdbc:mysql://HOST[:PORT]/auditDatabase</td>
</tr>
<tr>
<td>Derby</td>
<td>org.apache.derby.jdbc.EmbeddedDriver</td>
<td>jdbc:derby:directory:PATH_TO_GLOBUS_LOCATION/var/gram/auditDatabase</td>
</tr>
</tbody>
</table>

Don't change the parameter AUDITVERSION for now. In future releases we'll support more than one version.

7.4. Creating the Audit Database

Audit records are stored in a database which must be set up once.

7.4.1. MySQL

The following describes how to set up the audit database in MySQL:

1. Create a database inside of MySQL

2. Grant necessary privileges to the account that will be used to upload the audit records in the audit. Typically the "globus" account.

3. Use the schema to create the table

```
host:~ feller$ mysql -u root -p
Enter password:
```
Welcome to the MySQL monitor. Commands end with ; or \g.
Your MySQL connection id is 16
Server version: 5.0.37 MySQL Community Server (GPL)

Type 'help;' or '\h' for help. Type '\c' to clear the buffer.

mysql> create database auditDatabase;
Query OK, 1 row affected (0.09 sec)

mysql> GRANT ALL ON auditDatabase.* to globus@localhost identified by "foo";
Query OK, 0 rows affected (0.32 sec)

mysql> exit
Bye

host:~ feller$ mysql -u globus -p auditDatabase < ${GLOBUS_LOCATION}/share/globus_wsrf_gram/gram_audit_schema_mysql.sql
Enter password:
host:~ feller$

7.4.2. PostgreSQL

The following describes how to set up the audit database in PostgreSQL:

1. Create a database inside of PostgreSQL
2. Grant necessary privileges to the account that will be used to upload the audit records in the audit. Typically the "globus" account.
3. Use the schema to create the table:

   # Connect as postgres admin
   create database gt4audit
   create user gt4auditload with encrypted password '<password1>'
   create user gt4auditview with encrypted password '<password2>'
   
   \c gt4audit
   \i gram_audit_schema_postgres-8.0.sql
   grant insert on gram_audit_table to gt4auditload
   grant select on gram_audit_table to gt4auditview
   \q

   You must also update pg_hba.conf to allow connections from container host (pg_hba.conf configures client authentication and is stored in the database cluster's data directory):

   hostssl  gt4audit  gt4auditload  <containerhostip> 255.255.255.255 md5
   host    gt4audit  gt4auditload  <containerhostip> 255.255.255.255 md5
   hostssl  gt4audit  gt4auditview  <containerhostip> 255.255.255.255 md5
   host    gt4audit  gt4auditview  <containerhostip> 255.255.255.255 md5

7.4.3. Derby

During GT installation the Derby audit database is already created. It's location is ${GLOBUS_LOCATION}/var/gram/auditDatabase. If you ever have to create it manually, make sure that this directory does not exist and then call ${GLOBUS_LOCATION}/setup/globus/setup-gram-service-database. The user and password information can be found in ${GLOBUS_LOCATION}/share/globus_wsrf_gram/gram_audit_v1_schema_derby.sql.
1. Introduction

Starting with Globus Toolkit 4.1.0, the GRAM2 Job Manager supports using the Scheduler Event Generator (SEG) for obtaining job state information from the scheduler log files in place of running a poll command periodically. This has the effect in most situations of reducing the impact (in terms of CPU usage) of the job manager. This document describes how to configure and use this new feature.

2. Overview of Operation

The WS-GRAM Scheduler Event Generator (SEG) is a program which parses native log files generated by the schedulers supported by GRAM, and uses the information in them to issue events to stdout which are piped back to the WS-GRAM Job Manager service. This avoids the sometimes costly poll operation periodically done by the GRAM scheduler adapters.

For GRAM2, a program called `globus-job-manager-event-generator` runs the SEG and writes job state change records into a log file which all users can read. This log contains the minimal information about jobs to determine when they are queued, become active, and terminate. No user-specific or job-specific data is revealed in this log file. The GRAM2 Job Manager can be configured to use this log file as a source for job state change events. A single instance of the `globus-job-manager-event-generator` will be run for each scheduler on the system.

3. Configuration

3.1. Job Manager Configuration

By default, the job manager uses the GRAM2 script-based polling method. A new command line option (`-seg`) enables SEG-driven job state change notifications.

There are two ways to configure the job manager to use the scheduler event generator: globally, in the `$GLOBUS_LOCATION/etc/globus-job-manager.conf` file, or on a per-service basis in the service entry file in the `$GLOBUS_LOCATION/etc/grid-services` directory.

3.1.1. Global Job Manager Configuration

To enable using the Scheduler Event Generator interface for all Job Managers started from a particular GLOBUS_LOCATION, add a line containing the string
to the file $GLOBUS_LOCATION/etc/globus-job-manager.conf.

**Example 1. Example $GLOBUS_LOCATION/etc/globus-job-manager.conf**

```
-begin
-home "/opt/globus"
-globus-gatekeeper-host globus.yourdomain.org
-globus-gatekeeper-port 2119
-globus-gatekeeper-subject "/O=Grid/OU=Your Organization/CN=host/globus.yourdomain.org"
-globus-host-cputype i686
-globus-host-manufacturer pc
-globus-host-osname Linux
-globus-host-osversion 2.6.10
-save-logfile on_error
-state-file-dir /opt/globus/tmp/gram_job_state
-machine-type unknown
-end
```

### 3.1.2. Scheduler-specific Job Manager Configuration

To enable using the Scheduler Event Generator interface for a particular Job Manager, add the string `-seg` to the end of the line in the service's file in the `$GLOBUS_LOCATION/etc/grid-services` directory.

**Example 2. Example $GLOBUS_LOCATION/etc/grid-services/jobmanager-lsf**

```
```

⚠️ **Important**

The Job GRAM2 Job Manager does not support using the SEG for the fork scheduler. If the `-seg` option is passed to a fork Job Manager, it will be ignored.

### 3.2. `globus-job-manager-event-generator` Configuration

The `globus-job-manager-event-generator` program requires that the `globus_job_manager_event_generator` setup package be installed and run. This setup package creates the `$GLOBUS_LOCATION/etc/globus-job-manager-seg.conf` file and initializes a directory to use for the scheduler logs.

By default, this setup script will create a configuration entry and directory for each scheduler installed on the system. For each scheduler to be handled by the `globus-job-manager-event-generator` program, there must be an entry in the file in the pattern:

`SCHEDULER_TYPE_log_path=PATH`

The two variable substitutions for this pattern are

- **SCHEDULER_TYPE** Must match the name of the scheduler-event-generator module for the scheduler (supported with GT 4.1 are lsf, condor, and pbs).
- **PATH** A path to a directory which must be writable by the account which will run the `globus-job-manager-event-generator` program for the `SCHEDULER_TYPE`, and world-readable (or readable for a group which contains all users which will run jobs via GRAM on that system). Each directory specified in the configuration file must be unique, or behavior is undefined.
Example 3. Example $GLOBUS_LOCATION/etc/globus-job-manger-seg.conf

lsf_log_path=/opt/globus/var/globus-job-manager-seg-lsf
pbs_log_path=/opt/globus/var/globus-job-manager-seg-pbs

In this example, pbs and lsf schedulers are configured to use distinct subdirectories of the /opt/globus/var/ directory.

⚠️ Important

For best performance, the log paths should be persistent across system reboots and mounted locally (non-networked).

⚠️ Important

If a scheduler is added after the configuration step is done, administrator must rerun the setup package’s script ($GLOBUS_LOCATION/setup/globus/setup-seg-job-manager.pl) or modify the configuration file and create the log directory with appropriate permissions.

4. Running the
   globus-job-manager-event-generator

The globus-job-manager-event-generator script creates a log of all scheduler events related to a particular scheduler instance. This script was created for two purposes

- To avoid requiring that all GRAM users have the privileges to read the scheduler’s log file. Users may not be allowed read access to the scheduler’s log files. Since the Pre-WS GRAM Job Manager runs with the permissions of the user account, it may be unable to access the log files. Instead the globus-job-manager-event-generator program will run as a privileged user and then store job state change records in a file which GRAM2 users may access.

- To provide a simple format for the scheduler event generator logs so that the job manager will be able to quickly recover state information if the job manager is terminated and restarted. Some scheduler logs are difficult to parse, or inefficient for seeking to a particular timestamp (as is necessary for recovering job state change information). The data written by this script is easily locatable by date, and it is simple to remove old job information without compromising current job manager execution.

One instance of the globus-job-manager-event-generator must be running for each scheduler type to be implemented using the Scheduler Event Generator interface to receive job state changes. This program is located at $GLOBUS_LOCATION/sbin. The typical command line for this program is $GLOBUS_LOCATION/sbin/globus-job-manager-event-generator -s SCHEDULER_TYPE, where SCHEDULER_TYPE is the scheduler name of the SEG module which should be used to generate events (lsf, condor, pbs).

For example, to start the event generator program to monitor an LSF batch system:

$GLOBUS_LOCATION/sbin/globus-job-manager-event-generator -s lsf

⚠️ Important

If the globus-job-manager-event-generator is not running, no job state changes will be sent from any job manager program which is configured to use the SEG.
5. Troubleshooting

PROBLEM: The globus-job-manager-event-generator program terminates immediately with the output:

**Error: SCHEDULER not configured**

- Make sure that you specified the correct name for the `SCHEDULER` module on the command line to the `globus-job-manager-event-generator` program.

- There is no entry for `SCHEDULER` in the `$GLOBUS_LOCATION/etc/globus-job-manager-seg.conf` file. See the section on globus-job-manager-event-generator Configuration.

PROBLEM: The `globus-job-manager-event-generator` program terminates immediately with the output:

**Fault: globus_xio: Operation was canceled**

- The scheduler module selected on the command line could not be loaded by the SEG. Check that the name is correct, the module is installed, and the setup script for that module has been run.

PROBLEM: The Job Manager never receives any events from the scheduler.

- Verify that the directory specified in the `$GLOBUS_LOCATION/etc/globus-job-manager-seg.conf` for the scheduler exists, is writable by the account running the `globus-job-manager-event-generator` and is readable by the user account running the job manager.

- Verify that the `globus-job-manager-event-generator` program is running.

- Verify that the `globus-job-manager-event-generator` program has permissions to read the scheduler logs. To help diagnose this, run (as the account you wish to run the globus-job-manager-event-generator as) the command `$GLOBUS_LOCATION/libexec/globus-scheduler-event-generator -s SCHEDULER_TYPE -t 1` You should see events printed to the stdout of that process if it is working correctly.