

GT 4.2.1 GRAM4 : System Administrator's Guide

GT 4.2.1 GRAM4 : System Administrator's Guide

Introduction

This guide contains advanced configuration information for system administrators working with GRAM4. It provides references to information on procedures typically performed by system administrators, including installation, configuring, deploying, and testing the installation. It also describes additional prerequisites and host settings necessary for GRAM4 operation. Readers should be familiar with the [Key Concepts](#) and [Implementation Approach](#) for GRAM4 to understand the motivation for and interaction between the various deployed components.

Important

The information in this GRAM4 Admin Guide is in addition to the basic Globus Toolkit prerequisite, overview, installation, security configuration instructions in the [Installing GT 4.2.1](#). Read through this guide before continuing!

Table of Contents

1. Building and Installing	1
1. Prerequisites	1
2. Configuring	3
1. Typical Configuration	3
2. Non-default Configuration	4
3. Configuration Details	6
4. GRAM4 and GridFTP file system mapping	10
5. GRAM4 - RFT interaction	11
6. Defining a default local resource manager	11
7. Disabling an already installed local resource manager adapter	12
8. Job lifetime configuration	12
9. Registration with default WS MDS Index Service	13
10. Configuring thread-pools	14
3. Deploying	15
1. Deploying in Tomcat	15
4. Scalability and Performance Recommendations	16
1. Server-side Recommendations	16
5. Audit Logging	17
1. Overview	17
2. Audit and Accounting Records	17
3. Converting EPR to GRAM Service GJID	18
4. Accessing Audit and Accounting Information	18
5. For More Information	19
6. Configuration To Enable Audit Logging	19
6. Job Description Extensions Support	23
1. Requirements for Extensions Support	23
2. Supported Extension Constructs	23
3. Customizing Extensions Support	26
7. SoftEnv Support	28
1. Overview	28
2. Configuring SoftEnv Support	28
3. Dependencies	28
8. Testing	29
9. Security Considerations	30
10. Admin Debugging	31
1. Logging in Java WS Core	31
11. Troubleshooting	33
1. Troubleshooting tips	33
2. Java WS Core Errors	34
3. Errors	37
12. Usage statistics collection by the Globus Alliance	40
1. GRAM4-specific usage statistics	40
Glossary	41
Index	43

List of Tables

11.1. Java WS Core Errors	35
11.2. GRAM4 Errors	38

Chapter 1. Building and Installing

GRAM4 is built and installed as part of a default GT 4.2.1 installation. For basic installation instructions, see [Installing GT 4.2.1](#).

1. Prerequisites

1.1. Transport Level Security (TLS)

In order to use GRAM4, the container must be started with Transport Level security. The "-nosec" option should *not* be used with `globus-start-container`.

1.2. Functioning sudo

GRAM4 requires that the `sudo` command is installed and functioning on the service host where GRAM4 software will execute.

Authorization rules will need to be added to the `sudoers` file to allow the GRAM4 service account to execute (without a password) the `scheduler_adapter` in the accounts of authorized GRAM users. For sudo configuration details, see the [Configuring](#) section.

Platform Note: On AIX, sudo is not installed by default, but it is available as source and rpm here: [AIX 5L Toolbox for Linux Applications](#)¹

1.3. Local Scheduler

GRAM4 depends on a local mechanism for starting and controlling jobs. Included in the GRAM4 software is a Fork `scheduler`, which requires no special software installed to execute jobs on the local host. However, to enable GRAM4 to execute and manage jobs to a `batch scheduler`, the scheduler software must be installed and configured prior to configuring GRAM4.

1.4. Scheduler Adapter

GRAM4 depends on scheduler adapters to translate the GRAM4 `job description` document into commands understood by the local scheduler, as well as monitor the jobs.

Scheduler adapters included in the GT 4.2.1 release are: [PBS](#)², [Condor](#)³, [LSF](#)⁴

Third party schedulers adapters available for GT 4.2.1 are: [Sun Grid Engine](#)⁵

For configuration details, see "Configuring scheduler adapters" in the [Configuring](#) section.

¹ <http://www-1.ibm.com/servers/aix/products/aixos/linux/download.html>

² <http://www.openpbs.org/>

³ <http://www.cs.wisc.edu/condor/>

⁴ <http://www.platform.com/Products/Platform.LSF.Family/Platform.LSF/>

⁵ <http://www.lesc.ic.ac.uk/projects/SGE-GT4.html>

1.5. GridFTP

Though staging directives are processed by RFT (see next section), RFT uses GridFTP servers underneath to do the actual data movement. As a result, *there must be at least one GridFTP server that shares a file system with the execution nodes*. There is no special process to get staged files onto the execution node before the job executable is run. See the "Non-default GridFTP server" section of [Configuring GRAM4](#) for details on how to configure GRAM4 for your GridFTP servers used in your execution environment.

1.6. Reliable File Transfer Service (RFT)

GRAM4 depends on RFT to perform file staging and cleanup directives in a job description. For configuration details, see [System Administrator's Guide](#). *Important:* Jobs requesting these functions will fail if RFT is not properly setup.

Chapter 2. Configuring

1. Typical Configuration

1.1. Configuring sudo

When the credentials of the service account and the job submitter are different (multi user mode), then GRAM will prepend a call to sudo to the local adapter callout command. *Important:* If sudo is not configured properly, the command and thus job will fail.

As *root*, here are the two lines to add to the `/etc/sudoers` file for each `GLOBUS_LOCATION` installation, where `/opt/globus/GT4.2.1` should be replaced with the `GLOBUS_LOCATION` for your installation:

```
# Globus GRAM entries
globus ALL=(username1,username2)
NOPASSWD: /opt/globus/GT4.2.1/libexec/globus-gridmap-and-execute \
    -g /etc/grid-security/grid-mapfile \
    /opt/globus/GT4.2.1/libexec/globus-job-manager-script.pl *
globus ALL=(username1,username2)
NOPASSWD: /opt/globus/GT4.2.1/libexec/globus-gridmap-and-execute \
    -g /etc/grid-security/grid-mapfile \
    /opt/globus/GT4.2.1/libexec/globus-gram-local-proxy-tool *
```

The `globus-gridmap-and-execute` program is used to ensure that GRAM only runs programs under accounts that are in the `grid-mapfile`. In the sudo configuration, it is the first program called. It looks up the account in the `grid-mapfile` and then runs the requested command. It is redundant if sudo is properly locked down. This tool could be replaced with your own authorization program.

1.2. Configuring Scheduler Adapters

The GRAM4 scheduler adapters included in the release tarball are: [*PBS*](#), [*Condor*](#) and [*LSF*](#). To install, follow these steps (shown for *pbs*):

```
% configure --prefix=$GLOBUS_LOCATION --enable-wsgram-pbs ...
% make
% 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 last thing is to define the [GRAM and GridFTP file system mapping for PBS](#). A default mapping in this file is created to allow simple jobs to run. However, the actual file system mappings for your compute resource should be entered to ensure:

- files staging is performed correctly
- jobs with erroneous file path directives are rejected

Done! You have added the PBS scheduler adapters to your GT installation.

2. Non-default Configuration

2.1. Credentials

To run the container using just a user proxy, instead of host creds, edit the `$GLOBUS_LOCATION/etc/globus_wsrf_core/global_security_descriptor.xml` file, and either comment out the credentials section...

```
<?xml version="1.0" encoding="UTF-8"?>
<securityConfig xmlns="http://www.globus.org">
<!--
<credential>
<key-file value="/etc/grid-security/containerkey.pem"/>
<cert-file value="/etc/grid-security/containercert.pem"/>
<credential>
-->
<gridmap value="/etc/grid-security/grid-mapfile"/>
<securityConfig>
```

or replace the credentials section with a proxy file location...

```
<?xml version="1.0" encoding="UTF-8"?>
<securityConfig xmlns="http://www.globus.org">
<proxy-file value="<PATH TO PROXY FILE>" />
<gridmap value="/etc/grid-security/grid-mapfile"/>
<securityConfig>
```

Running in personal mode (user proxy), another GRAM configuration setting is required. For GRAM to authorize the RFT service when performing staging functions, it needs to know the subject DN for verification. Here are the steps:

```
% cd $GLOBUS_LOCATION/setup/globus
% ./setup-gram-service-common --staging-subject=
"/DC=org/DC=doegrids/OU=People/CN=Stuart Martin 564720"
```

You can get your subject DN by running this command:

```
% grid-cert-info -subject
```

2.2. GridFTP server

By default, the GridFTP server is assumed to run as root on localhost:2811. If this is not true for your site then change it by editing the GridFTP host and/or port in the [GRAM and GridFTP file system mapping](#) config file: `$GLOBUS_LOCATION/etc/globus_wsrf_gram/globus_gram_fs_map_config.xml`.

2.3. Container port

By default, the globus services will assume 8443 is the port the Globus container is using. However the container can be run under a non-standard port, for example:

```
% globus-start-container -p 4321
```

2.4. Gridmap

If you wish to specify a non-standard gridmap file in a multi-user installation, two basic configurations need to be changed:

- `$GLOBUS_LOCATION/etc/globus_wsrp_core/global_security_descriptor.xml`
 - As specified in the [gridmap config](#) instructions, add a `<gridmap value="..."/>` element to the file appropriately.
- `/etc/sudoers`
 - Change the file path after all `-g` options


```
-g /path/to/grid-mapfile
```

Example: *global_security_descriptor.xml*

```
...
<gridmap value="/opt/grid-mapfile"/>
...
```

sudoers

```
...
# Globus GRAM entries
globus ALL=(username1,username2)
NOPASSWD: /opt/globus/GT4.2.1/libexec/globus-gridmap-and-execute \
  -g /opt/grid-mapfile \
  /opt/globus/GT4.2.1/libexec/globus-job-manager-script.pl *
globus ALL=(username1,username2)
NOPASSWD: /opt/globus/GT4.2.1/libexec/globus-gridmap-and-execute \
  -g /opt/grid-mapfile \
  /opt/globus/GT4.2.1/libexec/globus-gram-local-proxy-tool *...
```

2.5. RFT deployment

RFT is used by GRAM to stage files in and out of the job execution environment. In the default configuration, RFT is hosted in the same container as GRAM and is assumed to have the same service path and standard service names. This need not be the case. For example, the most likely alternative scenario is that RFT would be hosted separately in a container on a different machine. In any case, both the RFT and the Delegation Service endpoints need to be adjustable to allow this flexibility. The following options can be passed to the *setup-gram-service-common* script to affect these settings:

```
--staging-protocol=<protocol> \
--staging-host=<host> \
--staging-port=<port> \
--staging-service-path=<RFT and Delegation factory service path> \
--staging-factory-name=<RFT factory service name> \
--staging-delegation-factory-name=<name of Delegation factory service used by RFT>
```

for example

```
% setup-gram-service-common \  
--staging-protocol=http \  
--staging-host=somemachine.fakedomain.net \  
--staging-port=8444 \  
--staging-service-path=/tomcat/services/ \  
--staging-factory-name=MyReliableFileTransferFactoryService \  
--staging-delegation-factory-name=MyDelegationFactoryServiceForRFT
```

will internally cause the GRAM service code to construct the following EPR addresses:

```
http://somemachine.fakedomain.net:8444/tomcat/services/MyReliableFileTransferFactoryService  
http://somemachine.fakedomain.net:8444/tomcat/services/MyDelegationFactoryServiceForRFT
```

2.6. Authorization

Note that, by default, two authorization checks are done during an invocation of WS-GRAM:

1. One authorization check is done by the container as configured by the `defaultAuthzParam` element in `$GLOBUS_LOCATION/etc/globus_wsrfg_core/global_security_descriptor.xml`
2. Another check is done by WS-GRAM when it calls the Perl modules which are used for job submission to the underlying local resource manager. This is configured by the `authzChain` element which is, by default, set to `gridmap` in `$GLOBUS_LOCATION/etc/globus_wsrfg_gram/managed-job-factory-security-config.xml` and `$GLOBUS_LOCATION/etc/globus_wsrfg_gram/managed-job-security-config.xml`. This check is done for additional security reasons to make sure that a potentially hacked globus user account still can only act on behalf of the users who are defined in a `grid-mapfile`.

The second `gridmap` check can be avoided by adding a system property to the environment variable `GLOBUS_OPTIONS` before starting the container:

```
export GLOBUS_OPTIONS="$GLOBUS_OPTIONS -Dorg.globus.exec.disablegge=true"
```

This however does not mean, that no authorization check at all is done. The container still checks if the client is authorized as defined in `$GLOBUS_LOCATION/etc/globus_wsrfg_core/global_security_descriptor.xml` but there's no further authorization check when calling the Perl modules. It's up to the GT4 container administrator to decide whether or not to have that additional authorization check. Note that a change in the `sudo` configuration is required in that case because `globus-gridmap-and-execute` will not be executed. `/opt/globus/GT4.2.1` should be replaced with the value of `$GLOBUS_LOCATION` for your installation:

```
# Globus GRAM entries  
globus ALL=(username1,username2)  
NOPASSWD: /opt/globus/GT4.2.1/libexec/globus-job-manager-script.pl *  
globus ALL=(username1,username2)  
NOPASSWD: /opt/globus/GT4.2.1/libexec/globus-gram-local-proxy-tool *
```

3. Configuration Details

3.1. JNDI configuration

The configuration of WSRF resources and application-level service configuration not related to service deployment is contained in [JNDI](#)¹ files. The JNDI-based GRAM configuration is of two kinds:

¹ <http://java.sun.com/products/jndi/>

3.1.1. Common job factory configuration

The file `$GLOBUS_LOCATION/etc/globus_wsrf_gram/jndi-config.xml` contains configuration information that is common to every local resource manager.

More precisely, the configuration data it contains pertains to the implementation of the GRAM WSRF resources (factory resources and job resources), as well as initial values of WSRF resource properties that are always published by any Managed Job Factory WSRF resource.

The data is categorized by service, because according to WSRF, in spite of the service/resource separation of concern, a given service will use only one XML Schema type of resource. In practice it is therefore clearer to categorize the configuration resource implementation by service, even if theoretically speaking a given resource implementation could be used by several services. For more information, refer to the [Java WS Core documentation](#).

Here is the decomposition, in JNDI objects, of the common configuration data, categorized by service. Each XYZHome object contains the same Globus Core-defined information for the implementation of the WSRF resource, such as the Java implementation class for the resource (`resourceClass` datum), the Java class for the resource key (`resourceKeyType` datum), etc.

- `ManagedExecutableJobService`
 - `ManagedExecutableJobHome`: configuration of the implementation of resources for the service.
- `ManagedMultiJobService`
 - `ManagedMultiJobHome`: configuration of the implementation of resources for the service
- `ManagedJobFactoryService`
 - `FactoryServiceConfiguration`: this encapsulates configuration information used by the factory service. Currently this identifies the service to associate to a newly created job resource in order to create an endpoint reference and return it.
 - `ManagedJobFactoryHome`: implementation of resources for the service `resourceClass`
 - `FactoryHomeConfiguration`: this contains GRAM application-level configuration data i.e. values for resource properties common to all factory resources. For instance, the path to the Globus installation, host information such as CPU type, manufacturer, operating system name and version, etc.

3.1.2. Local resource manager configuration

When a SOAP call is made to a GRAM factory service in order to submit a job, the call is actually made to a GRAM service-resource pair, where the factory resource represents the local resource manager to be used to execute the job.

There is one directory `globus_wsrf_gram_<manager>/` for each local resource manager supported by the GRAM installation.

For instance, let's assume the command line:

```
% ls etc | grep globus_wsrf_gram_
```

gives the following output:

```
globus_wsrf_gram_Fork
globus_wsrf_gram_LSF
globus_wsrf_gram_Multi
```

In this example, the Multi, Fork and *LSF* job factory resources have been installed. *Multi* is a special kind of local resource manager which enables the GRAM services to support multijobs.

The JNDI configuration file located under each manager directory contains configuration information for the GRAM support of the given local resource manager, such as the name that GRAM uses to designate the given resource manager. This is referred to as the *GRAM name* of the local resource manager.

For instance, `$GLOBUS_LOCATION/etc/globus_wsrfg_gram_Fork/jndi-config.xml` contains the following XML element structure:

```
<service name="ManagedJobFactoryService">
  <!-- LRM configuration: Fork -->
  <resource
    name="ForkResourceConfiguration"
    type="org.globus.exec.service.factory.FactoryResourceConfiguration">
    <resourceParams>
      [...]
      <parameter>
        <name>
          localResourceManagerName
        </name>
        <value>
          Fork
        </value>
      </parameter>
      <!-- Site-specific scratchDir
        Default: ${GLOBUS_USER_HOME}/.globus/scratch
      <parameter>
        <name>
          scratchDirectory
        </name>
        <value>
          ${GLOBUS_USER_HOME}.globus/scratch
        </value>
      </parameter>
      -->
    </resourceParams>
  </resource>
</service>
```

In the example above, the name of the local resource manager is *Fork*. This value can be used with the GRAM command line client in order to specify which factory resource to use when submitting a job. Similarly, it is used to create an endpoint reference to the chosen factory WS-Resource when using the GRAM client API.

In the example above, the *scratchDirectory* is set to `${GLOBUS_USER_HOME}/.globus/scratch`. This is the default setting. It can be configured to point to an alternate file system path that is common to the compute cluster and is typically less reliable (auto purging), while offering a greater amount of disk space (thus "scratch").

3.2. Security descriptor

The file `$GLOBUS_LOCATION/etc/globus_wsrfg_gram/managed-job-factory-security-config.xml` contains the Core security configuration for the GRAM *ManagedJobFactory* service:

- default security information for all remote invocations, such as:

- the authorization method, based on a Gridmap file (in order to resolve user credentials to local user names)
- limited proxy credentials will be rejected
- security information for the `createManagedJob` operation

The file `$GLOBUS_LOCATION/etc/globus_wsrfg_gram/managed-job-security-config.xml` contains the Core security configuration for the GRAM job resources:

- The default is to only allow the identity that called the `createManagedJob` operation to access the resource.

Note: GRAM does not override the container security credentials defined in `$GLOBUS_LOCATION/etc/globus_wsrfg_core/global_security_descriptor.xml`. These are the credentials used to authenticate all service requests.

3.3. Web service deployment descriptor

The file `$GLOBUS_LOCATION/etc/globus_wsrfg_gram/server-config.wsdd` contains information necessary to deploy and instantiate the GRAM services in the Globus container.

Three GRAM services are deployed:

- `ManagedExecutableJobService`: service invoked when querying or managing an *executable job*
- `ManagedMultiJobService`: service invoked when querying or managing a *multijob*
- `ManagedJobFactoryService`: service invoked when submitting a job

Each service deployment information contains the name of the Java service implementation class, the path to the WSDL service file, the name of the operation providers that the service reuses for its implementation of WSDL-defined operations, etc. More information about the service deployment configuration information can be found in [Configuring Java WS Core](#).

3.4. Scheduler-Specific Scheduler Event Generator configuration files

In addition to the service configuration described above, there are scheduler-specific configuration files for the Scheduler Event Generator modules. These files consist of name=value pairs separated by newlines. These files are:

3.4.1. `$GLOBUS_LOCATION/etc/globus-fork.conf`

Configuration for the Fork *SEG* module implementation. The attributes names for this file are:

`log_path` Path to the SEG Fork log (used by the `globus-fork-starter` and the *SEG*). The value of this should be the path to a world-writable file. The default value for this created by the Fork setup package is `$GLOBUS_LOCATION/var/globus-fork.log`. This file must be readable by the account that the *SEG* is running as.

3.4.2. `$GLOBUS_LOCATION/etc/globus-condor.conf`

Configuration for the *Condor* *SEG* module implementation. The attributes names for this file are:

`log_path` Path to the SEG Condor log (used by the `Globus::GRAM::JobManager::condor` perl module and Condor SEG module. The value of this should be the path to a world-readable and world-writable file. The default value for this created by the Fork setup package is `$GLOBUS_LOCATION/var/globus-condor.log`

3.4.3. `$GLOBUS_LOCATION/etc/globus-pbs.conf`

Configuration for the *PBS* SEG module implementation. The attributes names for this file are:

`log_path` Path to the SEG PBS logs (used by the `Globus::GRAM::JobManager::pbs` perl module and PBS SEG module. The value of this should be the path to the directory containing the server logs generated by PBS. For the SEG to operate, these files must have file permissions such that the files may be read by the user the SEG is running as.

3.4.4. `$GLOBUS_LOCATION/etc/globus-lsf.conf`

Configuration for the *LSF* SEG module implementation. The attributes names for this file are:

`log_path` Path to the SEG LSF log directory. This is used by the LSF SEG module. The value of this should be the path to the directory containing the server logs generated by LSF. For the SEG to operate, these files must have file permissions such that the files may be read by the user the SEG is running as.

4. GRAM4 and GridFTP file system mapping

The file `$GLOBUS_LOCATION/etc/globus_wsrfg_gram/globus_gram_fs_map_config.xml` contains information to associate local resource managers with GridFTP servers. GRAM uses the GridFTP server (via RFT) to perform all file staging directives. Since the GridFTP server and the Globus service container can be run on separate hosts, a mapping is needed between the common file system paths of these 2 hosts. This enables the GRAM services to resolve `file:///` staging directives to the local GridFTP URLs.

Below is the default Fork entry. Mapping a `jobPath` of `/` to `ftpPath` of `/` will allow any file staging directive to be attempted.

```
<map>
  <scheduler>Fork</scheduler>
  <ftpServer>
    <protocol>gsiftp</protocol>
    <host>myhost.org</host>
    <port>2811</port>
  </ftpServer>
  <mapping>
    <jobPath>/</jobPath>
    <ftpPath>/</ftpPath>
  </mapping>
</map>
```

For a *scheduler*, where jobs will typically run on a compute node, a default entry is not provided. This means staging directives will fail until a mapping is entered. Here is an example of a compute cluster with *PBS* installed that has 2 common mount points between the front end host and the GridFTP server host.

```
<map>
  <scheduler>PBS</scheduler>
  <ftpServer>
    <protocol>gsiftp</protocol>
    <host>myhost.org</host>
    <port>2811</port>
```

```

</ftpServer>
<mapping>
  <jobPath>/pvfs/mount1/users</jobPath>
  <ftpPath>/pvfs/mount2/users</ftpPath>
</mapping>
<mapping>
  <jobPath>/pvfs/jobhome</jobPath>
  <ftpPath>/pvfs/ftphome</ftpPath>
</mapping>
</map>

```

The file system mapping schema doc is [here](#)².

5. GRAM4 - RFT interaction

If a job description contains a fileStageIn, fileStageOut or a fileCleanUp element, or if globusrun-ws submits a job in streaming mode, GRAM4 uses RFT to perform transfer or delete requests. There are two ways GRAM4 can call RFT:

1. WS calls
2. Local Java calls, if RFT is located in the same container like GRAM4

By default GRAM4 in GT 4.2 is configured to use RFT located in the same GT4 container and to do local Java calls to call it. This improves performance in jobs with staging significantly. No additional security check is done in the call to RFT in that case; RFT trusts that a client had already been authenticated and authorized properly by GRAM4. We don't consider this to be a security risk because the client already passed security in the call to GRAM4, and because another security check is performed when RFT calls the GridFTP server(s) on behalf of the client using the delegated credentials.

If an admin wants to have different authorization for GRAM4 and RFT, or wants to configure GRAM4 to use RFT in a different container, or just does not want local Java calls, then they must be disabled by setting the parameter enableLocalInvocations to false in `/${GLOBUS_LOCATION}/etc/globus_wsrft_gram/jndi-config.xml`.

If `/${GLOBUS_LOCATION}/setup/globus/setup-gram-service-common` is used to configure GRAM4 to use RFT in a different container, then local Java calls will be disabled automatically.

6. Defining a default local resource manager

A client can submit a job without specifying the local resource manager (LRM) that should execute the job. By this it does not need to know which LRM is used by GRAM4: Condor, LSF, PBS, ...

To enable this there is a configuration parameter in the JNDI configuration that defines the default LRM if the client didn't specify it itself. By default it's set to Fork, but it can be changed by modifying the parameter defaultLocalResourceManager in `/${GLOBUS_LOCATION}/etc/globus_wsrft_gram/jndi-config.xml`. The following example illustrates PBS as default local resource manager:

```

<resource name="homeConfiguration"
  type="org.globus.exec.service.factory.FactoryHomeConfiguration">
  <resourceParams>
    ....
    <parameter>

```

² [../schemas/gram_fs_map.html](#)

```

    <name>
      defaultLocalResourceManager
    </name>
    <value>
      PBS
    </value>
  </parameter>
  ....
</resourceParams>
</resource>

```

A client can specify a local resource manager though. In that case the explicitly specified LRM is used in job submission.

7. Disabling an already installed local resource manager adapter

When GRAM4 is initialized during startup of the GT container the JNDI configuration is checked for configured local resource manager adapters. If you want to disable an already installed local resource manager adapter you have to make sure that it's removed from the JNDI configuration. The following explains a way how this could be done:

Say, you installed support for PBS and want to disable PBS now. A listing of the GRAM4 related directories in `$GLOBUS_LOCATION/etc` will look like this:

```

[martin@osg-test1 ~]$ cd $GLOBUS_LOCATION/etc && ls | grep globus_wsrp_gram
globus_wsrp_gram
globus_wsrp_gram_Fork
globus_wsrp_gram_Multi
globus_wsrp_gram_PBS

```

All you have to do is to remove `globus_wsrp_gram_PBS`, or better, create an archive before removing it in case you want to enable PBS support at a later time again. After doing that the output of the above command should look like this:

```

[martin@osg-test1 ~]$ cd $GLOBUS_LOCATION/etc && ls | grep globus_wsrp_gram
globus_wsrp_gram
globus_wsrp_gram_Fork
globus_wsrp_gram_Multi
globus_wsrp_gram_PBS.tar.gz

```

After restarting the GT server users won't be able to submit jobs to PBS anymore.

8. Job lifetime configuration

For a general introduction see section [Job Lifetime](#) in the GRAM4 approach.

There are 2 parameters in GRAM4's JNDI configuration in `$GLOBUS_LOCATION/etc/globus_wsrp_gram/jndi-config.xml` that have impact on lifetime of job resources:

maxJobLifetime	Max lifetime a client can specify in the initial job submission and in subsequent <code>setTerminationTime</code> calls. Default value is 1 year. A negative value means that there is no limit.
-----------------------	--

jobTTLOnProcessing	Amount of time a job resource keeps on existing after the job has been fully processed and is in a final state Done, Failed, UserTerminateDone, UserTerminateFailed, and the client did not specify a job lifetime. Default value is 24h. A negative value means that the job resource does not expire.
---------------------------	---

Values are specified in seconds.

The parameters are exposed as resource properties of the factory resources to enable a client to query the values of the configuration parameters.

If a client does not specify any lifetime at all the job will run to completion in any event. After processing the lifetime will be set to (now + jobTTLOnProcessing)

9. Registration with default WS MDS Index Service

9.1. Auto-registration

With a default GT 4.2.1 installation, the GRAM4 service is automatically registered with the default [WS MDS Index Service](#) running in the same container for monitoring and discovery purposes.

This is how auto-registration is configured:

There is a jndi resource defined in `$GLOBUS_LOCATION/etc/globus_wsrf_gram/jndi-config.xml` as follows :

```
<resource name="mdsConfiguration"
  type="org.globus.wsrfl.impl.servicegroup.client.MDSConfiguration">
  <resourceParams>
    <parameter>
      <name>reg</name>
      <value>true</value>
    </parameter>
    <parameter>
      <name>factory</name>
      <value>org.globus.wsrfl.jndi.BeanFactory</value>
    </parameter>
  </resourceParams>
</resource>
```

To configure the automatic registration of GRAM4 to the default WS MDS Index Service, change the value of the parameter `<reg>` as follows:

- `true` turns on auto-registration; this is the default in GT 4.2.1.
- `false` turns off auto-registration.

9.1.1. Configuring resource properties

By default, the `GLUECE :` resource property (which contains GLUE data) is sent to the default Index Service:

You can configure which resource properties are sent in GRAM4's registration.xml file, \$GLOBUS_LOCATION/etc/globus_wsrp_gram/registration.xml. The following is the relevant section of the file (as it is set by default):

```
<Content xsi:type="agg:AggregatorContent"
  xmlns:agg="http://mds.globus.org/aggregator/types">
  <agg:AggregatorConfig xsi:type="agg:AggregatorConfig">
    <agg:GetResourcePropertyPollType
      xmlns:glue="http://mds.globus.org/glue/ce/1.1">
      <!-- Specifies that the index should refresh information
        every 60000 milliseconds (once per minute) -->
      <agg:PollIntervalMillis>60000</agg:PollIntervalMillis>
      <!-- specifies the resource property that should be
        aggregated, which in this case is the GLUE cluster
        and scheduler information RP -->
      <agg:ResourcePropertyName>glue:GLUECE</agg:ResourcePropertyName>
    </agg:GetResourcePropertyPollType>
  </agg:AggregatorConfig>
  <agg:AggregatorData/>
</Content>
```

9.2. Manual registration

If a third party needs to register an GRAM4 service manually, see [Registering with mds-servicegroup-add](#) in the WS MDS Aggregator Framework documentation.

10. Configuring thread-pools

GRAM4 has two thread-pools that regulate internal processing.

One pool is responsible for processing all jobs in all states. The size of this pool limits the load GRAM4 produces in the GT container, and can be configured by modifying the parameter `runQueueThreadCount` in \$GLOBUS_LOCATION/etc/globus_wsrp_gram/jndi-config.xml. The default value is 10.

The second pool defines how many threads process job events dispatched to GRAM4 by the *Scheduler Event Generator* (SEG) of each configured local resource manager. The default number of threads is 1, i.e. there's one thread that forwards events of job state changes for all jobs of all configured local resource managers. This should be fine for almost all scenarios. If a SEG however provides a lot of events, maybe even periodical events, or if the container persistence directory is located on a slow NFS disk, setting the parameter `segEventProcessingThreadCount` in \$GLOBUS_LOCATION/etc/globus_wsrp_gram/jndi-config.xml to a higher value might make sense.

Chapter 3. Deploying

GRAM4 is deployed as part of a standard toolkit installation. Please refer to the [Installing GT 4.2.1](#) for details.

1. Deploying in Tomcat

GRAM4 has been tested to work without any additional setup steps when deployed into Tomcat. Please see [Deploying into Tomcat](#) for instructions. Also, for details on tested containers, see the [Section 7, “Tested platforms”](#).



Note

Currently only a single deployment is supported because of a limitation in the execution of the Scheduler Event Generator. One must set GLOBUS_LOCATION before starting Tomcat.

Chapter 4. Scalability and Performance Recommendations

This document includes recommendations for increasing the scalability and performance of GRAM4 in a Grid.

1. Server-side Recommendations

1. GRAM4 service and/or the container can run out of memory under lower settings. For this reason, set the Max container heap size to be 1GB.

```
GLOBUS_OPTIONS="-Xms256M -Xmx1024M"
```

2. The account the container runs under, typically "globus", can run out of open file descriptors. For this reason, set the open file descriptors to 16,384.

Specific settings can vary per operating system; for a "globus" user on redhat / RHEL based distributions, add the following to `/etc/security/limits.conf`:

```
globus          hard    nofile          16384
```

3. The GRAM4 service stores the per job metadata used for crash/ recovery in files on disk. By default, the container account's home dir is used, specifically `~/ .globus/persisted/`. Often this home dir is not located on a local disk, but on NFS. NFS is not needed for this purpose and can negatively effect performance. For this reason, configure the container to use a local disk.

```
GLOBUS_OPTIONS="-Dorg.globus.wsrfl.container.persistence.dir=/use/this/path"
```

Make sure you don't overwrite the above memory settings. You could provide both settings in the same GLOBUS_OPTIONS variable like:

```
GLOBUS_OPTIONS="-Xms256M -Xmx512M -Dorg.globus.wsrfl.container.persistence.dir=/use/this/path"
```

4. We recommend the following thread settings as part of the global configuration in `$GLOBUS_LOCA-TION/etc/globus_wsrfl_core/server-config.wsdd`:

```
<globalConfiguration:
  ...
  <parameter name="containerThreads" value="20"/>
  <parameter name="containerThreadsMax" value="40"/>
  <parameter name="containerThreadsHighWaterMark" value="10"/>
  ...
</globalConfiguration>
```

For more information, see global configurations under [Java WS Core](#).

Chapter 5. Audit Logging

1. Overview

GRAM4 includes mechanisms to provide access to audit and accounting information associated with jobs that GRAM4 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.

Audit logging in GRAM4 is done 3 times in a job's lifecycle:

1. when the processing starts,
2. when the job is submitted to the local resource manager (LRM), and
3. when it is fully processed or when it fails.

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:

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

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. Converting EPR to GRAM Service GJID

The GRAM4 service returns an EPR that is used to control the job. However, the EPR is an XML document and cannot effectively be used as a primary key for a database table. Therefore, the job's EPR needs to be converted to an acceptable GJID format.

A utility class, `org.globus.exec.utils.audit.AuditUtil`, is available both the GRAM service before storing the audit record and the GRAM client before getting audit information from the audit database.

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,

- 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 GRAM4 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. Configuration To Enable Audit Logging

Audit logging is turned off by default. It is configured entirely in Gram4's JNDI configuration in `/${GLOBUS_LOCATION}/etc/globus_wsrf_gram/jndi-config.xml`, using 2 sections: a general configuration section and a database section:

6.1. General configuration

```
<resource name="auditConfiguration" type="org.globus.exec.service.exec.utils.audit.AuditCo
  <resourceParams>
    . . . .
    <parameter>
      <name>enableAuditLogging</name>
      <value>>false</value>
    </parameter>
    <parameter>
      <name>auditVersion</name>
      <value>1</value>
    </parameter>
    <parameter>
      <name>fallbackStorageDirectory</name>
      <value>/opt/gt421/share/globus_wsrf_gram/</value>
    </parameter>
    <parameter>
      <name>dbUploadRetryInterval</name>
      <value>300</value>
    </parameter>
  </resourceParams>
</resource>
```

Parameter	Explanation
enableAuditLogging	true to turn audit logging on, false to turn it off
auditVersion	Currently only version 1 is supported.
fallbackStorageDirectory	If the insert or the update of a record into the database system fails because the database is down or misconfigured, the record is stored as a file in the dir-

¹ <http://www.globus.org/toolkit/docs/4.0/techpreview/ogsadai/>

² http://www.teragridforum.org/mediawiki/index.php?title=GRAM4_Audit

Parameter	Explanation
	ectory specified by this parameter. This ensures that no records are lost.Periodical attempts to upload the records being stored in this directory into the database are performed by Gram4. Once the upload of a fallback record was successful the record file will be deleted.
dbUploadRetryInterval	Time in seconds after which, periodically, an attempt is made to upload fallback records into the database.

6.2. Database configuration

```
<resource name="auditDatabase" type="javax.sql.DataSource">
  <resourceParams>
    ....
    <parameter>
      <name>driverClassName</name>
      <value>driver class name</value>
    </parameter>
    <parameter>
      <name>url</name>
      <value>db connection url</value>
    </parameter>
    <parameter>
      <name>username</name>
      <value>user to access the database</value>
    </parameter>
    <parameter>
      <name>password</name>
      <value>password to access the database</value>
    </parameter>
  </resourceParams>
</resource>
```

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.

DB system	driverClassName	url
MySQL	com.mysql.jdbc.Driver	jdbc:mysql://HOST[:PORT]/auditDatabase
PostgreSQL	org.postgresql.Driver	jdbc:mysql://HOST[:PORT]/auditDatabase
Derby	org.apache.derby.jdbc.Embedded-Driver	jdbc:derby:directory:PATH_TO_GLOBUS_LOCATION/var/gram/auditDatabase

6.3. Creating the Audit Database

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

6.3.1. 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 < ${GLOBUS_LOCATION}/share/globus_wsrfg
Enter password:
host:~ feller$

```

6.3.2. 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\g
create user gt4auditload with encrypted password '<password1>'\g
create user gt4auditview with encrypted password '<password2>'\g
\c gt4audit
\i gram_audit_schema_postgres-8.0.sql
grant insert on gram_audit_table to gt4auditload\g
grant select on gram_audit_table to gt4auditview\g
\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

```

6.3.3. Derby

During GT installation the Derby audit database is already created. Its 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_ws-rf_gram/gram_audit_v1_schema_derby.sql`.

Chapter 6. Job Description Extensions Support

The GRAM4 job description schema includes a section for extending the job description with custom elements. To make sense of this in the resource manager adapter Perl scripts, a Perl module named `Globus::GRAM::ExtensionsHandler` is provided to turn these custom elements into parameters that the adapter scripts can understand.

Note

Although only non-GRAM XML elements are allowed in the `<extensions>` element of the job description, the extensions handler makes no distinction based on namespace. Thus, `<foo:myparam>` and `<bar:myparam>` will both be treated as just `<myparam>`.

Familiarity with the adapter scripts is assumed in the following sub-sections.

1. Requirements for Extensions Support

- `XML::Parser` Perl module

2. Supported Extension Constructs

2.1. Simple String Parameters

Simple string extension elements are converted into single-element arrays with the name of the unqualified tag name of the extension element as the array's key name in the Perl job description hash. Simple string extension elements can be considered a special case of the string array construct in the next section.

For example, adding the following element to the `<extensions>` element of the job description as follows:

```
<extensions>
  <myparam>yahoo!</myparam>
</extensions>
```

will cause the `$description->myparam()` to return the following value:

```
'yahoo!'
```

2.2. String Array Parameters

String arrays are a simple iteration of the simple string element construct. If you specify more than one simple string element in the job description, these will be assembled into a multi-element array with the unqualified tag name of the extension elements as the array's key name in the Perl job description hash.

For example:

```
<extensions>
```

```
<myparams>Hello</myparams>
<myparams>World!</myparams>
</extensions>
```

will cause the `$description->myparams()` to return the following value:

```
[ 'Hello', 'World!' ]
```

2.3. Name/Value Parameters

Name/value extension elements can be thought of as string arrays with an XML attribute 'name'. This will cause the creation of a two-dimensional array with the unqualified extension element tag name as the name of the array in the Perl job description hash.

For example:

```
<extensions>
  <myvars name="pi">3.14159</myvars>
  <myvars name="mole">6.022 x 10^23</myvars>
</extensions>
```

will cause the `$description->myvars()` to return the following value:

```
[ [ 'pi', '3.14159' ], [ 'mole', '6.022 x 10^23' ] ]
```

2.4. PBS Node Selection Parameters

Node selection constraints in PBS can be specified in one of the following ways:

- generally, using a construct intended to eventually apply to all resource managers which support node selection
- explicitly, by specifying a simple string element.

The former will be more portable, but the latter will appeal to those familiar with specifying node constraints for PBS jobs.

To specify PBS node selection constraints explicitly, one can simply construct a single, simple string extension element named `nodes` with a value that conforms to the `#PBS -l nodes=...` PBS job description directive. The `Global::GRAM::ExtensionsHandler` module will make this available to the PBS adapter script by invoking `$description->{nodes}`. The updated PBS adapter package checks for this value and will create a directive in the PBS job description using this value.

To use the generic construct for specifying node selection constraints, use the `resourceAllocationGroup` element:

```
<extensions>
<resourceAllocationGroup>
<!-- Optionally select hosts by type and number... -->
<hostType>...</hostType>
<hostCount>...</hostCount>

<!-- *OR* by host names -->
```

```

<hostName>...</hostName>
<hostName>...</hostName>
. . .

<!-- With a total CPU count for this group... -->
<cpuCount>...</cpuCount>

<!-- *OR* an explicit number of CPUs per node... -->
<cpusPerHost>...</cpusPerHost>
. . .

<!-- And a total process count for this group... -->
<processCount>...</processCount>

<!-- *OR* an explicit number of processes per node... -->
<processesPerHost>...</processesPerHost>
</resourceAllocationGroup>
</extensions>

```

Extension elements specified according to the above pseudo-schema will be converted to an appropriate nodes parameter which will be treated as if an explicit nodes extension element were specified.

Multiple resourceAllocationGroup elements may be specified. This will simply append the constraints to the nodes parameter with a '+' separator.



Note

You cannot specify both hostType/hostCount and hostName elements. Similarly, one cannot specify both processCount and processesPerHost elements.

Here are some examples of using resourceAllocationGroup:

```

<!-- #PBS -l nodes=1:ppn=10 -->
<!-- 10 processes -->
<extensions>
<resourceAllocationGroup>
<cpuCount>10</cpuCount>
<processCount>10</processCount>
</resourceAllocationGroup>
</extensions>

<!-- #PBS -l nodes=activemural:ppn=10+5:ia64-compute:ppn=2 -->
<!-- 1 process (process default) -->
<extensions>
<resourceAllocationGroup>
<hostType>activemural</hostType>
<cpuCount>10</cpuCount>
</resourceAllocationGroup>
<resourceAllocationGroup>
<hostType>ia64-compute</hostType>

```

```
<hostCount>5</hostCount>
<cpusPerHost>2</cpusPerHost>
</resourceAllocationGroup>
</extensions>

<!-- #PBS -l nodes=vis001:ppn=5+vis002:ppn=5+comp014:ppn=2+comp015:ppn=2 -->
<!-- 15 total processes -->
<extensions>
<resourceAllocationGroup>
<hostName>vis001</hostName>
<hostName>vis002</hostName>
<cpuCount>10</cpuCount>
<processesPerHost>5</processesPerHost>
</resourceAllocationGroup>
<resourceAllocationGroup>
<hostName>comp014</hostName>
<hostName>comp015</hostName>
<cpusPerHost>2</cpusPerHost>
<processCount>5</processCount>
</resourceAllocationGroup>
</extensions>
```

3. Customizing Extensions Support

Two Perl modules must be edited to customize extensions support.

- The first is `ExtensionsHandler.pm`. This is where the GRAM4 job description XML of the `extensions` element is parsed and entries are added or appended to the Perl job description hash.
- The second module that needs to be edited is the particular resource manager adapter module that will use any new hash entries to either alter its behavior or create additional parameters in the resource manager job description.

3.1. Customizing ExtensionsHandler.pm

This module logs various things to the log file specified in the `logfile` extension element. If you place this element at the start of the extensions for which you are creating support, then you can look at the specified log file to get some idea of what the handler is doing. You can add new logging lines by using the `$self->log()` function. This simply takes a string that gets appended to the log file with a prefix of "`<date string> EXTENSIONS HANDLER:`".

There are three main subroutines that are used to handle parsing events and process them accordingly:

- `Char()`
- `StartTag()`
- `EndTag()`

More handlers can be specified for other specific events when creating the `XML::Parser` instance in `new()` (see the [XML::Parser](#)¹ documentation for details).

The following list describes what the three main subroutines currently do. Modify the subroutines as necessary to achieve your specific goal.

¹ <http://search.cpan.org/~coopercl/XML-Parser-2.31/Parser.pm>

<code>Char()</code>	Doesn't do anything but collect CDATA found between the current element's start and end tags. You can access the CDATA for the current element by using <code>\$self->{CDATA}</code> .
<code>StartTag()</code>	Responsible for collecting the attributes associated with the element. It also increments the counter, which keeps track of the number of child elements to the current extension element, and pushes the current element name onto the <code>@scope</code> queue for later use.
<code>EndTag()</code>	Takes the CDATA collected by <code>Char()</code> and creates new Perl job description hash entries. This is most likely where you will need to do most of your work when adding support for new extension elements. Two useful variables are <code>\$currentScope</code> and <code>\$parentScope</code> . These indicate the current element that is being parsed and the parent of the element being parsed respectively. This is useful for establishing a context from which to work. The <code>@scope</code> queue is piped at the end of this subroutine.

3.2. Customizing the Adapter Module

Each adapter and each extension's purpose is different, so there aren't any specific instructions for modifying the resource manager/scheduler adapter module. It is suggested that you spend some time trying to understand what the adapter does and how before making your changes.

Any new hash entries you created in `ExtensionsHandler.pm` (see the "Customizing ExtensionsHandler.pm" section above) can be accessed by calling `$description->entryname()` from the adapter module, where 'entryname' is the name of the entry that was added.

See the [construct documentation](#) above for more details on generic constructs that are already supported in `Extension-sHandler.pm`. This is often an easier route to implementing your extensions than creating a custom construct.

Chapter 7. SoftEnv Support

1. Overview

SoftEnv is a system designed to make it easier for users to define what applications they want to use, and easier for administrators to make applications available to users. SoftEnv has evolved from the original implementation called Soft designed at Northeastern University in 1994.

In some environments, like TeraGrid, it is desirable to make use of SoftEnv before a job is submitted to leverage the use of an exactly defined software environment in which the job will run.

2. Configuring SoftEnv Support

Because this feature is very specific and may not be available on many systems, support for SoftEnv is disabled by default in normal job submissions. There is a parameter in the JNDI configuration of GRAM4 to enable SoftEnv support in job submissions.

SoftEnv support must be enabled on a per-scheduler basis because the internal mechanisms to support SoftEnv vary between the different types of schedulers. Currently only the Fork, PBS and LSF schedulers can be configured to have SoftEnv support enabled (Condor is not yet supported).

To enable this feature, set the parameter *enableDefaultSoftwareEnvironment* in the scheduler specific JNDI configuration to `true`.

For example, to enable SoftEnv support in the Fork scheduler, set the *enableDefaultSoftwareEnvironment* in `$GLOBUS_LOCATION/etc/globus_wsrf_gram_Fork/jndi-config.xml` to `true`.

Enabled SoftEnv support means that a user's default environment will be created from his `.soft` file before each job submission automatically. The user does not need to provide extra SoftEnv keys in the `extensions` element of a job description. This is not done if the SoftEnv feature is disabled.

For more information and examples, please look in the [SoftEnv section of the User's Guide](#).

3. Dependencies

For the scheduler, Fork SoftEnv needs to be installed on the host in which the container is running.

For PBS and LSF, SoftEnv needs to be installed on the hosts where the jobs are executed.

Chapter 8. Testing

See the user guide for information about submitting a test job.

Chapter 9. Security Considerations

No special security considerations exist at this time.

Chapter 10. Admin Debugging

Because GRAM4 is built on Java WS Core, it uses the same sys admin logging, described below:

1. Logging in Java WS Core

The following information applies to Java WS Core and all services built on Java WS Core.

Java WS Core server side has two types of loggers. One logger is used for development logging and by default writes to standard out. The other logger includes system administration information and is [CEDPs best practices](#)¹ compliant.

On client side, only developer logging is available and is configured using `log4j.properties`.

1.1. Development Logging in Java WS Core

The following information applies to Java WS Core and those services built on it.

Logging in the Java WS Core is based on the [Jakarta Commons Logging](#)² API. Commons Logging provides a consistent interface for instrumenting source code while at the same time allowing the user to plug-in a different logging implementation. Currently we use [Log4j](#)³ as a logging implementation. Log4j uses a separate configuration file to configure itself. Please see Log4j documentation for details on the [configuration file format](#)⁴.

1.1.1. Configuring server side developer logs

Server side logging can be configured in `$GLOBUS_LOCATION/container-log4j.properties`, when the container is stand alone container. For tomcat level logging, refer to [Logging for Tomcat](#)⁵. The logger `log4j.appender.A1` is used for developer logging and by default writes output to the system output. By default it is set for all warnings in the Globus Toolkit package to be displayed.

Additional logging can be enabled for a package by adding a new line to the configuration file. Example:

```
#for debug level logging from org.globus.package.FooClass
log4j.category.org.globus.package.name.FooClass=DEBUG
#for warnings from org.some.warn.package
log4j.category.org.some.warn.package=WARN
```

1.1.2. Configuring client side developer logs

Client side logging can be configured in `$GLOBUS_LOCATION/log4j.properties`. The logger `log4j.appender.A1` is used for developer logging and by default writes output to the system output. By default it is set for all warnings in the Globus Toolkit package to be displayed.

¹ <http://cedps.net/index.php/LoggingBestPractices>

² <http://jakarta.apache.org/commons/logging/>

³ <http://logging.apache.org/log4j/>

⁴ [http://logging.apache.org/log4j/docs/api/org/apache/log4j/PropertyConfigurator.html#doConfigure\(java.lang.String,org.apache.log4j.spi.LoggerRepository\)](http://logging.apache.org/log4j/docs/api/org/apache/log4j/PropertyConfigurator.html#doConfigure(java.lang.String,org.apache.log4j.spi.LoggerRepository))

⁵ <http://tomcat.apache.org/tomcat-5.5-doc/logging.html>

1.2. Configuring system administration logs

The specific logger to edit will be `log4j.logger.sysadmin` in `$GLOBUS_LOCATION/container-log4j.properties`. There you can configure the following properties:

```
log4j.appender.infoCategory=org.apache.log4j.RollingFileAppender
log4j.appender.infoCategory.Threshold=INFO
log4j.appender.infoCategory.File=var/containerLog
log4j.appender.infoCategory.MaxFileSize=10MB
log4j.appender.infoCategory.MaxBackupIndex=2
```

Above implies the logging file is rolling with each file size limited to 10MB and the logging information is stored in `$GLOBUS_LOCATION/var/containerLog`.

1.3. Sample log file

The [sample log file](#)⁶ contains many log entries for various scenarios in the Java WS container.

⁶ <http://www.globus.org/toolkit/docs/4.2/4.2.1/common/javawscore/sample-container-log.txt>

Chapter 11. Troubleshooting

For a list of common errors in GT, see [Error Codes](#).

For information about sys admin logging, see [Chapter 10, Admin Debugging](#) in the GRAM4 Admin Guide.

1. Troubleshooting tips

In case you run into problems you can do the following

- Check the GRAM4 documentation. Maybe you'll find hints here to solve your problem.
- Send e-mails to one of several Globus e-mail lists. You'll have to subscribe to a list before you can send an e-mail to it. See [here](#)¹ for general e-mail lists and information on how to subscribe to a list and [here](#)² for GRAM specific lists.

Probably the best lists for GRAM4-related problems are gt-user@globus.org and gram-user@globus.org

- Check the container log for errors.

In case you don't find anything suspicious you can increase the log-level of GRAM4 or other relevant components. Maybe the additional logging-information will tell you what's going wrong. General information about container logging can be found [Logging in Java WS Core](#) section.

To get debug information from GRAM4, un-comment the following line in `$GLOBUS_LOCATION/container-log4j.properties` by removing the leading '#' and restart the GT4 server.

```
# log4j.category.org.globus.exec=DEBUG
```

The logging output can either be found on the console if you started the container using `globus-start-container` (maybe with arguments) or in `$GLOBUS_LOCATION/var/container.log` in if you started the container using the command `globus-start-container-detached`

¹ http://dev.globus.org/wiki/Mailing_Lists

² http://dev.globus.org/wiki/GRAM#Mailing_Lists

2. Java WS Core Errors

Table 11.1. Java WS Core Errors

Error Code	Definition
Failed to acquire notification consumer home instance from registry	Caused by <code>javax.naming.NameNotFoundException</code> : Name <code>services</code> is not bound in
The WS-Addressing 'To' request header is missing	This warning is logged by the container if the request did not contain the necessary <i>WS-Addressing</i> headers, those headers at all or is somehow misconfigured.
<code>java.io.IOException: Token length X > 33554432</code>	If you see this error in the container log, it usually means you are trying to connect to HTTPS server using <code>https</code> specifies 8443 as a port number and <code>http</code> as the protocol name.
<code>java.lang.NoSuchFieldError: DOCUMENT</code>	This error usually indicates a mismatch between the version of Apache Axis that the code was compiled with and the version currently running with.
<code>org.globus.wsrfl.InvalidResourceKeyException: Argument key is null / Resource key is missing</code>	These errors usually indicate that a resource key was not passed with the request or that an invalid resource key was used (the element <code>QName</code> of the resource key did not match what the service expected).
Unable to connect to localhost:xxx	Cannot resolve localhost. The machine's <code>/etc/hosts</code> isn't set up correctly and/or you do not have DNS for
<code>org.globus.common.ChainedIOException: Failed to initialize security context</code>	This may indicate that the user's proxy is invalid.
Error: <code>org.xml.sax.SAXException: Unregistered type: class xxx</code>	This may indicate that an Axis generated XML type, defined by the WS RLS XSD, was not properly registered upon deployment without intervention by the user, sometimes they do not.
No socket factory for 'https' protocol	<p>When a client fails with the following exception:</p> <pre>java.io.IOException: No socket factory for 'https' protocol at org.apache.axis.transport.http.HTTPSender.getSocket(HTTPSender.java:...) org.apache.axis.transport.http.HTTPSender.writeToSocket(HTTPSender.java:...) org.apache.axis.transport.http.HTTPSender.invoke(HTTPSender.java:...)</pre> <p>FIXME - it may have happened because...</p>

Error Code	Definition
No client transport named 'https' found	<p>When a client fails with the following exception:</p> <pre>No client transport named 'https' found at org.apache.axis.client.AxisClient.invoke(AxisClient.java:170) at org.apache.axis.client.Call.invokeEngine(Call.java:2726)</pre> <p>The client is most likely loading an incorrect client-config.wsdd configuration file.</p>
ConcurrentModificationException in Tomcat 5.0.x	<p>If the following exception is visible in the Tomcat logs at startup, it might cause the HTTPSValve to fail:</p> <pre>java.util.ConcurrentModificationException at java.util.HashMap\$HashIterator.nextEntry(HashMap.java:782) at java.util.HashMap\$EntryIterator.next(HashMap.java:824) at java.util.HashMap.putAllForCreate(HashMap.java:424) at java.util.HashMap.clone(HashMap.java:656) at mx4j.server.DefaultMBeanRepository.clone(DefaultMBeanRepository.</pre> <p>The HTTPSValve might fail with the following exception:</p> <pre>java.lang.NullPointerException at org.apache.coyote.tomcat5.CoyoteRequest.setAttribute(CoyoteRequestFacade.java:100) at org.apache.coyote.tomcat5.CoyoteRequestFacade.setAttribute(CoyoteRequestFacade.java:100) at org.globus.tomcat.coyote.valves.HTTPSValve.expose(HTTPSVAlve.java:100)</pre> <p>These exceptions will prevent the transport security from working properly in Tomcat.</p>
java.net.SocketException: Invalid argument or cannot assign requested address	<p>FIXME - what causes this?</p>
GAR deploy/undeploy fails with container is running error	<p>A GAR file can only be deployed or undeployed locally while the container is off. However, GAR deployment fails with this error even if the container is off. This usually happens if the container has crashed or was stopped from cleaning up its state files.</p>

3. Errors

Table 11.2. GRAM4 Errors

Error Code	Definition	Possible Solutions
globusrun-ws - error querying job state	<p>During job submission, an error like this occurs:</p> <pre>globusrun-ws failed: Delegating user creden- tials...Done. Sub- mitting job...Done. Job ID: xxxx Termin- ation time: xxxx Current job state: Unsubmitted globus- run-ws: Error querying job state globus_soap_mes- sage_module: Failed sending request ManagedJobPort- Type_GetMul- tipleResourceProper- ties. globus_xio: An end of file oc- curred</pre>	<p>Periodically, globusrun-ws will query the GRAM service to check on the job state. The "End of file" indicates that the GRAM server dropped a connection when globusrun-ws tried to read a response. This could be caused by temporary network issues between the client and service, or possibly caused by an overloaded service host.</p>
globusrun-ws - error querying job state	<p>During job submission, an error like this occurs:</p> <pre>globusrun-ws failed: Delegating user creden- tials...Done. Sub- mitting job...Done. Job ID: xxxx Termin- ation time: xxxx Current job state: Unsubmitted globus- run-ws: Error querying job state globus_soap_mes- sage_module: Failed sending request ManagedJobPort- Type_GetMul- tipleResourceProper- ties. globus_xio: System error in read: Connection reset by peer glo- bus_xio: A system call failed: Connec- tion reset by peer</pre>	<p>Periodically, globusrun-ws will query the GRAM service to check on the job state. The</p> <pre>System error in read: Connection reset by peer</pre> <p>indicates that the GRAM server dropped the connection while trying to write the response. This could be caused by temporary network issues between the client and service, or possibly caused by an overloaded service host.</p>

Error Code	Definition	Possible Solutions
glo- bus- run- ws - error sub- mit- ting job	During job submission, an error like this occurs: globusrun-ws -Ft PBS -F ht- tps://host.teragrid.org:8444 -submit -b -f /tmp/wsgram.rsl -o /tmp/wsgram.epr failed: Submitting job...Failed. globusrun-ws: Error submitting job globus_soap_message_module: Failed sending request ManagedJobFactoryPortType_createManagedJob. globus_xio: Operation was canceled globus_xio: Operation timed out	The Operation timed out indicates that the GRAM service was not able to accept the job request and respond in time. This could be caused by temporary network issues between the client and service, or possibly caused by an overloaded service host.

Chapter 12. Usage statistics collection by the Globus Alliance

1. GRAM4-specific usage statistics

The following usage statistics are sent by default in a UDP packet (in addition to the GRAM component code, packet version, timestamp, and source IP address) at the end of each job (i.e. when Done, Failed, UserTerminateDone or UserTerminateFailed state is entered).

- job creation timestamp (helps determine the rate at which jobs are submitted)
- *scheduler* type (Fork, *PBS*, *LSF*, *Condor*, etc...)
- jobCredentialEndpoint present in *RSL* flag (to determine if server-side user proxies are being used)
- fileStageIn present in RSL flag (to determine if the staging in of files is used)
- fileStageOut present in RSL flag (to determine if the staging out of files is used)
- fileCleanUp present in RSL flag (to determine if the cleaning up of files is used)
- CleanUp-Hold requested flag (to determine if streaming is being used)
- job type (Single, Multiple, MPI, or Condor)
- gt2 error code if job failed (to determine common scheduler script errors users experience)
- fault class name if job failed (to determine general classes of common faults users experience)

If you wish to disable this feature, please see the "Usage Statistics Configuration" section of [Configuring Java WS Core](#) for instructions.

Also, please see our [policy statement](#)¹ on the collection of usage statistics.

¹ ../../Usage_Stats.html

Glossary

B

batch scheduler See the definition for scheduler

C

Condor A job scheduler mechanism supported by GRAM. See <http://www.cs.wisc.edu/condor/> for more information.

J

job description Term used to describe a GRAM4 job for GT4.

L

LSF A job scheduler mechanism supported by GRAM.
For more information, see <http://www.platform.com/Products/Platform.LSF.Family/Platform.LSF/>.

M

multijob A job that is itself composed of several executable jobs; these are processed by the MMJS subjob.
See also [MMJS subjob](#)¹⁰.

P

Portable Batch System (PBS) A job scheduler mechanism supported by GRAM. For more information, see <http://www.openpbs.org>.

R

Resource Specification Language (RSL) Term used to describe a GRAM job for GT2 and GT3. (Note: This is not the same as RLS - the Replica Location Service)

S

scheduler Term used to describe a job scheduler mechanism to which GRAM interfaces. It is a networked system for submitting, controlling, and monitoring the workload of batch jobs in one or more computers. The jobs or tasks are scheduled for execu-

⁷ <http://www.platform.com/Products/Platform.LSF.Family/Platform.LSF/>

¹⁰ #mmjs-subjob

tion at a time chosen by the subsystem according to an available policy and availability of resources. Popular job schedulers include Portable Batch System (PBS), Platform LSF, and IBM LoadLeveler.

scheduler adapter

The interface used by GRAM to communicate/interact with a job scheduler mechanism. In GT 4.x, this is both the perl submission scripts and the SEG program.

Scheduler Event Generator (SEG)

The Scheduler Event Generator (SEG) is a program which uses scheduler-specific monitoring modules to generate job state change events. Depending on scheduler-specific requirements, the SEG may need to run with privileges to enable it to obtain scheduler event notifications. As such, one SEG runs per scheduler resource. For example, on a host which provides access to both PBS and fork jobs, two SEGs, running at (potentially) different privilege levels will be running. One SEG instance exists for any particular scheduled resource instance (one for all homogeneous PBS queues, one for all fork jobs, etc). The SEG is implemented in an executable called the globus-scheduler-event-generator, located in the Globus Toolkit's libexec directory.

superuser do (sudo)

Allows a system administrator to give certain users (or groups of users) the ability to run some (or all) commands as root or another user while logging the commands and arguments. See <http://www.courtesan.com/sudo/> for more information.

W

Web Services Addressing (WSA)

The WS-Addressing specification defines transport-neutral mechanisms to address web services and messages. Specifically, it defines XML elements to identify web service endpoints and to secure end-to-end endpoint identification in messages. See the [W3C WS Addressing Working Group](http://www.w3.org/2002/ws/addr/)¹⁴ for details.

¹⁴ <http://www.w3.org/2002/ws/addr/>

Index

A

audit logging, 17

C

configuration interface, 3
 default local resource manager, 11
 disabling local resource manager adapter, 12
 file system mapping, GRAM4 and GridFTP, 10
 GRAM4 with RFT, 11
 JNDI, 6
 common job factory, 7
 local resource manager, 7
 job lifetime, 12
 non-default, 4
 authorization, 6
 credentials, 4
 GridFTP server, 4
 gridmap, 5
 port, 4
 RFT, 5
 scheduler-specific, 9
 security descriptor, 8
 thread-pools, 14
 typical, 3
 scheduler adapters, 3
 sudo, 3
 web service deployment descriptor, 9
 WS MDS Index
 default auto-registration, 13
 manually registering, 14
configuring, 3
 default local resource manager, 11
 disabling local resource manager adapter, 12
 file system mapping, GRAM4 and GridFTP, 10
 GRAM4 with RFT, 11
 JNDI, 6
 common job factory, 7
 local resource manager, 7
 job lifetime, 12
 non-default, 4
 authorization, 6
 credentials, 4
 GridFTP server, 4
 gridmap, 5
 port, 4
 RFT, 5
 scheduler-specific, 9
 security descriptor, 8
 thread-pools, 14

typical, 3
 scheduler adapters, 3
 sudo, 3
 web service deployment descriptor, 9
WS MDS Index
 default auto-registration, 13
 manually registering, 14

D

debugging
 logging, 31
deploying, 15
 tomcat, 15

E

errors, 34, 37

I

installing
 prerequisites, 1
 GridFTP, 2
 local scheduler, 1
 RFT, 2
 scheduler adapter, 1
 sudo, 1
 transport level security, 1

J

job description extensions, 23

L

logging
 CEDPS-compliant, 31
 debugging, 31

P

performance guide, 16
 server-side, 16

S

SoftEnv, 28

T

troubleshooting, 33
 check container log, 33
 check documentation, 33
 errors, 33
 mailing lists, 33

U

usage statistics, 40

GT 4.2.1 GRAM4: User's Guide

GT 4.2.1 GRAM4: User's Guide

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.

Table of Contents

1. Using GRAM4	1
1. Generating a valid proxy	1
2. Delegating credentials	1
3. Local resource managers interfaced by a GRAM4 installation	2
2. Job description overview	4
1. Job Description Schema Reference	4
2. Single-Job Description	4
3. Multi-Job Description	4
4. Staging Directives	5
5. Substitution Variables	6
6. Extensions	6
3. Submitting jobs	12
1. Simple interactive job	12
2. Simple batch job	12
3. Using a contact string	13
4. Streaming output	13
5. Using a job description	14
6. Using a contact string in the job description	15
7. Specifying a local resource manager	15
8. Using job credentials	16
9. Job with staging	17
10. Specifying a local user id in the job description	19
11. Using substitution variables	19
12. Using custom job description extensions	19
13. Multi-Job	20
14. Submitting MPI Jobs	21
4. Getting information about jobs	22
1. Resource properties of a job	22
2. List of all jobs in a GRAM4 instance	24
5. Terminating jobs	25
1. Jobs in batch mode	25
2. Jobs in interactive mode	25
6. Job lifetime	26
1. Server-side settings that impact job lifetime	26
2. Client-side information	27
7. Job hold and release	28
8. Client-Side Generated Submission ID	30
9. Specifying SoftEnv keys in the job description	31
10. Job and process rendezvous	33
I. GRAM4 Commands	34
globusrun-ws	35
11. Troubleshooting	42
1. Troubleshooting tips	42
2. Java WS Core Errors	43
3. Errors	46
12. Known Problems in GRAM4	49
1. Known Problems	49
13. Usage statistics collection by the Globus Alliance	53
1. GRAM4-specific usage statistics	53
Glossary	54
Index	56

List of Tables

11.1. Java WS Core Errors	44
11.2. GRAM4 Errors	47

Chapter 1. Using GRAM4

1. Generating a valid proxy

In order to generate a valid proxy file, use the `grid-proxy-init` tool available under `$GLOBUS_LOCATION/bin`:

```
% bin/grid-proxy-init
Your identity: /O=Grid/OU=GlobusTest/OU=simpleCA.mymachine/OU=mymachine/CN=John Doe
Enter GRID pass phrase for this identity:
Creating proxy ..... Done
Your proxy is valid until: Tue Oct 26 01:33:42 2004
```

2. Delegating credentials

There are three different uses of delegated credentials:

1. for use by the *MEJS* to create a remote user proxy
2. for use by the MEJS to contact RFT
3. for use by RFT to contact the GridFTP servers. The EPRs to each of these are specified in three job description elements -- they are `jobCredentialEndpoint`, `stagingCredentialEndpoint`, and `transferCredentialEndpoint` respectively. Please [Job Description Schema Reference](#)¹ and [RFT transfer request schema](#)² documentation for more details about these elements.

The `globusrun-ws` client can either delegate these credentials automatically for a particular job, or it can reuse pre-delegated credentials (see next paragraph) through the use of command-line arguments for specifying the credentials' EPR files. Please see the [GRAM4 Commands](#) for details on these command-line arguments.

It is possible to use delegation command-line tools to obtain and refresh delegated credentials in order to use them when submitting jobs to GRAM4. This, for instance, enables the submission of many jobs using a shared set of delegated credentials. This can significantly decrease the number of remote calls for a set of jobs, thus improving performance.

The following example shows how to delegate credentials. `globus-credential-delegate` delegates to the specified delegation factory on `lucky0.mcs.anl.gov`, prints some information and stores the endpoint reference of the delegated credentials into the file `delegCred.epr`

```
[martin@osg-test1 ~]$ globus-credential-delegate \  
> -s https://lucky0.mcs.anl.gov:8443/wsrp/services/DelegationFactoryService \  
> delegCred.epr  
Delegated credential EPR:  
Address: https://lucky0.mcs.anl.gov:8443/wsrp/services/DelegationService  
Reference property[0]:  
<ns1:DelegationKey xmlns:ns1="http://www.globus.org/08/2004/delegationService">  
  55e2a450-58be-11dd-b83c-e4ec640dfe13  
</ns1:DelegationKey>
```

¹ [../schemas/gram_job_description.html](#)

² [../schemas/rft_types.html](#)

To destroy the delegated credential use `wsrf-destroy`:

```
[martin@osg-test1 jobs]$ wsrf-destroy -e delegCred.epr
Destroy operation was successful
```

For more information about the delegation command-line tools see [Command-line tools](#)

3. Local resource managers interfaced by a GRAM4 installation

A GRAM4 instance can interface to more than one local resource manager (LRM), as shown in the previous section. A user can explicitly specify what LRM should be used for a job. But in a larger Grid it might be confusing for users to remember which LRM's are available on which machines.

That's why GRAM4 configures a default local resource manager, which is used for job submission if the client didn't explicitly specify one.

3.1. Finding available local resource managers

You can check the resource property `availableLocalResourceManagers` of a GRAM4 factory service to get that information. Replace host and port in the below example to query against other containers:

```
[martin@osg-test1 ~]$ globus-wsrf-get-property \
-s https://osg-test1.unl.edu:8443/wsrf/services/ManagedJobFactoryService \
"{http://www.globus.org/namespaces/2008/03/gram/job}availableLocalResourceManagers"
```

The result on that machine is (formatted for better readability) shows that the local resource managers Fork, Multi, Condor and PBS are available:

```
<ns1:availableLocalResourceManagers
  xmlns:ns1="http://www.globus.org/namespaces/2008/03/gram/job">
  <ns1:localResourceManager>Fork</ns1:localResourceManager>
  <ns1:localResourceManager>Multi</ns1:localResourceManager>
  <ns1:localResourceManager>Condor</ns1:localResourceManager>
  <ns1:localResourceManager>PBS</ns1:localResourceManager>
</ns1:availableLocalResourceManagers>
```

A more typical result in a production environment is probably Fork, Multi and just one additional LRM like Condor, PBS or LSF.

3.2. Finding the default local resource manager

You can check the resource property `defaultLocalResourceManagers` of a GRAM4 factory service to get that information. Replace host and port in the below example to query against other containers:

```
[martin@osg-test1 ~]$ globus-wsrf-get-property \
-s https://osg-test1.unl.edu:8443/wsrf/services/ManagedJobFactoryService \
"{http://www.globus.org/namespaces/2008/03/gram/job}localResourceManager"
```

The result on that machine shows that PBS is the default local resource managers:

```
<ns1:localResourceManager xmlns:ns1="http://www.globus.org/namespaces/2008/03/gram/job">  
  PBS  
</ns1:localResourceManager>
```

Chapter 2. Job description overview

1. Job Description Schema Reference

This chapter gives an overview of job description, but does not explain all elements that can be put into a job description. For detailed information about all elements and their ordering please see [Job Description Schema documentation](#)¹.

2. Single-Job Description

The general form of a *job description* used to start a single job (meant for creating a Managed Executable Job Resource instance) is as follows:

```
<job>
  <!--put additional elements here-->
  <executable><!--put executable here--></executable>
  <!--put additional elements here-->
</job>
```

Here is a basic example of a job description for a single-job:

```
<job>
  <executable>bin/echo</executable>
  <argument>Testing</argument>
  <argument>1...2...3</argument>
  <stdout>${GLOBUS_USER_HOME}/stdout</stdout>
  <stderr>${GLOBUS_USER_HOME}/stderr</stderr>
</job>
```

3. Multi-Job Description

The general form of a job description used to start a multi-job (meant for creating a Managed Multi Job Resource instance) is as follows:

```
<multiJob>
  <!--Put subjob default elements here.-->
  <job>
    <factoryEndpoint
      xmlns:gram="http://www.globus.org/namespaces/2008/03/gram/job"
      xmlns:wsa="http://www.w3.org/2005/08/addressing">
      <wsa:Address>
        <!--put ManagedJobFactoryService address here-->
      </wsa:Address>
      <wsa:ReferenceParameters>
        <gram:ResourceID><!--put scheduler type here--></gram:ResourceID>
      </wsa:ReferenceParameters>
    </factoryEndpoint>
```

¹ ../schemas/gram_job_description.html

```

    <executable><!--put executable path here--></executable>
  </job>
  <!--put additional job elements here-->
</multiJob>

```

Here is a basic example of a job description for a multi-job:

```

<multiJob>
  <executable>/bin/echo</executable>
  <stdout>${GLOBUS_USER_HOME}/stdout</stdout>
  <stderr>${GLOBUS_USER_HOME}/stderr</stderr>
  <job>
    <factoryEndpoint
      xmlns:gram="http://www.globus.org/namespaces/2008/03/gram/job"
      xmlns:wsa="http://www.w3.org/2005/08/addressing">
      <wsa:Address>
        https://mymachine.mydomain.com:8443/wsrf/services/ManagedJobFactoryService
      </wsa:Address>
      <wsa:ReferenceParameters>
        <gram:ResourceID>PBS</gram:ResourceID>
      </wsa:ReferenceParameters>
    </factoryEndpoint>
    <argument>Testing</argument>
    <argument>1...2...3</argument>
  </job>
  <job>
    <factoryEndpoint
      xmlns:gram="http://www.globus.org/namespaces/2008/03/gram/job"
      xmlns:wsa="http://www.w3.org/2005/08/addressing">
      <wsa:Address>
        https://myothermachine.myotherdomain.org:8443/wsrf/services/ManagedJobFact
      </wsa:Address>
      <wsa:ReferenceParameters>
        <gram:ResourceID>Pbs</gram:ResourceID>
      </wsa:ReferenceParameters>
    </factoryEndpoint>
    <argument>Hi There!</argument>
    <argument>Dear John!</argument>
  </job>
</multiJob>

```

4. Staging Directives

The GRAM4 job description schema imports types from the RFT job description schema for specifying staging directives (i.e. `fileStageIn`, `fileStageOut`, and `fileCleanUp`). See [Chapter 3, RFT transfer request](#) for details on these imported types.

Since `fileStageIn` and `fileStageOut` are of type `TransferRequestType`² and `fileCleanUp` is of type `DeleteRequestType`³, mentally replace "transferRequest" with "fileStageIn" or "fileStageOut", and "deleteRequest" with "fileCleanUp" in the [RFT transfer request documentation](#). The 'Request Options' section is of particular usefulness.

² [../../data/rft/rft_job_description.html#type_TransferRequestType](#)

³ [../../data/rft/rft_job_description.html#type_DeleteRequestType](#)

5. Substitution Variables

Job description variables are special strings in a job description that are replaced by the GRAM service with values that the client-side does not *a priori* know.

The set of variables is fixed in the gram service implementation. This is different from previous implementations of *RSL* substitutions in GT2 and GT3, where a user could define a new variable for use inside a job description document. This was done to preserve the simplicity of the job description XML schema (relatively to the GT3.2 RSL schema), which does not require a specialized XML parser to serialize a job description document.

Valid variables and their description:

GLOBUS_USER_HOME	The path to the home directory for the local account/user
GLOBUS_USER_NAME	The local account the job is running under
GLOBUS_JOB_ID	UUID of the job, created on the server-side
GLOBUS_SCRATCH_DIR	An alternative directory where a file system is shared with the compute nodes that a user might want to use. Typically it will provide more space than the users default HOME dir. This directory's value may contain <code>\${GLOBUS_USER_HOME}</code> , which will be replaced with value of that substitution.
GLOBUS_LOCATION	Path to the Globus Toolkit installation

Variable substitutions may not occur in all job description attributes. Their use is restricted to those which contain arbitrary string data and which may be used to access the local resource associated with a job. The list of attributes which may contain variables is:

- executable
- directory
- argument
- environment
- stdin
- stdout
- stderr
- libraryPath
- fileStageIn
- fileStageOut
- fileCleanUp

6. Extensions

To allow adding features to GRAM4 while avoiding breaking compatibility between versions, an extensibility point was included in the job description schema. This appears as the `<extensions>` element at the bottom of a job description document. Starting with version 4.2.1 of the Globus Toolkit, GRAM4 will support both a number of specific

extensions as well as generic constructs that can be used for passing custom values to the resource manager/scheduler adapter Perl modules.

6.1. Supported Extensions

The following are specific supported extensions to the GRAM4 job description schema. They do not require any modification of the resource manager/scheduler adapter Perl modules.

6.1.1. Condor specific parameters

If a user submits a job to Gram4 specifying Condor as local resource manager a condor-specific job description will be created from the XML job description which will be used when the job is submitted to Condor. A user can influence the creation of the condor-specific job description by adding `condorsubmit` elements to the extensions element:

```
<job>
  ...
  <extensions>
    <condorsubmit name="nameOfAnElement">valueOfTheElement</condorsubmit>
  </extensions>
</job>
```

More than one `condorsubmit` element can be placed in the extensions element.

The following example shows how to set a different `Requirements` element than is added by default. By default Gram4 adds a `Requirement` element and sets the parameter `OpSys` and `Arch` to values that fit the head-node where Gram4 is running. If e.g. the operating system on the head-node is Linux and the architecture is X86_64, the `Requirements` element in a Condor job description will look like

```
Requirements=OpSys == "LINUX" && Arch == "X86_64"
```

If this is not what is needed, requirements can be added as follows:

```
<job>
  <executable>/bin/date</executable>
  <extensions>
    <condorsubmit name="Requirements">OpSys == "LINUX" &amp;&amp; (Arch == "X86_64" || Arc
  </extensions>
</job>
```

Note that the special char `&` must be coded as `&`.

6.1.2. PBS Node Selection Parameters

Node selection constraints in PBS can be specified in two ways, generally using a construct intended to eventually apply to all resource managers which support node selection, or explicitly by specifying a simple string element. The former will be more portable, but the later will appeal to those familiar with specifying node constraints for PBS jobs.

6.1.2.1. Using the nodes extensions element

To specify PBS node selection constraints explicitly, one can simply construct a single, simple string extension element named `nodes` with a value that conforms to the `#PBS -l nodes=...` PBS job description directive. The `Globus::GRAM::ExtensionsHandler` module will make this available to the PBS adapter script by invoking `$description->{nodes}`. The updated PBS adapter package checks for this value and will create a directive in the PBS job description using this value.

For example the following nodes extensions element

```
...
<extensions>
  <nodes>activemural:ppn=10+5:ia64-compute:ppn=2</nodes>
</extensions>
...
```

will result in the following directive in the PBS job description:

```
#PBS -l nodes=activemural:ppn=10+5:ia64-compute:ppn=2
```

6.1.2.2. Using the resourceAllocationGroup extensions element

To use the generic construct for specifying node selection constraints, use the resourceAllocationGroup element:

```
<extensions>
  <resourceAllocationGroup>
    <!-- Optionally select hosts by type and number... -->
    <hostType>...</hostType>
    <hostCount>...</hostCount>

    <!-- *OR* by host names -->

    <hostName>...</hostName>
    <hostName>...</hostName>
    . . .

    <!-- With a total CPU count for this group... -->
    <cpuCount>...</cpuCount>

    <!-- *OR* an explicit number of CPUs per node... -->
    <cpusPerNode>...</cpusPerNode>
    . . .

    <!-- And a total process count for this group... -->
    <processCount>...</processCount>

    <!-- *OR* an explicit number of processes per node... -->
    <processesPerNode>...</processesPerNode>
  </resourceAllocationGroup>
</extensions>
```

Extension elements specified according to the above pseudo-schema will be converted to an appropriate nodes parameter which will be treated as if an explicit nodes extension element were specified. Multiple resourceAllocationGroup elements may be specified. This will simply append the constraints to the nodes parameter with a '+' separator. Note that one cannot specify both hostType/hostCount and hostName elements. Similarly, one cannot specify both processCount and processesPerNode elements.

Here are some examples of using resourceAllocationGroup:

```
<!-- #PBS -l nodes=1:ppn=10 -->
<!-- 10 processes -->
<extensions>
  <resourceAllocationGroup>
    <cpuCount>10</cpuCount>
    <processCount>10</processCount>
  </resourceAllocationGroup>
</extensions>

<!-- #PBS -l nodes=activemural:ppn=10+5:ia64-compute:ppn=2 -->
<!-- 1 process (process default) -->
<extensions>
  <resourceAllocationGroup>
    <hostType>activemural</hostType>
    <cpuCount>10</cpuCount>
  </resourceAllocationGroup>
  <resourceAllocationGroup>
    <hostType>ia64-compute</hostType>
    <hostCount>5</hostCount>
    <cpusPerHost>2</cpusPerHost>
  </resourceAllocationGroup>
</extensions>

<!-- #PBS -l nodes=vis001:ppn=5+vis002:ppn=5+comp014:ppn=2+comp015:ppn=2 -->
<!-- 15 total processes -->
<extensions>
  <resourceAllocationGroup>
    <hostName>vis001</hostName>
    <hostName>vis002</hostName>
    <cpuCount>10</cpuCount>
    <processesPerHost>5</processesPerHost>
  </resourceAllocationGroup>
  <resourceAllocationGroup>
    <hostName>comp014</hostName>
    <hostName>comp015</hostName>
    <cpusPerHost>2</cpusPerHost>
    <processCount>5</processCount>
  </resourceAllocationGroup>
</extensions>
```

6.2. Additional Extension Constructs

The following are general constructs that are supported by the ExtensionsHandler.pm Perl module. Although no modifications to ExtensionsHandler.pm are required, you will need to edit the appropriate resource manager/scheduler adapter Perl module as necessary to affect the submission of jobs to the local resource manager/batch scheduler.

The GRAM4 job description schema includes a section for extending the job description with custom elements. To make sense of this in the resource manager adapter Perl scripts, a Perl module named Globus::GRAM::ExtensionsHandler is provided to turn these custom elements into parameters that the adapter scripts can understand.

It should be noted that although non-GRAM XML elements only are allowed in the `<extensions>` element of the job description, the extensions handler makes no distinction based on namespace. Thus, `<foo:myparam>` and `<bar:myparam>` will both be treated as just `<myparam>`.

Familiarity with the adapter scripts is assumed in the following subsections.

6.2.1. Simple String Parameters

Simple string extension elements are converted into single-element arrays with the name of the unqualified tag name of the extension element as the array's key name in the Perl job description hash. Simple string extension elements can be considered a special case of the string array construct in the next section.

For example, adding the following element to the `<extensions>` element of the job description:

```
<extensions>
  <myparam>yahoo!</myparam>
</extensions>
```

will cause the `$description->myparam()` to return the following value:

```
'yahoo!'
```

6.2.2. String Array Parameters

String arrays are a simple iteration of the simple string element construct. If you specify more than one simple string element in the job description, these will be assembled into a multi-element array with the unqualified tag name of the extension elements as the array's key name in the Perl job description hash.

For example:

```
<extensions>
  <myparams>Hello</myparams>
  <myparams>World!</myparams>
</extensions>
```

will cause the `$description->myparams()` to return the following value:

```
[ 'Hello', 'World!' ]
```

6.2.3. Name/Value Parameters

Name/value extension elements can be thought of as string arrays with an XML attribute 'name'. This will cause the creation of a two-dimensional array with the unqualified extension element tag name as the name of the array in the Perl job description hash.

For example:

```
<extensions>
  <myvars name="pi">3.14159</myvars>
  <myvars name="mole">6.022 x 10^23</myvars>
```

```
</extensions>
```

will cause the `$description->myvars()` to return the following value:

```
[ [ 'pi', '3.14159'], ['mole', '6.022 x 10^23'] ]
```

6.3. Supporting custom extensions in the Perl adapter modules.

See the System Administrator's Guide section on [Configuring GRAM4](#) for information on how to customize the resource manager/scheduler adapter Perl modules

Chapter 3. Submitting jobs

1. Simple interactive job

Use the `globusrun-ws` program to submit a simple job without writing a job description document. Use the `-c` argument, a job description will be generated assuming the first arg is the executable and the remaining are arguments. For example:

```
% globusrun-ws -submit -c /bin/touch touched_it
Submitting job...Done.
Job ID: uuid:4a92c06c-b371-11d9-9601-0002a5ad41e5
Termination time: 04/23/2005 20:58 GMT
Current job state: Active
Current job state: CleanUp
Current job state: Done
Destroying job...Done.
```

Confirm on the server-side that the job worked by verifying the file was touched:

```
% ls -l ~/touched_it
-rw-r--r--  1 smartin globdev 0 Apr 22 15:59 /home/smartin/touched_it

% date
Fri Apr 22 15:59:20 CDT 2005
```

Note: You did not tell `globusrun-ws` where to run your job, so the default of `localhost` was used.

Also note, that `globusrun-ws` destroyed the job after it was fully processed.

We call this kind of job interactive, because `globusrun-ws` does not return after submission. It subscribes for status update notifications of the job and informs the user about a status change as soon as it changes. Once it gets the information the the job has been fully processed it destroys the job, which means that internal state belonging to the job is cleaned up on the server-side.

2. Simple batch job

Now we submit the same job in batch mode. The option `-b` tells `globusrun-ws` to submit in batch mode, and `-o` causes that a reference (also named `EndpointReference`) of the is stored in the file `myJob.epr`, which is used later to check for job status and to terminate the job.

```
% globusrun-ws -submit -b -o myJob.epr -c /bin/touch touched_it
globusrun-ws -submit -b -o myJob.epr -c /bin/date
Submitting job...Done.
Job ID: uuid:5ad25b06-22f7-11dd-8482-0013d4c3b957
Termination time: 05/16/3008 03:22 GMT
```

`globusrun-ws` returns as soon as the job has been submitted. No status information is printed. The user has to do additional calls to check the status of the job. See section [Get information about jobs](#) how to do that.

The user should also demand job termination when the job is done, to clean up internal state belonging to the job on the server-side. See section [Jobs in batch mode](#) in the [Terminating jobs](#) section how to do that.

Again the job had been submitted to localhost

3. Using a contact string

Use globusrun-ws to submit the same touch job, but this time tell globusrun-ws to run the job on another machine (lucky0.mcs.anl.gov:8443). A GT4 server with GRAM4 installed must run on that machine and listen on port 8443.

```
% globusrun-ws -submit \  
  -F https://lucky0.mcs.anl.gov:8443/wsrp/services/ManagedJobFactoryService \  
  -c /bin/touch touched_it  
Submitting job...Done.  
Job ID: uuid:3050ad64-b375-11d9-be11-0002a5ad41e5  
Termination time: 04/23/2005 21:26 GMT  
Current job state: Active  
Current job state: CleanUp  
Current job state: Done  
Destroying job...Done.
```

Type globusrun-ws -help to learn the details about the contact string.

4. Streaming output

A user can request that the output of the program is sent back directly to the client as soon as it's available. This is useful if a user does not want to do additional file staging for a quick job. To enable this, specify the `-s` option.

```
[martin@osg-test1 ~]$ globusrun-ws -submit \  
  -F https://lucky0.mcs.anl.gov:8443/wsrp/services/ManagedJobFactoryService \  
  -s -c /bin/echo hello world!  
Delegating user credentials...Done.  
Submitting job...Done.  
Job ID: uuid:1731f602-22fe-11dd-879c-0013d4c3b957  
Termination time: 05/16/3008 04:10 GMT  
Current job state: Active  
Current job state: CleanUp-Hold  
hello world!  
Current job state: CleanUp  
Current job state: Done  
Destroying job...Done.  
Cleaning up any delegated credentials...Done.
```

If you want the output of the job to be written to a local file instead of the terminal you'll have to add the `-so` option:

```
[martin@osg-test1 ~]$ globusrun-ws -submit \  
  -F https://lucky0.mcs.anl.gov:8443/wsrp/services/ManagedJobFactoryService \  
  -s -so job.out -c /bin/echo hello world!  
Delegating user credentials...Done.  
Submitting job...Done.  
Job ID: uuid:1731f602-22fe-11dd-879c-0013d4c3b957  
Termination time: 05/16/3008 04:10 GMT  
Current job state: Active  
Current job state: CleanUp-Hold  
Current job state: CleanUp  
Current job state: Done
```

```
Destroying job...Done.  
Cleaning up any delegated credentials...Done.
```

Check the output in the specified file:

```
[martin@osg-test1 ws-gram]$ cat job.out  
hello, world!
```

Note that a GridFTP server must be running on the remote machine (lucky0) to enable streaming.

Note that streaming output adds some overhead to the submission and will probably be significantly slower compared to a job without streaming. An alternative to streaming is to use staging to transport the output of the executable back to the client. This however requires that a GridFTP server is running on the client machine.

5. Using a job description

The specification of a job to submit is to be written by the user in a job description XML file.

Here is an example of a simple job description:

```
<job>  
  <executable>/bin/echo</executable>  
  <argument>this is an example_string </argument>  
  <argument>Globus was here</argument>  
  <stdout>${GLOBUS_USER_HOME}/stdout</stdout>  
  <stderr>${GLOBUS_USER_HOME}/stderr</stderr>  
</job>
```

Tell globusrun-ws to read the job description from a file, using the -f argument:

```
% bin/globusrun-ws -submit -f simple.xml  
Submitting job...Done.  
Job ID: uuid:c51fe35a-4fa3-11d9-9cfc-000874404099  
Termination time: 12/17/2004 20:47 GMT  
Current job state: Active  
Current job state: CleanUp  
Current job state: Done  
Destroying job...Done.
```

Note the usage of the substitution variable `${GLOBUS_USER_HOME}` which resolves to the user home directory.

Here is an example with more job description parameters:

```
<?xml version="1.0" encoding="UTF-8"?>  
<job>  
  <executable>/bin/echo</executable>  
  <directory>/tmp</directory>  
  <argument>12</argument>  
  <argument>abc</argument>  
  <argument>34</argument>  
  <argument>this is an example_string </argument>  
  <argument>Globus was here</argument>  
  <environment>  
    <name>PI</name>
```

```
    <value>3.141</value>
  </environment>
  <stdin>/dev/null</stdin>
  <stdout>stdout</stdout>
  <stderr>stderr</stderr>
  <count>2</count>
</job>
```

Note that in this example, a `<directory>` element specifies the current directory for the execution of the command on the execution machine to be `/tmp`, and the standard output is specified as the relative path `stdout`. The output is therefore written to `/tmp/stdout`:

```
% cat /tmp/stdout
12 abc 34 this is an example_string  Globus was here
```

6. Using a contact string in the job description

Instead of specifying the contact string on the command-line, you can also put it in the job description:

```
<job xmlns:wsa="http://www.w3.org/2005/08/addressing">
  <factoryEndpoint>
    <wsa:Address>
      https://osg-test1.unl.edu:8443/wsrf/services/ManagedJobFactoryService
    </wsa:Address>
  </factoryEndpoint>
  <executable>/bin/date</executable>
</job>
```

Submit the job with the following command (assuming the above description has been stored in the file `job.xml`):

```
% bin/globusrun-ws -submit -f job.xml
```

Note

This time you don't have to specify the `-F` option.

7. Specifying a local resource manager

Note that at this point you didn't specify any local resource manager related information. If a user does not specify anything then the job is run by the default local resource manager, that is defined on the server-side. If an admin e.g. configured Condor as default local resource manager, then the jobs submitted so far will be managed by Condor on the server-side.

Check the section [Local resource managers interfaced by a GRAM4 installation](#) to find out which local resource managers are available in a GRAM4 installation and which one is configured as the default.

7.1. Submitting to the default local resource manager

As said, if you want to submit a job to the default local resource manager, all you have to do is to just NOT specify any local resource manager in your submission, neither in the job description, nor on the command-line. The above examples show how to do it.

7.2. Submitting to a non-default local resource manager

If you want to submit a job to a non-default local resource manager, or if you just want to be explicit in what you specify, you'll have to specify the local resource manager in your submission. Using `globusrun-ws`, there are two ways to specify a local resource manager:

- as command-line argument of `globusrun-ws` (`-Ft <lrn>`)
- in the `factoryEndpoint` element in the job description

Example: the following job will be submitted to Condor:

```
globusrun-ws -submit \
  -F osg-test1.unl.edu:8443/wsrf/services/ManagedJobFactoryService \
  -Ft Condor \
  -c /bin/date
```

Or with a job description that contains a `factoryEndpoint`:

```
<job xmlns:wsa="http://www.w3.org/2005/08/addressing"
  xmlns:gram="http://www.globus.org/namespaces/2008/03/gram/job">
  <factoryEndpoint>
    <wsa:Address>
      https://osg-test1.unl.edu:8443/wsrf/services/ManagedJobFactoryService
    </wsa:Address>
    <wsa:ReferenceParameters>
      <gram:ResourceID>Condor</gram:ResourceID>
    </wsa:ReferenceParameters>
  </factoryEndpoint>
  <executable>/bin/date</executable>
</job>
```

Submit that job (assuming the description is stored in the file `myJob.xml`):

```
globusrun-ws -submit -f myJob.xml
```

8. Using job credentials

If your jobs needs a user proxy certificate e.g. because it calls other programs that require user authentication/authorization, you can delegate a job credential which is used to create a user proxy certificate on the server-side and which can then be used by your job. You can delegate a job credential per job, or you can make use of an already existing delegated credential.

8.1. Job delegation per job

You can let `globusrun-ws` delegate a credential for a particular job for you using the `-J` option:

```
globusrun-ws -submit -J -c /bin/[martin@osg-test1 ~]$ globusrun-ws -submit -J -c /bin/date
Delegating user credentials...Done.
Submitting job...Done.
Job ID: uuid:b0de558e-87f3-11dd-a85a-0013d4c3b957
Termination time: 09/21/3008 15:41 GMT
Current job state: Active
```

```
Current job state: CleanUp
Current job state: Done
Destroying job...Done.
Cleaning up any delegated credentials...Done.
```

Note that globusrun-ws also requests the deletion of the delegated credential after the job finished. This will also remove the user proxy file that had been created for the job.

8.2. Making use of an already delegated credential

If you already delegated a credential and have an EPR of it you can tell globusrun-ws to use it using the `-Jf` option. Note that the delegated credential must be located in the same GT server you are using to submit your job. See section [Delegating credentials](#) for how to delegate a credential.

```
[martin@osg-test1 ~]$ globusrun-ws -submit -Jf delegCred.epr -c /bin/date
Submitting job...Done.
Job ID: uuid:28dfbb5c-87f6-11dd-a34d-0013d4c3b957
Termination time: 09/21/3008 15:58 GMT
Current job state: Active
Current job state: CleanUp
Current job state: Done
Destroying job...Done.
```

Note that globusrun-ws does not delegate this time, and also does not request the deletion of the delegated credential. Also the user proxy file, which is created from the credential won't be removed; it is bound to the delegated credential, and not to the job. You'll have to explicitly destroy the delegated credential in this case which then will also remove the user proxy file. See section [Delegating credentials](#) how to request the destruction of a delegated credential.

9. Job with staging

In order to do file staging one must add specific elements to the job description and delegate credentials appropriately (see [Section 2, "Delegating credentials"](#)). The file transfer directives follow the [RFT syntax](#)¹, which allows only for third-party transfers. Each file transfer must therefore specify a source URL and a destination URL. URLs are specified as GridFTP URLs (for remote files) or as file URLs (for files local to the service--these are converted internally to full GridFTP URLs by the service).

For instance, in the case of staging a file *in*, the source URL would be a GridFTP URL (for instance `gsiftp://job.submitting.host:2811/tmp/mySourceFile`) resolving to a source document accessible on the file system of the job submission machine (for instance `/tmp/mySourceFile`). At run-time the Reliable File Transfer service used by the MEJS on the remote machine would reliably fetch the remote file using the GridFTP protocol and write it to the specified local file (for instance `file:///${GLOBUS_USER_HOME}/my_transferred_file`, which resolves to `~/my_transferred_file`). Here is how the stage-in directive would look like:

```
<fileStageIn>
  <transfer>
    <sourceUrl>gsiftp://job.submitting.host:2811/tmp/mySourceFile</sourceUrl>
    <destinationUrl>file:///${GLOBUS_USER_HOME}/my_transferred_file</destinationUrl>
  </transfer>
</fileStageIn>
```

¹ ../schemas/rft_types.html

Note: additional RFT-defined quality of service requirements can be specified for each transfer. See the RFT documentation for more information.

Here is an example job description with file stage-in and stage-out:

```
<job>
  <executable>my_echo</executable>
  <directory>${GLOBUS_USER_HOME}</directory>
  <argument>Hello</argument>
  <argument>World!</argument>
  <stdout>${GLOBUS_USER_HOME}/stdout</stdout>
  <stderr>${GLOBUS_USER_HOME}/stderr</stderr>
  <fileStageIn>
    <transfer>
      <sourceUrl>gsiftp://job.submitting.host:2811/bin/echo</sourceUrl>
      <destinationUrl>file:///${GLOBUS_USER_HOME}/my_echo</destinationUrl>
    </transfer>
  </fileStageIn>
  <fileStageOut>
    <transfer>
      <sourceUrl>file:///${GLOBUS_USER_HOME}/stdout</sourceUrl>
      <destinationUrl>gsiftp://job.submitting.host:2811/tmp/stdout</destinationUrl>
    </transfer>
  </fileStageOut>
  <fileCleanup>
    <deletion>
      <file>file:///${GLOBUS_USER_HOME}/my_echo</file>
    </deletion>
  </fileCleanup>
</job>
```

Note that the job description XML does not need to include a reference to the schema that describes its syntax. As a matter of fact it is possible to omit the namespace in the GRAM job description XML elements as well. The submission of this job to the GRAM services causes the following sequence of actions:

1. The `/bin/echo` executable is transferred from the submission machine to the GRAM host file system. The destination location is the HOME directory of the user on behalf of whom the job is executed by the GRAM services (see `<fileStageIn>`).
2. The transferred executable is used to print a test string (see `<executable>`, `<directory>` and the `<argument>` elements) on the standard output, which is redirected to a local file (see `<stdout>`).
3. The standard output file is transferred to the submission machine (see `<fileStageOut>`).
4. The file that was initially transferred during the stage-in phase is removed from the file system of the GRAM installation (see `<fileCleanup>`).

Submit that job (assuming the description is stored in the file `myJob.xml`):

```
globusrun-ws -submit -S -f myJob.xml
```

The flag `-S` tells `globusrun-ws` to delegate credentials so that Gram4 can call the file transfer service RFT on behalf of the submitting user, and that RFT can interact with the gridftp servers on behalf of the submitting user.

If you already delegated credentials (see [Delegating credentials](#) for how to delegate a credential) and have an endpoint reference of that delegated credentials stored in the file `delegCred.epr` and want them to be used for the transfers instead of `globusrun-ws` delegating new credentials, you can tell `globusrun-ws` to use your credentials:

```
globusrun-ws -submit -Sf delegCred.epr -Tf delegCred.epr -f myJob.xml
```

The `-Sf` flag tells that the specified credential is to be used by Gram4 to call RFT on behalf of the user, and the `-Tf` flag tells that the specified credential is to be used by RFT to interact with the GridFTP servers.

10. Specifying a local user id in the job description

If a user has more than one user account on a server and the distinguished name (DN) of the user's certificate is mapped to all these user accounts, a user can specify which local account should be used by GRAM4 for the job submission. By default the first local user account that is defined is used for job submission. If this is not the one that should be used the user must explicitly specify the account to be used. The following dummy job description shows how to do this:

```
<job>
  <localUserId>stu</localUserId>
  <executable>/bin/date</executable>
  <stdout>${GLOBUS_USER_HOME}/stdout</stdout>
  <stderr>${GLOBUS_USER_HOME}/stderr</stderr>
</job>
```

11. Using substitution variables

To allow for customization of values, such as paths, on a per-job basis; a job description substitution variable named "GLOBUS_JOB_ID" can be used.

For example:

```
<job>
  <executable>/bin/date</executable>
  <stdout>/tmp/stdout.${GLOBUS_JOB_ID}</stdout>
  <stderr>/tmp/stderr.${GLOBUS_JOB_ID}</stderr>
  <fileStageOut>
    <transfer>
      <sourceUrl>file:///tmp/stdout.${GLOBUS_JOB_ID}</sourceUrl>
      <destinationUrl>gsiftp://mymachine.mydomain.com/out.${GLOBUS_JOB_ID}</destinationUrl>
    </transfer>
  </fileStageOut>
</job>
```

More information about substitution variables can found [here](#).

12. Using custom job description extensions

Basic support is provided for specifying custom extensions to the job description. There are plans to improve the usability of this feature, but at this time it involves a bit of work.

Specifying the actual custom elements in the job description is trivial. Simply add any elements that you need between the beginning and ending `extensions` tags at the bottom of the job description as in the following basic example:

```
<job>
  <executable>/home/user1/myapp</executable>
  <extensions>
    <myData>
      <flag1>on</flag1>
      <flag2>off</flag2>
    </myData>
  </extensions>
</job>
```

To handle this data, you will have to alter the appropriate Perl scheduler script (i.e. `$GLOBUS_LOCATION/lib/perl/Globus/GRAM/JobManager/fork.pm` for the Fork scheduler, etc...) to parse the data returned from the `$description->extensions()` sub.

For more information about extensions see the [Extensions](#) section.

13. Multi-Job

The job description XML schema allows for specification of a *multijob* i.e. a job that is itself composed of several executable jobs, which we will refer to as *subjobs* (*note*: subjobs cannot be multijobs, so the structure is not recursive). This is useful for instance in order to bundle a group of jobs together and submit them as a whole to a remote GRAM installation.

Note that no relationship can be specified between the subjobs of a multijob. The subjobs are submitted to job factory services in their order of appearance in the multijob description.

Within a [multijob description](#)², each subjob description must come along with an endpoint for the factory to submit the subjob to. This enables the at-once submission of several jobs to different hosts. The factory to which the multijob is submitted acts as an intermediary tier between the client and the eventual executable job factories.

Here is an example of a multijob description:

```
<?xml version="1.0" encoding="UTF-8"?>
<multiJob xmlns:wsa="http://www.w3.org/2005/08/addressing">
  <factoryEndpoint>
    <wsa:Address>
      https://localhost:8443/wsrf/services/ManagedJobFactoryService
    </wsa:Address>
  </factoryEndpoint>
  <directory>${GLOBUS_LOCATION}</directory>
  <count>1</count>

  <job>
    <factoryEndpoint>
      <wsa:Address>https://localhost:8443/wsrf/services/ManagedJobFactoryService</wsa:A
    </factoryEndpoint>
    <executable>/bin/date</executable>
```

² [./schemas/gram_job_description.html#element_multiJob](#)

```

    <stdout>${GLOBUS_USER_HOME}/stdout.p1</stdout>
    <stderr>${GLOBUS_USER_HOME}/stderr.p1</stderr>
    <count>2</count>
</job>

<job>
  <factoryEndpoint>
    <wsa:Address>https://localhost:8443/wsrf/services/ManagedJobFactoryService</wsa:A
  </factoryEndpoint>
  <executable>/bin/echo</executable>
  <argument>Hello World!</argument>
  <stdout>${GLOBUS_USER_HOME}/stdout.p2</stdout>
  <stderr>${GLOBUS_USER_HOME}/stderr.p2</stderr>
  <count>1</count>
</job>
</multiJob>

```

Submit the multi-job with the following command:

```

% bin/globusrun-ws -submit -f test_multi.xml
  Delegating user credentials...Done.
  Submitting job...Done.
  Job ID: uuid:bd9cd634-4fc0-11d9-9ee1-000874404099
  Termination time: 12/18/2004 00:15 GMT
  Current job state: Active
  Current job state: CleanUp
  Current job state: Done
  Destroying job...Done.
  Cleaning up any delegated credentials...Done.

```

Note

When you submit a multi-job you don't have to specify the local resource manager, you can do so though. The fact that it's a multi-job is detected on the server-side and the right "local resource manager" Multi is used automatically.

Note

In this multi-job description the sub-jobs are submitted to the default local resource manager. If you want them to be submitted to a non-default local resource manager you'll have to specify that in an additional ReferenceParameters element in the factoryEndpoint element of each sub-job. See [here](#) for more information about this.

A multijob resource is created by the factory and exposes a set of WSRF resource properties different than the resource properties of an executable job. The state machine of a multijob is also different since the multijob represents the *overall* execution of all the executable jobs it is composed of.

14. Submitting MPI Jobs

This [document from DGrid³](#) describes how to submit MPI batch jobs to compute clusters using GRAM4.

³ http://www.gac-grid.org/project-documents/deliverables/wp1/Run_MPI_Jobs_on_Grid-D_1_4.pdf

Chapter 4. Getting information about jobs

1. Resource properties of a job

A job submitted to GRAM4 has a set of resource properties that contain information about the job a user might be interested in, like

- status of the remote job
- local user id the job is run under
- exit code of the executable after it finished
- information about errors
- ...

The following links give a complete list of all available resource properties of [executable jobs](#) and [multi jobs](#).

The rest of this section assumes that a job had been submitted in batch mode, and that the EPR of the job is available in the file `myJob.epr`.

To check for status of the job, use `globusrun-ws`:

```
[martin@osg-test1 ~]$ globusrun-ws -status -j myJob.epr
Current job state: Active
```

If a job failed, `globusrun-ws` will print an error that indicates what happened:

```
[martin@osg-test1 ~]$ globusrun-ws -status -j myJob.epr
Current job state: Failed
globusrun-ws: Job failed: Invalid executable path "/bin/sleeeeeeep".
```

More general-purpose programs exist that are not limited to resource properties of job resources and that let you get information about the other resource properties:

- **wsrf-get-resource-property**: Get a single resource property
- **wsrf-get-resource-properties**: Get a list of resource properties
- **wsrf-query**: Query the resource property document

The following examples show how to use these commands to get information about an executable job.

Use `wsrf-get-property` to get the state of the job (resource property state):

```
[martin@osg-test1 ~]$ wsrf-get-property -e myJob.epr \
  "{http://www.globus.org/namespaces/2008/03/gram/job/types}state"
<ns1:state xmlns:ns1="http://www.globus.org/namespaces/2008/03/gram/job/types">
  Done
</ns1:state>
```

Get the values of the two resource properties `localUserId` and `localJobId` in one request:

```
[martin@osg-test1 ~]$ wsrfl-get-properties -e myJob.epr \  
  "{http://www.globus.org/namespaces/2008/03/gram/job/types}localUserId" \  
  "{http://www.globus.org/namespaces/2008/03/gram/job/exec}localJobId" \  
  
<ns1:localUserId xmlns:ns1="http://www.globus.org/namespaces/2008/03/gram/job/types">  
  feller  
</ns1:localUserId>  
<ns2:localJobId xmlns:ns2="http://www.globus.org/namespaces/2008/03/gram/job/exec">  
  51370.osg-test1.unl.edu  
</ns2:localJobId>
```

Get the whole resource property document of a job:

```
[martin@osg-test1 ~]$ wsrfl-query -e myJob.epr "/*" \  
  
<ns0:managedJobResourceProperties  
  xmlns:ns0="http://www.globus.org/namespaces/2008/03/gram/job"  
  xmlns:ns04="http://www.w3.org/2001/XMLSchema-instance"  
  xmlns:ns05="http://www.w3.org/2001/XMLSchema"  
  xmlns:ns1="http://www.globus.org/namespaces/2008/03/gram/job/exec"  
  xmlns:ns10="http://www.globus.org/namespaces/2008/03/gram/job/description"  
  xmlns:ns11="http://www.w3.org/2005/08/addressing"  
  xmlns:ns14="http://docs.oasis-open.org/wsn/b-2"  
  xmlns:ns2="http://www.globus.org/namespaces/2008/03/gram/job/types"  
  xmlns:ns6="http://docs.oasis-open.org/wsrfl-2"  
  xmlns:ns8="http://www.globus.org/namespaces/2008/04/rendezvous">  
<ns1:localJobId>51370.osg-test1.unl.edu</ns1:localJobId>  
<ns2:exitCode>0</ns2:exitCode>  
<ns3:localUserId xmlns:ns3="http://www.globus.org/namespaces/2008/03/gram/job/types">  
  feller  
</ns3:localUserId>  
<ns4:userSubject xmlns:ns4="http://www.globus.org/namespaces/2008/03/gram/job/types">  
  /DC=org/DC=doegrids/OU=People/CN=John Doe 807394  
</ns4:userSubject>  
<ns5:holding xmlns:ns5="http://www.globus.org/namespaces/2008/03/gram/job/types">  
  false  
</ns5:holding>  
<ns6:TerminationTime>3008-05-15T21:12:56.067Z</ns6:TerminationTime>  
<ns7:state xmlns:ns7="http://www.globus.org/namespaces/2008/03/gram/job/types">  
  Active  
</ns7:state>  
<ns8:Capacity>1</ns8:Capacity>  
<ns9:CurrentTime xmlns:ns9="http://docs.oasis-open.org/wsrfl-2">  
  2008-05-15T21:17:03.849Z  
</ns9:CurrentTime>  
<ns10:serviceLevelAgreement>  
  <ns10:job>  
    <ns10:factoryEndpoint>  
      <ns11:Address>  
        https://osg-test1.unl.edu:8443/wsrfl/services/ManagedJobFactoryService  
      </ns11:Address>  
      <ns12:ReferenceParameters xmlns:ns12="http://www.w3.org/2005/08/addressing">
```

```
<ns5:ResourceID ns04:type="ns05:string"
  xmlns:ns5="http://www.globus.org/namespaces/2008/03/gram/job">
  PBS
</ns5:ResourceID>
</ns12:ReferenceParameters>
</ns10:factoryEndpoint>
<ns10:executable>/bin/sleep</ns10:executable>
<ns10:directory>/home/john</ns10:directory>
<ns10:argument xmlns:xsd="http://www.w3.org/2001/XMLSchema"
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xsi:type="xsd:string">
  600
</ns10:argument>
<ns10:stdout>/dev/null</ns10:stdout>
<ns10:stderr>/dev/null</ns10:stderr>
<ns10:count>1</ns10:count>
</ns10:job>
</ns10:serviceLevelAgreement>
<ns13:RendezvousCompleted
  xmlns:ns13="http://www.globus.org/namespaces/2008/04/rendezvous">
  false
</ns13:RendezvousCompleted>
<ns14:FixedTopicSet>false</ns14:FixedTopicSet>
<ns16:TopicSet Dialect="http://docs.oasis-open.org/wsn/t-1/TopicExpression/Simple"
  xmlns:ns15="http://www.globus.org/namespaces/2008/04/rendezvous"
  xmlns:ns16="http://docs.oasis-open.org/wsn/b-2">
  ns15:RendezvousCompleted
</ns16:TopicSet>
<ns18:TopicSet Dialect="http://docs.oasis-open.org/wsn/t-1/TopicExpression/Simple"
  xmlns:ns17="http://www.globus.org/namespaces/2008/03/gram/job"
  xmlns:ns18="http://docs.oasis-open.org/wsn/b-2">
  ns17:stateChangeInformation
</ns18:TopicSet>
<ns19:TopicExpressionDialect xmlns:ns19="http://docs.oasis-open.org/wsn/b-2">
  http://docs.oasis-open.org/wsn/t-1/TopicExpression/Simple
</ns19:TopicExpressionDialect>
</ns0:managedJobResourceProperties>
```

The output might not be so nicely formatted like in the above examples.

For more information about these commands check the [Java WS Core's User's Guide](#).

2. List of all jobs in a GRAM4 instance

There is currently no way to get a list of all jobs managed by a GRAM4 instance or to get a list of all jobs of a particular user managed by GRAM4. A user or client has to keep track of all jobs it submitted if this is of interest.

However, if audit logging is configured in GRAM4 (see [Audit Logging](#) in the [System Administrator's Guide](#)), a user could request a list of all jobs could be requested from an administrator. But this information is probably by default not available on the fly.

Chapter 5. Terminating jobs

Terminating a job using globusrun-ws means interrupting job processing and destroying all job-related data on the server-side, including delegated credentials if globusrun-ws did the delegation itself.

If the job is still running and not already fully processed, termination will cause the job to go through a series cleanup steps in GRAM4 before the job-related data is destroyed. The cleanup steps being performed depend on the job and the state it is in. In general this includes cancellation of a running job at the local resource manager and running fileCleanUp if so specified in the job description. Termination at the local resource manager however will only be performed if the job did not already finished executing. This also applies to fileCleanUp: If no fileCleanUp is specified in the job description or if the job already passed fileCleanUp when the termination request comes in, then this step is skipped.

Depending on load in GRAM4 performing these cleanup steps may take a while.

1. Jobs in batch mode

The following shows how to terminate a job that had been submitted in batch mode, assuming the EPR of the job is stored in the file myJob.epr:

```
[martin@osg-test1 ~]$ globusrun-ws -kill -j myJob.epr
Current job state: Active
Current job state: UserTerminateDone
Requesting original job description...Done.
Destroying job...Done.
```

2. Jobs in interactive mode

If globusrun-ws is run in interactive mode `Ctrl-C` will cause job termination.

```
[martin@osg-test1 ~]$ globusrun-ws -submit -c /bin/sleep 30
Submitting job...Done.
Job ID: uuid:56b15176-22d4-11dd-8bdd-0013d4c3b957
Termination time: 05/15/3008 23:12 GMT
Current job state: Active
Canceling...Current job state: UserTerminateDone
Canceled.
Destroying job...Done.
```

Repetitive `Ctrl-C` cause globusrun-ws to return and not wait for success of termination.

Chapter 6. Job lifetime

For a general introduction see section [Job Lifetime](#) in the GRAM4 approach.

Jobs submitted to WS-GRAM have a lifetime. If the lifetime of a ManagedJob resource expires the job will be terminated and finally the job resource will be destroyed and the job's persistence data will be removed.

For executable jobs the user-relevant steps in termination are:

- Cancellation of the job at the local resource manager if it's still running.
- Performing fileCleanUp if specified in the job description and the job did not already pass this step.

If a multi job expires all sub-jobs will be terminated.

1. Server-side settings that impact job lifetime

There are 2 resource properties (RP's) of GRAM4's factory service that have impact on lifetime of job resources:

maxJobLifetime	Max lifetime a client can specify in the initial job submission and in subsequent setTerminationTime calls. Default value is 1 year. A negative value means that there is no limit.
jobTTLAfterProcessing	Amount of time a job resource keeps on existing after the job has been fully processed and is in a final state Done, Failed, UserTerminateDone, UserTerminateFailed, and the client did not specify a job lifetime. Default value is 24h. A negative value means that the job resource does not expire.

Values are specified in seconds. A client can query the RP's to find out their values like explained in the following examples.

Getting the value of the RP maxLifetime:

```
[martin@osg-test1 ~]$ globus-wsrf-get-property \  
-s https://osg-test1.unl.edu:8443/wsrf/services/ManagedJobFactoryService \  
{http://www.globus.org/namespaces/2008/03/gram/job}maxJobLifetime
```

The result (in this case 1 year) is:

```
<ns1:maxJobLifetime xmlns:ns1="http://www.globus.org/namespaces/2008/03/gram/job">  
31536000  
</ns1:maxJobLifetime>
```

Getting the value of the RP jobTTLAfterProcessing:

```
[martin@osg-test1 ~]$ globus-wsrf-get-property \  
-s https://osg-test1.unl.edu:8443/wsrf/services/ManagedJobFactoryService \  
{http://www.globus.org/namespaces/2008/03/gram/job}jobTTLAfterProcessing
```

The result (in this case 24h) is:

```
<ns1:jobTTLAfterProcessing xmlns:ns1="http://www.globus.org/namespaces/2008/03/gram/job">  
86400  
</ns1:jobTTLAfterProcessing>
```

2. Client-side information

This section explains how the above parameters impact a client and what the actual lifetime of a job is, when a user does not specify a lifetime at all or when he/she specifies a lifetime for a job.

2.1. Specifying no lifetime in submission

The job does not expire until it is fully processed. After that the lifetime will be set to (now + jobTTLAfterProcessing). By this it is guaranteed that a job runs to completion (including fileStageOut and fileCleanUp) and a client has the ability to query the status of a job for a while before it will be removed.

The default C-client globusrun-ws by default does not set a lifetime

2.2. Specifying a lifetime in submission

The job will definitely be terminated when the lifetime expires regardless of the status of the job. A client can however extend the lifetime before the lifetime expires (restricted by maxJobLifetime if set > -1 by the admin). If a client specifies a termination time in the past or a termination time that exceeds maxJobLifetime an UnableToSetTerminationTimeFault is thrown by MJFS.createManagedJob()

Using the C-client globusrun-ws you can set a lifetime for a job in 2 ways. The first example shows how to set a relative lifetime, i.e. the job will expire in 48h from now:

```
globusrun-ws -submit -term "+48:00" -b -o myJob.epr -f myJob.xml
```

The second example shows how to set an absolute lifetime. The job will expire at the given date:

```
globusrun-ws -submit -term "10/23/2008 12:00" -b -o myJob.epr -f myJob.xml
```

In both example the job had been submitted in batch mode, which makes sense for longer running jobs. For more information about globusrun-ws see [here](#).

2.3. Setting a new lifetime on an existing job

In case a requested new termination time conflicts with the maxJobLifetime setting provided by an admin a TerminationTimeRejectedException is thrown. The following example shows how to set a new termination time of a job resource (assuming that the Endpoint Reference (EPR) of the job is stored in the file myJob.epr). The new lifetime is provided in seconds (604800 in this example [one week]):

```
[martin@osg-test1 ~]$ wsrp-set-termination-time -e myJob.epr 604800
```

The output could be something like this:

```
requested: Tue May 13 09:27:15 CDT 2008
scheduled: Tue May 13 09:27:15 CDT 2008
```

Chapter 7. Job hold and release

It is possible to specify in a job description that the job be put on hold when it reaches a chosen state (see [GRAM Approach](#) documentation for more information about the executable job state machine, and see the [job description XML schema documentation](#)¹ for information about how to specify a held state). This is useful for instance when a GRAM client wishes to directly access output files written by the job (as opposed to waiting for the stage-out step to transfer files from the job host). The client would request that the file cleanup process be held until released, giving the client an opportunity to fetch all remaining/buffered data after the job completes but *before* the output files are deleted.

This is used by `globusrun-ws` in order to ensure client-side streaming of remote files in batch mode.

Valid hold states are:

- StageIn
- StageOut
- CleanUp
- Pending

The following job description (later referred to by `job.xml`) shows how to specify a hold state in the job description:

```
<job>
  <holdState>CleanUp</holdState>
  <executable>/bin/date</executable>
  <stdout>${GLOBUS_USER_HOME}/stdout</stdout>
  <stderr>${GLOBUS_USER_HOME}/stderr</stderr>
  <fileCleanUp>
    <deletion>
      <file>gsiftp://osg-test1.unl.edu:2811/${GLOBUS_USER_HOME}/stdout</file>
    </deletion>
    <deletion>
      <file>gsiftp://osg-test1.unl.edu:2811/${GLOBUS_USER_HOME}/stderr</file>
    </deletion>
  </fileCleanUp>
</job>
```

Submitting the job in batch mode:

```
[martin@osg-test1 tmp]$ globusrun-ws -submit -S -f myJob.xml -b -o myJob.epr
Delegating user credentials...Done.
Submitting job...Done.
Job ID: uuid:837941d4-1085-11dd-b401-0013d4c3b957
Termination time: 04/22/3008 16:02 GMT
[martin@osg-test1 tmp]$
```

Checking for status:

¹ [../schemas/mj_types.html#element_holdState](#)

```
[martin@osg-test1 tmp]$ globusrun-ws -status -j myJob.epr
Current job state: CleanUp-Hold
[martin@osg-test1 tmp]$
```

Releasing the job:

```
globusrun-ws -release -j myJob.epr
```

Checking for status after the release:

```
[martin@osg-test1 tmp]$ globusrun-ws -status -j myJob.epr
Current job state: Done
```

Removing the job:

```
[martin@osg-test1 ~]$ globusrun-ws -kill -j myJob.epr
Requesting original job description...Done.
Destroying job...Done.
```

Chapter 8. Client-Side Generated Submission ID

A submission ID may be used in the GRAM protocol for reliability in the face of message faults or other transient errors in order to ensure that at most one instance of a job is executed, i.e. to prevent accidental duplication of jobs under rare circumstances with client retry on failure. By default, the *globusrun-ws* program will generate a submission ID (*uuid*). One can override this behavior by supplying a submission ID as a command line argument.

If a user is unsure whether a job was submitted successfully, he should resubmit using the same ID as was used for the previous attempt. If the job had already been accepted by the container in the first submission no new job is started but the Endpoint Reference of the first job is returned back to the client.

Note that the client-generated submission ID is *not* the ID GRAM4 uses for the job on the server-side. GRAM4 internally generates it's own UUID for the job to circumvent the risk of potentially problematic client-side submission IDs.

The client-generated submission ID shows up in all server-side logs (container-log, CEDPS-Troubleshooting admin log, records in the audit database) and is linked with the server-side job UUID. This enables debugging in situations where the user only knows about the client-side generated submission ID.

Chapter 9. Specifying SoftEnv keys in the job description

For a short introduction to SoftEnv please have a look at the [SoftEnv chapter](#).

If SoftEnv is enabled on the server-side, nothing needs to be added to a job description to set up the environment which is specified in the `.soft` file in the remote home directory of the user before the job is submitted to the scheduler.

If a different software environment should be used than the one specified in the remote `.soft` file, the user must provide SoftEnv parameters in the extensions element of the job description.

The schema of the extension element for software selection in the job description is as follows:

```
<element name="softenv" type="xsd:string">
```

For example, to add the SoftEnv commands `@teragrid-basic`, `+intel-compilers`, `+atlas`, and `+tgcp` to the job process' environment, the user would specify the following `<extensions>` element in the job description:

```
<extensions>
  <softenv>@teragrid-basic</softenv>
  <softenv>+intel-compilers</softenv>
  <softenv>+atlas</softenv>
  <softenv>+tgcp</softenv>
</extensions>
```

So far there is no way for a user to learn from the remote service itself whether or not SoftEnv support is enabled. Currently, the only way to check this is to submit a job with `/bin/env` as the executable and watch the results.

The following table describes what happens in various scenarios if SoftEnv is disabled or enabled on the server side:

	Disabled on server side	Enabled on server side
User provides no SoftEnv extensions:	No SoftEnv environment is configured before job submission, even if the user has a <code>.soft</code> file in their remote home directory.	If the user has a <code>.soft</code> file (and no <code>.nosoft</code> file) in their remote home directory, then the environment defined in the <code>.soft</code> file will be configured before job submission. If the user has a <code>.nosoft</code> file in their remote home directory, no environment will be prepared.
User provides valid SoftEnv extensions:	If SoftEnv is not installed on the server then no environment will be configured If SoftEnv is installed, the environment the user specifies in the <code><extensions></code> elements overwrites any SoftEnv configuration the user specifies in a <code>.soft</code> or <code>.nosoft</code> file in their remote home directory. The environment will be configured as specified by the user in the <code><extensions></code> elements before job submission.	The specified environment overwrites any SoftEnv configuration the user specifies in a <code>.soft</code> or a <code>.nosoft</code> file in their remote home directory. The environment will be configured as specified by the user in the <code><extensions></code> elements before job submission.

	Disabled on server side	Enabled on server side
User provides invalid SoftEnv extensions:	<p>If SoftEnv is not installed on the server, then no environment will be configured.</p> <p>If SoftEnv is installed, the environment the user specifies in the <extensions> elements overwrites any SoftEnv configuration the user specifies in a .soft or a .nosoft file in their remote home directory. Only the valid keys in the SoftEnv <extensions> elements will be configured. If no valid key is found, no environment will be configured. SoftEnv warnings are logged to the stdout of the job.</p>	<p>The specified environment overwrites any SoftEnv configuration the user specifies in a .soft or a .nosoft file in their remote home directory. Only the valid keys in the SoftEnv <extensions> elements will be configured. If no valid key is found, no environment will be configured. SoftEnv warnings are logged to stdout of the job.</p>
	<p>In general, jobs do not fail if they have SoftEnv extensions in their description and SoftEnv is disabled (or not even installed) on the server side. But they will fail if they rely on environments being set up before job submission.</p>	

 **Note**

In the current implementation, it is not possible to call executables directly whose paths are defined in SoftEnv without specifying the complete path to the executable.

For example, if a database query must be executed using the **mysql** command and **mysql** is not in the default path, then the direct use of **mysql** as an executable in the jobs description document will fail, even if the use of SoftEnv is configured. The **mysql** command must be written to a script which is in the default path.

Thus a job submission with the following job description document will fail:

```
<job>
...
<executable>mysql</executable>
...
</job>
```

But when the command is embedded inside a shell script which is specified as the executable in the job description document, it will work:

```
#!/bin/sh
...
mysql ...
...
```

 **Note**

The use of invalid SoftEnv keys in the extension part of the job description document does not generate errors.

Chapter 10. Job and process rendezvous

GRAM4 services implement a WS Rendezvous¹ mechanism to perform synchronization between job processes in a multiprocess job and between subjobs in a multijob. The job application can in fact register binary information, for instance process information or subjob information, and get notified when all the other processes or subjobs have registered their own information. This is for instance useful for parallel jobs which need to rendezvous at a "barrier" before proceeding with computations, in the case when no native application API is available to help do the rendezvous.

¹ ../../wsrendezvous/

GRAM4 Commands

Name

globusrun-ws -- Official job submission client for GRAM4

```
globusrun-ws -submit [-batch] [-quiet] [-no-cleanup] [-streaming] [-streaming-out filename] [-streaming-err filename] [-host-authz] [-self-authz] [-subject-authz subject name] [-private] [-http-timeout milliseconds] [-debug] [-allow-ipv6] [-passive] [-nodcau] [[-factory-epr-file filename] [[-factory contact] | [-factory-type type]]] [[-submission-id uuid] | [-submission-id-file filename]] [-submission-id-output-file filename] [-job-epr-output-file filename] [-job-delegate] [-staging-delegate] [-job-credential-file filename] [-staging-credential-file filename] [-transfer-credential-file filename] [-termination [+HH:MMmm/dd/yyyy HH:MM] ] [[-job-description-file filename] | [-job-command [--] program arg ...]]
globusrun-ws -validate -job-description-file filename
globusrun-ws -monitor -job-epr-file filename [-quiet] [-no-cleanup] [-streaming] [-streaming-out filename] [-streaming-err filename] [-host-authz] [-self-authz] [-subject-authz subject name] [-private] [-http-timeout milliseconds] [-debug] [-allow-ipv6] [-passive] [-nodcau]
globusrun-ws -status -job-epr-file filename [-host-authz] [-self-authz] [-subject-authz subject name] [-private] [-http-timeout milliseconds] [-debug]
globusrun-ws -kill -job-epr-file filename [-host-authz] [-self-authz] [-subject-authz subject name] [-private] [-http-timeout milliseconds] [-debug]
globusrun-ws -help
globusrun-ws -usage [-submit] [-validate] [-monitor] [-status] [-kill]
globusrun-ws -version(s)
```

Description

globusrun-ws (GRAM4 client) is a program for submitting and managing jobs to a local or remote job host. GRAM4 provides secure job submission to many types of *job scheduler* for users who have the right to access a job hosting resource in a Grid environment. All GRAM4 submission options are supported transparently through the embedded request document input. globusrun-ws offers additional features to fetch job output files incrementally during the run as well as to automatically delegate credentials needed for certain optional GRAM4 features. Online and batch submission modes are supported with reattachment (recovery) for jobs whether they were started with this client or another GRAM4 client application.

Command options

Quiet mode

A variety of protocol status messages, warning messages, and output data may be printed to standard output and error under multiple command modes. The *quiet mode* suppresses all but fatal standard error messages in order to have clean outputs for use in scripting or with the *streaming output mode* where application output is retrieved and output.

-q, -quiet If supplied, all non-fatal status and protocol-related messages are suppressed.

Debug mode

-dbg, -debug If supplied, all soap messages and ftp control messages will be displayed on stderr.

Protocol Options

Service authorization

Usually, secure communication includes mutual authentication. In addition to the service authorizing the client for the requested operation(s), an authorization decision is made by the client to determine whether the remote service is the one intended.

- host, -host-authz The GSI "host authorization" rule is used to verify that the service is using a host credential appropriate for the underlying service address information. This is the default.

- self, -self-authz The GSI "self authorization" rule is used to verify that the service is using a (proxy) credential derived from the same identity as the client's.

- subject, -subject-authz subject name The service must be using a credential with the exact subject name provided by this option.

Security Protocol

The client uses secure transport for all https endpoints and secure message for http. Secure conversation is currently unsupported.

- p, -private If supplied, privacy-protection is enabled between globusrun-ws and GRAM4 or GridFTP services. It is a fatal error to select privacy protection if it is not available due to build options or other security settings. **Note:** Currently only supported with https endpoints.

Timeouts

- T, -http-timeout milli-seconds Set timeout for HTTP socket, in milliseconds, for all Web services interactions. The default value is 120000 (2 minutes).

Signal handling

- n, -no-cleanup If supplied, the default behavior of trapping interrupts (SIG_INTR) and cancelling the job is disabled. Instead, the interrupt simply causes the tool to exit without affecting the ManagedJob resource.

Submit options

- submit The -submit command submits (or *resubmits*) a job to a job host using an XML-based job description document. The -submit command can submit jobs in one of three output modes: batch, interactive, or interactive-streaming.

Output Mode

The user can select several tool behaviors following submission. In *batch mode*, the tool prints the resulting ManagedJob EPR as the sole standard output (unless in *quiet mode*) and exits. In *interactive mode*, the tool keeps running in order to monitor job status. Interactive mode is qualitatively equivalent to a batch-mode submission immediately followed a second invocation of globusrun-ws using the -monitor command. In interactive mode, an optional *streaming mode* where job output files are fetched and output from globusrun-ws.

- b, -batch If supplied, the batch mode is enabled. The default is interactive mode. The tool prints the resulting ManagedJob EPR as the sole standard output (unless in quiet mode) and exits.
- s, -streaming The standard output and standard error files of the job are monitored and data is written to the corresponding output of globusrun-ws. The standard output will contain ONLY job output data, while the standard error may be a mixture of job error output as well as globusrun-ws messages, unless the *quiet mode* is also enabled.
- Streaming output depends on the ability to access job outputs via GridFTP. If -streaming mode is selected and the *job description* does not already specify output file redirection for the job host, then globusrun-ws adds unique output file name redirections and automatic cleanup directives to the job description.
- If you are using -batch mode, but intend to use -streaming with -monitor, you may want to still include -streaming. -streaming always introduces a 'CleanUp Hold' state which ensures that all the data is streamed before the files are destroyed. If you do use -streaming with -batch, you **must** come back with -monitor so the hold can be released.
- This option implies -staging-delegate if the stdout and stderr entries are not specified in the job description.
- so, -stdout-file filename append stdout out stream to the specified file instead of to stdout.
- se, -stderr-file filename append stderr out stream to the specified file instead of to stderr.

Streaming Options

Streaming makes use of GridFTP client calls to retrieve user data. The following options apply to such transfers.

- ipv6, -allow-ipv6 Allow streaming transfers to use IPV6.
- passive Force streaming transfers to use MODE S to allow for passive mode transfers. (Useful if you're behind a firewall, but expensive because there is no connection caching).
- nodcau Disable data channel authentication on streaming transfers

Factory information

Addressing information for the ManagedJobFactory target of this submission must be provided. If neither option is specified, and no EPR is supplied in the job description, then "-factory localhost -factory-type fork" is assumed.

- Ff, -factory-epr-file filename If supplied, this option causes the EPR for the ManagedJobFactory to be read from the given file. This EPR is used as the service endpoint for submission of the job.
- F, -factory contact If supplied, this option causes an EPR to be constructed using ad-hoc methods that depend on GT implementation details. For interoperability to other implementations of GRAM4_, the -factory-epr-file option should be used instead.

[protocol://][hostname|hostaddr][:port][/service]

Default values form the following contact information if not overridden:

https://localhost:8443/wsrp/services/ManagedJobFactoryService

-Ft, -factory-type type In the absence of `-factory-epr-file`, this option refines the behavior of the `-factory` option to select a specific type of scheduler. The default is "Fork" for single jobs and "Multi" for *multijobs*.

Job description

A description of the job to be submitted must be provided with the `-submit` command, either using the GRAM4 XML description syntax or a simpler Unix command and argument list.

-f, -job-description-file filename If supplied, this option causes the job description to be read from the given file. This description is modified according to the other options and passed in the GRAM4 submission messages. The root element of this file must be 'job' for a single job or 'multiJob' for a multijob.

-c, -job-command [--] prog [arg ...] If supplied, this option take all remaining globusrun-ws arguments as its arguments; therefore it must appear last among globusrun-ws options. This option causes globusrun-ws to generate a simple job description with the named program and arguments.

Submission ID

A submission ID may be used in the GRAM4 protocol for robust reliability in the face of message faults or other transient errors to ensure that at most one instance of a job is executed, i.e. to prevent accidental duplication of jobs under rare circumstances with client retry on failure. The globusrun-ws tool always uses this feature, requiring either a submission ID to be passed in as input or a new unique ID to be created by the tool itself. If a new ID is created, it should be captured by the user who wishes to exploit this reliability interface. The ID in use, whether created or passed as input, will be written to the optional output file when provided, as well as to the standard error output unless the *quiet mode* is in effect.

If a user is unsure whether a job was submitted successfully, he should resubmit using the same ID as was used for the previous attempt.

-I, -submission-id ID If supplied, this option causes the job to be submitted using the given ID in the reliability protocol.

-If, -submission-id-file filename If supplied, this option causes the ID to be read from the given file. It is an error to use both mechanisms to provide an input ID.

-Io, -submission-id-output-file file-name If supplied, the ID in use is written to the given file, whether this ID was provided by the user or given by one of the above input options.

Job EPR output

A successful submission will create a new ManagedJob resource with its own unique EPR for messaging. The globusrun-ws tool will output this EPR to a file when requested and as the sole standard output when running in batch mode. When running in streaming output mode, it is possible that the EPR will not be output and the user's only recourse is to submit again with the same submission ID and job request in order to reattach to the existing job.

-o, -job-epr-output-file filename If supplied, the created ManagedJob EPR will be written to the given file following successful submission. The file will not be written if the submission fails.

Delegation

The job description supports the optional identification of delegated credentials for use by the GRAM4 services. These features are passed through globusrun-ws without modification. However, globusrun-ws can also perform delegation

and construct these optional request elements before submitting it to the service. The only delegation performed by default (if an endpoint does not already exist) is the multijob level jobCredential.

- J, -job-delegate If supplied AND the job description does not already provide a jobCredential element, globusrun-ws will delegate the client credential to GRAM4 and introduce the corresponding element to the submission input.
- S, -staging-delegate If supplied AND the job description does include staging or cleanup directives AND the job description does not already provide the necessary stagingCredential or transferCredential element(s), globusrun-ws will delegate the client credential to GRAM4 and RFT, and introduce the corresponding elements to the submission input.

This option is implied by -streaming
- Jf, -job-credential-file filename: If supplied AND the job description does not already provide a jobCredential element, globusrun-ws will copy the supplied epr into the job description. This should be an epr returned from the DelegationFactoryService intended for use by the job (or, in the case of a multijob, for authenticating to the subjobs).

note: for multijob descriptions, only the top level jobCredential will be copied into.
- Sf, -staging-credential-file filename: If supplied AND the job description does not already provide a stagingCredential element, globusrun-ws will copy the supplied epr into the job description. This should be an epr returned from the DelegationFactoryService intended for use with the RFT service associated with the ManagedJobService.

note: this option is ignored for multijobs.
- Tf, -transfer-credential-file filename: If supplied, globusrun-ws will copy the epr into each of the stage in, stage out, and cleanup elements that do not already contain a transferCredential element. This should be an epr returned from the DelegationFactoryService intended for use by RFT to authenticate with the target gridftp server.

note: this option is ignored for multijobs.

Lifetime

The ManagedJob resource supports lifetime management in the form of a scheduled destruction. The default lifetime requested by the client is infinite, subject to server policies.

- term, -termination mm/dd/yyyy Set an absolute termination time.
HH:MM
- term, -termination +HH:MM Set a termination time relative to the successful creation of the job. The default is +24:00

Validate options

- validate The -validate command checks the job description for syntax errors and a subset of semantic errors without making any service requests.

Job description

-f, -job-description-file filename This option causes the job description to be read from the given file. This description is checked for validity.

Monitor options

-monitor The -monitor command attaches to an existing job in interactive or interactive-streaming output modes.

Job

Addressing information for the ManagedJob target of this command must be provided.

-j, -job-epr-file filename If supplied, this option causes the EPR for the ManagedJob to be read from the given file. This EPR is used as the endpoint for service requests.

Output mode

In the default *interactive mode*, the tool keeps running in order to monitor job status. In the optional *interactive-streaming mode*, the job output files are fetched and output from globusrun-ws as well.

-s, -streaming See Output mode under Submit Options above for details on streaming.

Status options

-status The -status command reports the current state of the job and exits. See the [External States of the Managed Job Services](#) section of the developer guid for information on valid job states.

See the Job options for the -monitor command.

Kill options

-kill The -kill command requests the immediate cancellation of the job and exits.

Help options

-help Outputs an overview of the commands and features of the command.

Usage options

-usage Outputs brief usage information for the command.

Version options

-version Outputs version information for the command.

Job Handling

For every job that globusrun-ws delegates a credential, globusrun will augment the user's job description, adding annotations that will later tell globusrun-ws to destroy the credential after the job has been destroyed. Below are 2 job annotation examples. globusrun-ws only delegated the job cred...

```
<extensions>
<globusrunAnnotation>
<automaticJobDelegation>true</automaticJobDelegation>
<automaticStagingDelegation>>false</automaticStagingDelegation>
<automaticStageInDelegation>>false</automaticStageInDelegation>
<automaticStageOutDelegation>>false</automaticStageOutDelegation>
<automaticCleanUpDelegation>>false</automaticCleanUpDelegation>
</globusrunAnnotation>
</extensions>
```

globusrun-ws delegated the job, staging and stage in cred...

```
<extensions>
<globusrunAnnotation>
<automaticJobDelegation>true</automaticJobDelegation>
<automaticStagingDelegation>true</automaticStagingDelegation>
<automaticStageInDelegation>true</automaticStageInDelegation>
<automaticStageOutDelegation>>false</automaticStageOutDelegation>
<automaticCleanUpDelegation>>false</automaticCleanUpDelegation>
</globusrunAnnotation>
</extensions>
```

Environment

X509_USER_PROXY Overrides the default selection of user credentials when using GSI security.

Exit Codes

The client returns negative error codes for client errors, 0 for success, and positive error codes from the submitted job (where possible)

Chapter 11. Troubleshooting

For a list of common errors in GT, see [Error Codes](#).

For information about sys admin logging, see [Chapter 10, Admin Debugging](#) in the GRAM4 Admin Guide.

1. Troubleshooting tips

In case you run into problems you can do the following

- Check the GRAM4 documentation. Maybe you'll find hints here to solve your problem.
- Send e-mails to one of several Globus e-mail lists. You'll have to subscribe to a list before you can send an e-mail to it. See [here](#)¹ for general e-mail lists and information on how to subscribe to a list and [here](#)² for GRAM specific lists.

Probably the best lists for GRAM4-related problems are gt-user@globus.org and gram-user@globus.org

- Check the container log for errors.

In case you don't find anything suspicious you can increase the log-level of GRAM4 or other relevant components. Maybe the additional logging-information will tell you what's going wrong. General information about container logging can be found [Logging in Java WS Core](#) section.

To get debug information from GRAM4, un-comment the following line in `$GLOBUS_LOCATION/container-log4j.properties` by removing the leading '#' and restart the GT4 server.

```
# log4j.category.org.globus.exec=DEBUG
```

The logging output can either be found on the console if you started the container using `globus-start-container` (maybe with arguments) or in `$GLOBUS_LOCATION/var/container.log` in if you started the container using the command `globus-start-container-detached`

¹ http://dev.globus.org/wiki/Mailing_Lists

² http://dev.globus.org/wiki/GRAM#Mailing_Lists

2. Java WS Core Errors

Table 11.1. Java WS Core Errors

Error Code	Definition
Failed to acquire notification consumer home instance from registry	Caused by <code>javax.naming.NameNotFoundException</code> : Name <code>services</code> is not bound in
The WS-Addressing 'To' request header is missing	This warning is logged by the container if the request did not contain the necessary <i>WS-Addressing</i> headers, those headers at all or is somehow misconfigured.
java.io.IOException: Token length X > 33554432	If you see this error in the container log, it usually means you are trying to connect to HTTPS server using <code>https</code> specifies 8443 as a port number and <code>http</code> as the protocol name.
java.lang.NoSuchFieldError: DOCUMENT	This error usually indicates a mismatch between the version of Apache Axis that the code was compiled with and the version currently running with.
org.globus.wsrfl.InvalidResourceKeyException: Argument key is null / Resource key is missing	These errors usually indicate that a resource key was not passed with the request or that an invalid resource key was used (the element QName of the resource key did not match what the service expected).
Unable to connect to localhost:xxx	Cannot resolve localhost. The machine's <code>/etc/hosts</code> isn't set up correctly and/or you do not have DNS for
org.globus.common.ChainedIOException: Failed to initialize security context	This may indicate that the user's proxy is invalid.
Error: org.xml.sax.SAXException: Unregistered type: class xxx	This may indicate that an Axis generated XML type, defined by the WS RLS XSD, was not properly registered upon deployment without intervention by the user, sometimes they do not.
No socket factory for 'https' protocol	<p>When a client fails with the following exception:</p> <pre> java.io.IOException: No socket factory for 'https' protocol at org.apache.axis.transport.http.HTTPSender.getSocket(HTTPSender.java:100) org.apache.axis.transport.http.HTTPSender.writeToSocket(HTTPSender.java:110) org.apache.axis.transport.http.HTTPSender.invoke(HTTPSender.java:120) </pre> <p>FIXME - it may have happened because...</p>

Error Code	Definition
No client transport named 'https' found	<p>When a client fails with the following exception:</p> <pre>No client transport named 'https' found at org.apache.axis.client.AxisClient.invoke(AxisClient.java:170) at org.apache.axis.client.Call.invokeEngine(Call.java:2726)</pre> <p>The client is most likely loading an incorrect <code>client-config.wsdd</code> configuration file.</p>
ConcurrentModificationException in Tomcat 5.0.x	<p>If the following exception is visible in the Tomcat logs at startup, it might cause the HTTPSValve to fail:</p> <pre>java.util.ConcurrentModificationException at java.util.HashMap\$HashIterator.nextEntry(HashMap.java:782) at java.util.HashMap\$EntryIterator.next(HashMap.java:824) at java.util.HashMap.putAllForCreate(HashMap.java:424) at java.util.HashMap.clone(HashMap.java:656) at mx4j.server.DefaultMBeanRepository.clone(DefaultMBeanRepository.</pre> <p>The HTTPSValve might fail with the following exception:</p> <pre>java.lang.NullPointerException at org.apache.coyote.tomcat5.CoyoteRequest.setAttribute(CoyoteRequestFacade.java:100) at org.apache.coyote.tomcat5.CoyoteRequestFacade.setAttribute(CoyoteRequestFacade.java:100) at org.globus.tomcat.coyote.valves.HTTPSValve.expose(HTTPSVAlve.java:100)</pre> <p>These exceptions will prevent the transport security from working properly in Tomcat.</p>
java.net.SocketException: Invalid argument or cannot assign requested address	<p>FIXME - what causes this?</p>
GAR deploy/undeploy fails with container is running error	<p>A GAR file can only be deployed or undeployed locally while the container is off. However, GAR deployment fails with this error even if the container is off. This usually happens if the container has crashed or was stopped from cleaning up its state files.</p>

3. Errors

Table 11.2. GRAM4 Errors

Error Code	Definition	Possible Solutions
<p>globusrun-ws - error querying job state</p>	<p>During job submission, an error like this occurs: globusrun-ws failed: Delegating user credentials...Done. Submitting job...Done. Job ID: xxxx Termination time: xxxx Current job state: Unsubmitted globusrun-ws: Error querying job state globus_soap_message_module: Failed sending request ManagedJobPortType_GetMultipleResourceProperties. globus_xio: An end of file occurred</p>	<p>Periodically, globusrun-ws will query the GRAM service to check on the job state. The "End of file" indicates that the GRAM server dropped a connection when globusrun-ws tried to read a response. This could be caused by temporary network issues between the client and service, or possibly caused by an overloaded service host.</p>
<p>globusrun-ws - error querying job state</p>	<p>During job submission, an error like this occurs: globusrun-ws failed: Delegating user credentials...Done. Submitting job...Done. Job ID: xxxx Termination time: xxxx Current job state: Unsubmitted globusrun-ws: Error querying job state globus_soap_message_module: Failed sending request ManagedJobPortType_GetMultipleResourceProperties. globus_xio: System error in read: Connection reset by peer globus_xio: A system call failed: Connection reset by peer</p>	<p>Periodically, globusrun-ws will query the GRAM service to check on the job state. The System error in read: Connection reset by peer indicates that the GRAM server dropped the connection while trying to write the response. This could be caused by temporary network issues between the client and service, or possibly caused by an overloaded service host.</p>

Error Code	Definition	Possible Solutions
glo- bus- run- ws - error sub- mit- ting job	During job submission, an error like this occurs: globusrun-ws -Ft PBS -F ht- tps://host.teragrid.org:8444 -submit -b -f /tmp/wsgram.rsl -o /tmp/wsgram.epr failed: Submitting job...Failed. globusrun-ws: Error submitting job globus_soap_message_module: Failed sending request ManagedJobFactoryPortType_createManagedJob. globus_xio: Operation was canceled globus_xio: Operation timed out	The Operation timed out indicates that the GRAM service was not able to accept the job request and respond in time. This could be caused by temporary network issues between the client and service, or possibly caused by an overloaded service host.

Chapter 12. Known Problems in GRAM4

1. Known Problems

The following problems and limitations are known to exist for GRAM4 at the time of the 4.2.1 release:

1.1. Limitations

- [list limitations]

1.2. Outstanding bugs

- [Bug 3384](http://bugzilla.globus.org/bugzilla/show_bug.cgi?id=3384):¹ Inconsistent jobType/count parameter semantics
- [Bug 3529](http://bugzilla.globus.org/bugzilla/show_bug.cgi?id=3529):² setup/postinstall fatal errors should be warnings
- [Bug 3575](http://bugzilla.globus.org/bugzilla/show_bug.cgi?id=3575):³ SEG dependent on GLOBUS_LOCATION env var
- [Bug 3748](http://bugzilla.globus.org/bugzilla/show_bug.cgi?id=3748):⁴ WS-GRAM Pluggable Resource Manager Backend
- [Bug 3803](http://bugzilla.globus.org/bugzilla/show_bug.cgi?id=3803):⁵ Default scratchDirectory doesn't exist
- [Bug 3892](http://bugzilla.globus.org/bugzilla/show_bug.cgi?id=3892):⁶ Out of date performance data?
- [Bug 3948](http://bugzilla.globus.org/bugzilla/show_bug.cgi?id=3948):⁷ Service must release all of its resources on deactivation
- [Bug 4181](http://bugzilla.globus.org/bugzilla/show_bug.cgi?id=4181):⁸ Allow File Staging To/From globusrun-ws application without an external server
- [Bug 4182](http://bugzilla.globus.org/bugzilla/show_bug.cgi?id=4182):⁹ Improve Condor/Fork Job Monitoring for reliability and security
- [Bug 4513](http://bugzilla.globus.org/bugzilla/show_bug.cgi?id=4513):¹⁰ LD_LIBRARY_PATH should not be set if no library_path is specified
- [Bug 4550](http://bugzilla.globus.org/bugzilla/show_bug.cgi?id=4550):¹¹ Multijob code not checking for existence of job credential
- [Bug 4684](http://bugzilla.globus.org/bugzilla/show_bug.cgi?id=4684):¹² Loading persisted jobs with expired delegation resources causes stacktraces
- [Bug 4719](http://bugzilla.globus.org/bugzilla/show_bug.cgi?id=4719):¹³ globus runs /usr/bin/env without checking for \u
- [Bug 4734](http://bugzilla.globus.org/bugzilla/show_bug.cgi?id=4734):¹⁴ Missing wsa:Action for GRAM4 rendezvous register operations

¹ http://bugzilla.globus.org/bugzilla/show_bug.cgi?id=3384

² http://bugzilla.globus.org/bugzilla/show_bug.cgi?id=3529

³ http://bugzilla.globus.org/bugzilla/show_bug.cgi?id=3575

⁴ http://bugzilla.globus.org/bugzilla/show_bug.cgi?id=3748

⁵ http://bugzilla.globus.org/bugzilla/show_bug.cgi?id=3803

⁶ http://bugzilla.globus.org/bugzilla/show_bug.cgi?id=3892

⁷ http://bugzilla.globus.org/bugzilla/show_bug.cgi?id=3948

⁸ http://bugzilla.globus.org/bugzilla/show_bug.cgi?id=4181

⁹ http://bugzilla.globus.org/bugzilla/show_bug.cgi?id=4182

¹⁰ http://bugzilla.globus.org/bugzilla/show_bug.cgi?id=4513

¹¹ http://bugzilla.globus.org/bugzilla/show_bug.cgi?id=4550

¹² http://bugzilla.globus.org/bugzilla/show_bug.cgi?id=4684

¹³ http://bugzilla.globus.org/bugzilla/show_bug.cgi?id=4719

¹⁴ http://bugzilla.globus.org/bugzilla/show_bug.cgi?id=4734

- [Bug 4761](#).¹⁵ Scheduler Tutorial is missing WS-GRAM setup package
- [Bug 4778](#).¹⁶ WS-Fork job manager doesn't set environment up for mpi jobs
- [Bug 4787](#).¹⁷ no lifetime management for WS Rendezvous
- [Bug 4817](#).¹⁸ Condor OS and ARCH do not have dynamic defaults
- [Bug 4864](#).¹⁹ environment variables containing '=' get escaped
- [Bug 4918](#).²⁰ user account details are cached even for unknown users
- [Bug 4944](#).²¹ Multijob resources never yield to memory pressure and can't be destroyed
- [Bug 5012](#).²² Container in livelock state for an incorrectly mapped DN
- [Bug 5017](#).²³ gram[24] tests that need to be updated
- [Bug 5397](#).²⁴ GRAM4 recovery of persisted job resources needs to be reviewed
- [Bug 5402](#).²⁵ "invalid password" error messages
- [Bug 5433](#).²⁶ public interface doc lists non-public/internal APIs
- [Bug 5471](#).²⁷ GRAM Jobs Hang in Unsubmitted State
- [Bug 5484](#).²⁸ Review and Update 4.2 GRAM doc
- [Bug 5515](#).²⁹ Destruction of subscription resources in a container shutdown/restart
- [Bug 5516](#).³⁰ Destruction of subscription resources in a container shutdown/restart
- [Bug 5611](#).³¹ GramJob API changes to improve performace and efficiency
- [Bug 5698](#).³² Allow a prologue/epilogue script for 'mpi' and 'multiple' jobs
- [Bug 5712](#).³³ Gram auditing: local_job_id format variations
- [Bug 5713](#).³⁴ GRAM auditing: Failed database connection loses audit records

¹⁵ http://bugzilla.globus.org/bugzilla/show_bug.cgi?id=4761

¹⁶ http://bugzilla.globus.org/bugzilla/show_bug.cgi?id=4778

¹⁷ http://bugzilla.globus.org/bugzilla/show_bug.cgi?id=4787

¹⁸ http://bugzilla.globus.org/bugzilla/show_bug.cgi?id=4817

¹⁹ http://bugzilla.globus.org/bugzilla/show_bug.cgi?id=4864

²⁰ http://bugzilla.globus.org/bugzilla/show_bug.cgi?id=4918

²¹ http://bugzilla.globus.org/bugzilla/show_bug.cgi?id=4944

²² http://bugzilla.globus.org/bugzilla/show_bug.cgi?id=5012

²³ http://bugzilla.globus.org/bugzilla/show_bug.cgi?id=5017

²⁴ http://bugzilla.globus.org/bugzilla/show_bug.cgi?id=5397

²⁵ http://bugzilla.globus.org/bugzilla/show_bug.cgi?id=5402

²⁶ http://bugzilla.globus.org/bugzilla/show_bug.cgi?id=5433

²⁷ http://bugzilla.globus.org/bugzilla/show_bug.cgi?id=5471

²⁸ http://bugzilla.globus.org/bugzilla/show_bug.cgi?id=5484

²⁹ http://bugzilla.globus.org/bugzilla/show_bug.cgi?id=5515

³⁰ http://bugzilla.globus.org/bugzilla/show_bug.cgi?id=5516

³¹ http://bugzilla.globus.org/bugzilla/show_bug.cgi?id=5611

³² http://bugzilla.globus.org/bugzilla/show_bug.cgi?id=5698

³³ http://bugzilla.globus.org/bugzilla/show_bug.cgi?id=5712

³⁴ http://bugzilla.globus.org/bugzilla/show_bug.cgi?id=5713

- [Bug 5714](#).³⁵ GRAM Auditing: additional data in audit records
- [Bug 5725](#).³⁶ Gram auditing: housekeeping for the auditRecords database
- [Bug 5770](#).³⁷ GRAM4 auditing: inconsistent data in job_description column of DB
- [Bug 5776](#).³⁸ GRAM4 auditing: Need for an INFO log message
- [Bug 5777](#).³⁹ GRAM2 auditing: database connection times out
- [Bug 5778](#).⁴⁰ GRAM2 auditing: no error message on db update failure
- [Bug 5805](#).⁴¹ Change threadpools from Gram custom implementation to java.util.concurrent
- [Bug 5820](#).⁴² Improve Condor Logfile Processing in GRAM
- [Bug 5843](#).⁴³ Swap custom threadpool with ExecutorServices from java.util.concurrent
- [Bug 5853](#).⁴⁴ create automated tests for globus-job-*-ws programs
- [Bug 5859](#).⁴⁵ Java GramJob getExitCode() always returns 0
- [Bug 5969](#).⁴⁶ GRAM Job submission failed!!!
- [Bug 6002](#).⁴⁷ globusrun-ws hangs indefinitely
- [Bug 6019](#).⁴⁸ CEDPS: Add executable path to log statement
- [Bug 6043](#).⁴⁹ Confusing JobID in timeout message
- [Bug 6065](#).⁵⁰ Deleting delegated credential does not work
- [Bug 6069](#).⁵¹ JDD documentation issues
- [Bug 6072](#).⁵² specification of RAM per process in parallel jobs
- [Bug 6091](#).⁵³ GRAM4 JDD substitution detection is too broad
- [Bug 6192](#).⁵⁴ Apply relevant VDT patches

³⁵ http://bugzilla.globus.org/bugzilla/show_bug.cgi?id=5714

³⁶ http://bugzilla.globus.org/bugzilla/show_bug.cgi?id=5725

³⁷ http://bugzilla.globus.org/bugzilla/show_bug.cgi?id=5770

³⁸ http://bugzilla.globus.org/bugzilla/show_bug.cgi?id=5776

³⁹ http://bugzilla.globus.org/bugzilla/show_bug.cgi?id=5777

⁴⁰ http://bugzilla.globus.org/bugzilla/show_bug.cgi?id=5778

⁴¹ http://bugzilla.globus.org/bugzilla/show_bug.cgi?id=5805

⁴² http://bugzilla.globus.org/bugzilla/show_bug.cgi?id=5820

⁴³ http://bugzilla.globus.org/bugzilla/show_bug.cgi?id=5843

⁴⁴ http://bugzilla.globus.org/bugzilla/show_bug.cgi?id=5853

⁴⁵ http://bugzilla.globus.org/bugzilla/show_bug.cgi?id=5859

⁴⁶ http://bugzilla.globus.org/bugzilla/show_bug.cgi?id=5969

⁴⁷ http://bugzilla.globus.org/bugzilla/show_bug.cgi?id=6002

⁴⁸ http://bugzilla.globus.org/bugzilla/show_bug.cgi?id=6019

⁴⁹ http://bugzilla.globus.org/bugzilla/show_bug.cgi?id=6043

⁵⁰ http://bugzilla.globus.org/bugzilla/show_bug.cgi?id=6065

⁵¹ http://bugzilla.globus.org/bugzilla/show_bug.cgi?id=6069

⁵² http://bugzilla.globus.org/bugzilla/show_bug.cgi?id=6072

⁵³ http://bugzilla.globus.org/bugzilla/show_bug.cgi?id=6091

⁵⁴ http://bugzilla.globus.org/bugzilla/show_bug.cgi?id=6192

- [Bug 6203](#).⁵⁵ audit record not inserted for RSL argument containing single quotes
- [Bug 6204](#).⁵⁶ Drain Mode for the GRAM service
- [Bug 6279](#).⁵⁷ document streaming overhead in globusrun-ws
- [Bug 6289](#).⁵⁸ ws-gram multiJob submissions fail with extensions element
- [Bug 6350](#).⁵⁹ Infinite loop in org.globus.exec.service.utils.UserProxyCreator
- [Bug 6351](#).⁶⁰ Infinite loop in org.globus.exec.service.utils.UserProxyCreator
- [Bug 6376](#).⁶¹ WS GRAM container becomes non responsive
- [Bug 6387](#).⁶² add service security descriptor checks in job submission
- [Bug 6400](#).⁶³ Fix audit logging problems in Gram2 and Gram4 in globus_4_0_branch
- [Bug 6402](#).⁶⁴ Throughput tester destroying shared delegation credentials

⁵⁵ http://bugzilla.globus.org/bugzilla/show_bug.cgi?id=6203

⁵⁶ http://bugzilla.globus.org/bugzilla/show_bug.cgi?id=6204

⁵⁷ http://bugzilla.globus.org/bugzilla/show_bug.cgi?id=6279

⁵⁸ http://bugzilla.globus.org/bugzilla/show_bug.cgi?id=6289

⁵⁹ http://bugzilla.globus.org/bugzilla/show_bug.cgi?id=6350

⁶⁰ http://bugzilla.globus.org/bugzilla/show_bug.cgi?id=6351

⁶¹ http://bugzilla.globus.org/bugzilla/show_bug.cgi?id=6376

⁶² http://bugzilla.globus.org/bugzilla/show_bug.cgi?id=6387

⁶³ http://bugzilla.globus.org/bugzilla/show_bug.cgi?id=6400

⁶⁴ http://bugzilla.globus.org/bugzilla/show_bug.cgi?id=6402

Chapter 13. Usage statistics collection by the Globus Alliance

1. GRAM4-specific usage statistics

The following usage statistics are sent by default in a UDP packet (in addition to the GRAM component code, packet version, timestamp, and source IP address) at the end of each job (i.e. when Done, Failed, UserTerminateDone or UserTerminateFailed state is entered).

- job creation timestamp (helps determine the rate at which jobs are submitted)
- *scheduler* type (Fork, *PBS*, *LSF*, *Condor*, etc...)
- jobCredentialEndpoint present in *RSL* flag (to determine if server-side user proxies are being used)
- fileStageIn present in RSL flag (to determine if the staging in of files is used)
- fileStageOut present in RSL flag (to determine if the staging out of files is used)
- fileCleanUp present in RSL flag (to determine if the cleaning up of files is used)
- CleanUp-Hold requested flag (to determine if streaming is being used)
- job type (Single, Multiple, MPI, or Condor)
- gt2 error code if job failed (to determine common scheduler script errors users experience)
- fault class name if job failed (to determine general classes of common faults users experience)

If you wish to disable this feature, please see the "Usage Statistics Configuration" section of [Configuring Java WS Core](#) for instructions.

Also, please see our [policy statement](#)¹ on the collection of usage statistics.

¹ ../../Usage_Stats.html

Glossary

C

Condor A job scheduler mechanism supported by GRAM. See <http://www.cs.wisc.edu/condor/> for more information.

G

globusrun-ws A command line program used to submit jobs to a GRAM4 service. See the the GRAM4 Commandline page.

J

job description Term used to describe a GRAM4 job for GT4.

job scheduler See the term [scheduler](#)⁴.

L

LSF A job scheduler mechanism supported by GRAM.
For more information, see <http://www.platform.com/Products/Platform.LSF.Family/Platform.LSF/>⁷.

M

Managed Executable Job Service (MEJS) [FIXME]

multijob A job that is itself composed of several executable jobs; these are processed by the MMJS subjob.

See also [MMJS subjob](#)¹⁰.

P

Portable Batch System (PBS) A job scheduler mechanism supported by GRAM. For more information, see <http://www.openpbs.org>.

R

Resource Specification Language (RSL) Term used to describe a GRAM job for GT2 and GT3. (Note: This is not the same as RLS - the Replica Location Service)

⁴ #scheduler

⁷ <http://www.platform.com/Products/Platform.LSF.Family/Platform.LSF/>

¹⁰ #mmjs-subjob

S

scheduler

Term used to describe a job scheduler mechanism to which GRAM interfaces. It is a networked system for submitting, controlling, and monitoring the workload of batch jobs in one or more computers. The jobs or tasks are scheduled for execution at a time chosen by the subsystem according to an available policy and availability of resources. Popular job schedulers include Portable Batch System (PBS), Platform LSF, and IBM LoadLeveler.

U

Universally Unique Identifier (UUID)

Identifier that is immutable and unique across time and space.

W

Web Services Addressing (WSA)

The WS-Addressing specification defines transport-neutral mechanisms to address web services and messages. Specifically, it defines XML elements to identify web service endpoints and to secure end-to-end endpoint identification in messages. See the [W3C WS Addressing Working Group](http://www.w3.org/2002/ws/addr/)¹⁴ for details.

¹⁴ <http://www.w3.org/2002/ws/addr/>

Index

B

- bugs
 - outstanding, 49

C

- command line client
 - globusrun-ws, 35

E

- errors, 43, 46

J

- jobs
 - hold and release, 28
 - lifetime, 26
 - client-side settings, 27
 - server-side settings, 26
 - preparing
 - delegating credentials, 1
 - finding available LRMs, 2
 - finding default LRM, 2
 - generate valid proxy, 1
 - querying
 - all jobs in a GRAM4 instance, 24
 - resource properties, 22
 - rendezvous, 33
 - SoftEnv keys, 31
 - submission ID, 30
 - submitting
 - MPIJob, 21
 - multijob, 20
 - request streaming output, 13
 - simple batch, 12
 - simple interactive, 12
 - specifying LRM, default, 15
 - specifying LRM, non-default, 16
 - using a job description, 14
 - using a job description with contact string, 15
 - using contact string, 13
 - using custom job description extensions, 19
 - using substitution variables, 19
 - with job description, specifying local user, 19
 - with staging, 17
 - terminating, 25
 - batch mode, 25
 - interactive mode, 25

L

- limitations, 49

S

- submitting jobs
 - globusrun-ws, 35

T

- troubleshooting, 42
 - check container log, 42
 - check documentation, 42
 - errors, 42
 - mailing lists, 42

U

- usage statistics, 53

GT 4.2.1 GRAM4: Developer's Guide

GT 4.2.1 GRAM4: Developer's Guide

Introduction

This guide is intended to help a developer create compatible GRAM4 clients and alternate service implementations.

The key concepts for the GRAM component have not changed. Its purpose is still to provide the mechanisms to execute remote applications for a user. Given an XML job description, GRAM submits the job to a scheduling system such as *PBS* or *Condor*, or to a simple fork-based way of spawning processes, and monitors it until completion. More details can be found here:

<http://www.globus.org/toolkit/docs/3.2/gram/key>

Table of Contents

1. Before you begin	1
1. Feature summary	1
2. Tested platforms	1
3. Backward compatibility summary	2
4. Technology dependencies	2
5. Security Considerations	2
2. Selective Concepts	3
1. Job Submission	3
2. Job status change subscriptions	3
3. Job Termination	4
3. Scenarios	9
1. C	9
2. Java	15
4. Tutorials	22
5. Architecture and design overview	23
1. Job States	23
6. APIs	27
1. Programming Model Overview	27
2. Component API	27
7. Services and WSDL	29
1. Protocol overview	29
2. Operations	29
3. Resource properties	31
4. WSDL and Schema Definition	36
8. Debugging	38
1. Development Logging in Java WS Core	38
2. Enabling debug logging for GRAM classes	38
3. Instrumented timings logging	39
4. Debugging script execution	40
9. Troubleshooting	41
1. Troubleshooting tips	41
2. Java WS Core Errors	42
3. Errors	45
10. Related Documentation	48
11. Internal Components	49
Glossary	50
Index	52

List of Figures

5.1. Managed Executable Job Service Internal State Transition Diagram	26
---	----

List of Tables

9.1. Java WS Core Errors	43
9.2. GRAM4 Errors	46

Chapter 1. Before you begin

1. Feature summary

New Features new since 4.0.x

- New terminate method in the client-side GramJob API
- Improved job lifetime management for users and admins
- Added configuration for "default" Local Resource Managers

Other Standard Supported Features

- Remote job execution and management
- Uniform and flexible interface to batch scheduling systems
- File staging before and after job execution
- File / directory clean up after job execution (after file stage out)
- Service auditing for each submitted

Deprecated Features

- With the addition of the new terminate method in the GramJob API, the destroy method is no longer necessary. For backward compatibility, the destroy method was left in the GramJob API, but it simply calls the terminate method. During the 4.2.x series, clients using the destroy method should change to instead use terminate. In GT 4.4, the plan is to remove the destroy method.

2. Tested platforms

Tested platforms for GRAM4:

- Linux
 - Fedora Core 1 i686
 - Fedora Core 3 i686
 - Fedora Core 3 yup xeon
 - RedHat 7.3 i686
 - RedHat 9 x86
 - Debian Sarge x86
 - Debian 3.1 i686

Tested containers for GRAM4:

- Java WS Core container

3. Backward compatibility summary

Protocol changes since GRAM4 in the GT4.0 series:

- The Java WS Core Framework has been updated from the draft versions of the WSRF/WSN and WS Addressing specifications to the final versions WSRF 1.2, WSN 1.3 and WS Addressing 1.0. There is no backward compatibility between this version and any previous versions.

4. Technology dependencies

GRAM depends on the following GT components:

- Java WS Core
- Transport-Level Security
- Delegation Service
- RFT
- GridFTP
- MDS - internal libraries
- The XML::Parser Perl module is required <http://search.cpan.org/~msergeant/XML-Parser/Parser.pm>

Other scheduler adapters available for GT 4.2.1 release:

- [SGE scheduler adapter interface](#)¹
- IBM LoadLeveler (As of release 3.3.1). For more information see "What's new" in the [LoadLeveler product documentation](#)²
- other batch schedulers... (where the GRAM scheduler interface has been implemented)

5. Security Considerations

No special security considerations exist at this time.

¹ <http://www.lesc.ic.ac.uk/projects/SGE-GT4.html>

² <http://publib.boulder.ibm.com/infocenter/clresctr/index.jsp?topic=/com.ibm.cluster.infocenter.doc/library.html>

Chapter 2. Selective Concepts

This chapter gives an overview and specifics about various topics and concepts of GRAM4.

1. Job Submission

1.1. Client-side generated job ID

A client can generate a job-id and pass it to the call to `ManagedJobFactoryService.createManagedJob()`. This id has been subject of misunderstandings. Please check the section [Client-Side Generated Submission ID](#) in the [User's guide](#) if you have doubts about it.

1.2. Job lifetime

A client can provide a lifetime for a job in job submission. However, sometimes it's hard or impossible to estimate an appropriate lifetime, especially with factors beyond the user's knowledge, like queuing time in the remote local resource manager or load in the remote GT4 container. Please check the following links for information about job lifetime concepts in GRAM4 in GT 4.2:

- [Job Lifetime](#) in the [Execution key concepts](#)
- [Job Lifetime](#) in the [User's guide](#)

2. Job status change subscriptions

2.1. Subscribing for status change notifications

A client can subscribe for job status notifications of a job in two ways

- Subscribe on the call to `ManagedJobFactoryService.createManagedJob()`: GRAM4 creates the subscription resource for the client and returns an EPR of it as part of the response of the `createManagedJob()` call.
- Subscribe in a separate WS call after the call to `ManagedJobFactoryService.createManagedJob()` returned.

The second approach has two disadvantages:

1. Two WS calls as opposed to one WS call in the first approach.
2. There's a risk to miss notifications, because the subscribe call is done after the job had been created.

2.2. Destroying subscription resources

Subscription resources of a job are destroyed automatically on the server-side when the job resource goes away, i.e. when a job resource is destroyed on the server-side. There's no need for the client to manually destroy a subscription resource.

3. Job Termination

3.1. Introduction

Job termination can happen for three reasons:

1. **Processing errors:** Any error that occurs during job processing. This can be an error when files are staged in, an invalid executable, an error of the job in the local resource manager, an error while staging files out, etc.
2. **Job resource expiration:** If the lifetime specified by a client the job will be terminated.
3. **Client cancellation:** A client requests the termination of a job.

If a job is still running and not already fully processed, termination will cause the job to go through a series cleanup steps in GRAM4 before the job-related data is destroyed. The cleanup steps being performed depend on the job and the state it is in. In general this includes cancellation of a running job at the local resource manager and running fileCleanUp if so specified in the job description. Termination at the local resource manager however will only be performed if the job did not already finished executing. This also applies to fileCleanUp: If no fileCleanUp is specified in the job description or if the job already passed fileCleanUp when the termination request comes in, then this step is skipped.

As in normal processing, errors may also happen in the clean up phase. It might e.g. be interesting for a user whether the cancellation of a running job at the local resource manager was successful or not, or whether the specified fileCleanUp had been processed successfully .

This section focuses on the interface GRAM4 offers a client to terminate a job.

In GRAM4 in 4.2 a client cannot request synchronous destruction of a job anymore like it was in GRAM4 in the 4.0 series. The reason for that is that many concurrent destroy requests at a time can cause the GT4 container to become unresponsive. Unlike destroy() the call to terminate() is asynchronous, i.e. returns quickly and does not block until the cleanup steps are all done. The new termination method applies to both ManagedExecutableJobResources (MEJRs) and ManagedMultiJobResources (MMJRs) and is a replacement for the destroy() method in GRAM4 in the 4.0 series.

The new termination method is supposed to cope with different scenarios for jobs in different states, which resulted in an interface that might not be intuitive at first glance. This section explains parameters, return value and faults of the call to terminate() and the implications for jobs in various states.

3.2. User termination method definition

The terminate() method is defined as follows

```
public TerminateOutputType terminate (  
    TerminateInputType parameters)  
    throws RemoteException, ResourceUnknownFaultType,  
    DelegatedCredentialDestroyFaultType,  
    ResourceNotTerminatedFaultType
```

TerminateInputType contains:

- boolean destroyAfterCleanup
- boolean continueNotifying
- boolean destroyDelegatedCredentials

TerminateOutputType contains:

- boolean terminationCompleted

3.3. Explanations for MEJRs

3.3.1. Arguments

destroyAfterCleanup: If set to true the job resource will be destroyed once all cleanup steps are done.

continueNotifying: If set to true a client will be notified about the success of the termination. This happens via a notification bound to the same topic a client subscribes to for normal state change information. A client who is not interested in the success of the termination can set this to false.

destroyDelegatedCredentials: If this is set to true, all delegated credentials that are specified in the job description (if any) will be destroyed after all cleanup steps are done. They must not be destroyed earlier because staging credentials are needed during failureFileCleanUp. Setting this parameter and destroyAfterCleanup both to true enables a client to completely go away after the termination c

3.3.2. Return value

terminationCompleted: Indicates whether termination of the MEJR completed or not when the call to terminate() returns. It is true in case the job is already in a final state, i.e. no cleanup steps had to be done. Otherwise it is false.

If it is true no further notifications will be sent, even if the client requested it, because the job is in a final state where no state transition is happening anymore. If destroyAfterCleanup had been set to true the client can be sure that the job resource has been destroyed when the call to terminate() returns.

If it is false, the job entered the clean up phase and the client can find out about success or failure of the termination either by listening to state change notifications or by querying the status of the MEJR.

3.3.3. Exceptions

ResourceUnknownFaultType: Thrown when the job resource to be terminated does not exist.

DelegatedCredentialDestroyFaultType: Thrown if the client demanded the destruction of delegated credentials and this failed. If this exception is thrown all termination steps succeeded, and only the destruction of the delegated credential failed.

ResourceNotTerminatedFaultType An error occurred during termination. This should not happen at all, but in case unforeseen things happen it will indicate that termination failed.

3.4. Explanations for MMJRs

3.4.1. Arguments

destroyAfterCleanup: If this is set to true the job resource will be destroyed once all cleanup steps are done. This is when the termination calls to all SJs went successfully and the MMJR received final notifications of all SJs.

continueNotifying: If set to true a client will be notified about the success of the termination. This happens via a notification bound to the same topic a client subscribes to for normal state change information. A client who is not interested in the success of the termination can set this to false.

destroyDelegatedCredentials: If this is set to true, all delegated credentials for the MMJR that are specified in the job description will be destroyed when the MMJR is destroyed. They must not be destroyed earlier because job credentials are needed for potential repetitive termination calls. Otherwise the MMJR wouldn't be able to interact with SJ's anymore.

Setting this parameter and `destroyAfterCleanup` both to true enables a client to completely go away after the termination call in case the success of the cleanup steps is not of importance.

3.4.2. Return value

terminationCompleted: Indicates whether termination of the MMJR completed or not. It is true in case the job is already in a final state, i.e. all SJs are in a final state. Otherwise it is false.

If it is true no further notifications will be sent, even if the client requested it, because the job is in a final state where no state transition is happening anymore. If `destroyAfterCleanup` had been set to true the client can be sure that the job resource has been destroyed when the call to `terminate()` returns.

If it is false, the job entered the clean up phase, i.e. termination calls to all subjobs had been sent, and the client can find out about success or failure of the termination either by listening to state change notifications or by querying the status of the MMJR.

3.4.3. Exceptions

ResourceUnknownFaultType: Thrown when the resource to be terminated does not exist.

DelegatedCredentialDestroyFaultType: Thrown if the client demanded the destruction of delegated credentials and this failed. If this exception is thrown all termination steps succeeded, and only the destruction of the delegated credential failed.

ResourceNotTerminatedFaultType: An error occurred during the call to terminate e.g. a termination call to at least one SJ caused a `ResourceNotTerminatedFaultType` exception. Note that a failure in terminating a SJ does not prevent from terminate calls to other SJs. But in this case, the MMJR will not be destroyed after an error from terminating one of the SJs.

3.5. New final job states

Additional to Done and Failed, two new final states `UserTerminateDone` and `UserTerminateFailed` are introduced. These states are part of the notification message sent if a client subscribed for notifications, and they are values of the `ResourceProperty state` that can be queried by a client. Final states and their meaning:

3.5.1. Explanations for MEJRs

Done: The MEJR has been fully processed

Failed: A processing error occurred that resulted in a termination initialized by GRAM4.

UserTerminateDone: The client had called `terminate()` and all cleanup steps have been processed successfully.

UserTerminateFailed: The client had called `terminate()` and at least one cleanup step had not been processed successfully.

3.5.2. Explanations for MMJRs

Done: All SJs are in state Done

Failed: Failed is not necessarily a final state, because a MMJR transitions into state Failed if the first SJ fails. Failed is only a final state if at least one SJ failed and all other SJs are in state Done or Failed. If just one out of N (N>1) SJs failed and the client terminates the MMJR, the state Failed will transition to state UserTerminateDone in case of termination success or to state.

UserTerminateDone: At least one SJ is in state UserTerminateDone and all other SJs are in state UserTerminateDone, Failed or Done.

UserTerminateFailed: At least one SJ is in state UserTerminateFailed and all other SJs are in state UserTerminateDone, UserTerminateFailed, Done or Failed.

3.6. Termination related faults

UserTerminateDone and UserTerminateFailed indicate only whether termination was successful or not. In case of errors in the clean up phase the following faults indicate what had happened. They are part of the notification message to client in case termination failed, and are set in the ResourceProperty `fault`.

These faults will only be set in a resource or are sent to a client as part of a notification message if the clean up steps have to be performed and the job did not already pass the correspondent state. As an example: For a job, that run to completion in the local resource manager, no attempt will be made to cancel it in the local resource manager.

No automated action can be done when these faults are found, they just indicate problems that would have to be manually taken care of.

StagingTerminateFaultType: Interruption of a running transfer failed.

LocalResourceManagerJobTerminateFaultType: Cancellation of the job at the local resource manager failed.

DelegatedCredentialDestroyFaultType: Destroying delegated credentials failed.

3.7. Termination scenarios for MEJRs

Now, all the above might be a bit abstract and hard to understand. Probably the most common scenarios are summarized in this section and illustrate which settings should be chosen in a certain scenario. Since the Java API `GramJob` is commonly used, some of these use-cases refer to it.

1. **A client let `GramJob` delegate and wants to terminate a job and just go away. Information about success of the termination is unimportant.**

```
destroyAfterCleanup=true, continueNotifying=false, destroyDelegatedCredentials=true
```

2. **A client delegated itself and does not want the delegated credential to be destroyed and wants to terminate a job and just go away. Information about success of the termination is unimportant.**

Same like in (1) but `destroyDelegatedCredentials=false`

3. **A client let `GramJob` delegate and wants to terminate a job, get information about the success of termination, subscribed for notifications.**

```
destroyAfterCleanup=false, destroyDelegatedCredentials=true continueNotifying=true
```

If the call to `terminate()` returns true the job has been fully terminated and destroyed. If it returns 'false' the client has to wait for the final notification message of the job (state `UserTerminateDone`, `UserTerminateFailed`). In case

of problems it can find out from the faults that are part of the notification message what happened. In case it does not hear about the job the notification message might have been lost and the client should query the RP 'state' for job status the RP 'fault' to check what happened.

Once termination finished the client should send a second termination call to request destruction of the job resource with parameters `destroyAfterCleanup=true`, `destroyDelegatedCredentials=false` `continueNotifying=false`

4. **A client delegated itself and does not want the delegated credential to be destroyed and wants to terminate a job, get information about the success of termination, subscribed for notifications.**

Same like (3), but `destroyDelegatedCredential=false`

5. **A client let GramJob delegate and wants to terminate a job, get information about the success of termination, didn't subscribe for notifications.**

Same like (3), but periodical querying the RP's state and fault is necessary in case the call to terminate didn't return true.

6. **A client delegated itself and does not want the delegated credential to be destroyed and wants to terminate a job, get information about the success of termination, didn't subscribe for notifications.**

Same like (4), but periodical querying the RP's state and fault is necessary in case the call to terminate didn't return true.

3.8. Repetitive termination

Termination is a kind of one-way street: Once a clean up step is passed there's no way back, and there's no way to repeat a certain step. The only thing GRAM4 provides is information what went wrong by providing the faults along the way. Repetitive termination calls e.g.do not cause repetitive cancellations at the local resource manager.

Repetitive termination calls however can make sense for scenario 3 in the last section: Terminate a job, but don't destroy after clean up to ensure that no information is lost. Then do a second termination call to destroy the job resource.

Chapter 3. Scenarios

1. C

The following is a general scenario for submitting a job using the C stubs and APIs. Please consult the [C WS Core API](#), [GRAM4 API](#) documentation for details on the APIs used in the code excerpts.

1.1. Loading the job description

```
const char *                file = "job.xml";
globus_soap_message_handle_t message;
wsgram_CreateManagedJobInputType1 input;

globus_soap_message_handle_init_from_file(&message, file);

globus_soap_message_deserialize_element_unknown(message, &element);

if(strcmp(element.local, "job") == 0)
{
    wsgram_JobDescriptionType2 *    jd;

    input.choice_value.type = wsgram_CreateManagedJobInputType_job;
    jd = &input.choice_value.value.job;

    wsgram_JobDescriptionType_deserialize(&element, jd, message, 0);
}
else if(strcmp(element.local, "multiJob") == 0)
{
    wsgram_JobDescriptionType3 *    mjd;

    input.choice_value.type = wsgram_CreateManagedJobInputType_multiJob;
    mjd = &input.choice_value.value.multiJob;

    wsgram_MultiJobDescriptionType_deserialize(&element, mjd, message, 0);
}
xsd_QName_destroy_contents(&element);
globus_soap_message_handle_destroy(message);
```

This code sets the choice value of the `wsgram_CreateManagedJobInputType` to be the appropriate type depending on whether the *job description* is a job or *multijob* request.

1.2. Setting the security attributes

```
globus_soap_message_attr_t    message_attr;
```

¹ [/api/c/globus_c_gram_client_bindings/html/group__wsgram__CreateManagedJobInputType.html](#)

² [/api/c/globus_c_gram_client_bindings/html/group__wsgram__JobDescriptionType.html](#)

³ [/api/c/globus_c_gram_client_bindings/html/group__wsgram__MultiJobDescriptionType.html](#)

```

globus_soap_message_attr_init(&message_attr);

/*
 * Set authentication mode to host authorization: other possibilities are
 * GLOBUS_SOAP_MESSAGE_AUTHZ_HOST_IDENTITY or
 * GLOBUS_SOAP_MESSAGE_AUTHZ_HOST_SELF.
 */
globus_soap_message_attr_set(
    message_attr,
    GLOBUS_SOAP_MESSAGE_AUTHZ_METHOD_KEY,
    NULL,
    NULL,
    (void *) GLOBUS_SOAP_MESSAGE_AUTHZ_HOST);

/*
 * Set message protection level. GLOBUS_SOAP_MESSAGE_AUTH_PROTECTION_PRIVACY
 * for encryption.
 */
globus_soap_message_attr_set(
    message_attr,
    GLOBUS_SOAP_MESSAGE_AUTH_PROTECTION_KEY,
    NULL,
    NULL,
    (void *) GLOBUS_SOAP_MESSAGE_AUTH_PROTECTION_PRIVACY);

```

1.3. Creating the factory client handle

```

ManagedJobFactoryService_client_handle_t    factory_handle;

result = ManagedJobFactoryService_client_init(
    &factory_handle,
    message_attr,
    NULL);

```

1.4. Querying for factory resource properties

1.4.1. One at a time

```

/*
 * localResourceManager, or other resource property names as defined in the
 * WSDL
 */
xsd_QName                                property_name =
{
    "http://www.globus.org/namespaces/2008/03/gram/job",
    "localResourceManager"
};
wsrp_GetResourcePropertyResponseType *    property_response;
int                                        fault_type;
xsd_any *                                  fault;

```

```
ManagedJobFactoryPortType_GetResourceProperty(
    factory_handle,
    endpoint,
    &property_name,
    &property_response,
    (ManagedJobFactoryPortType_GetResourceProperty_fault_t *) &fault_type,
    &fault);
```

If this is successful, then `property_response`'s `any` field will contain the deserialized data in the `value` field of the first element in the array.

```
xsd_string * localResourceManager = property_response->any.elements[0].value;

printf("local resource manager is %s\n", *localResourceManager);
```

1.5. Creating the notification consumer

The notification consumer can be either passed in as part of the `wsgam_CreateManagedJobInputType` or through a separate invocation of `ManagedJobPortType_Subscribe_epr()`.

```
globus_service_engine_t          engine;
wsa_EndpointReferenceType        consumer_reference;

globus_service_engine_init(&engine, NULL, NULL, NULL, NULL, NULL);

globus_notification_create_consumer(
    &consumer_reference,
    engine,
    notify_callback,
    NULL);
```

1.6. Creating the job resource

First, prepare the other parts of the `wsgam_CreateManagedJobInputType` structure.

```
/*
 * You can set input.InitialTerminationTime to be a timeout if interested.
 * The xsd_dateTime type is a struct tm pointer.
 */
time_t          term_time = time(NULL);
globus_uuid_t   uuid;
wsa_AttributedURI * job_id;
wsa_EndpointReferenceType * factory_epr;
xsd_any *       reference_property;
wsgam_CreateManagedJobOutputType * output = NULL;
xsd_QName       factory_reference_id_qname =
{
    "http://www.globus.org/namespaces/2008/03/gram/job",
    "ResourceID"
```

```

};

term_time += 60 * 60; /* 1 hour later */
xsd_dateTime_copy(&input.InitialTerminationTime, gmtime(&term_time));

/*
 * Set unique JobID. This is used to reliably create jobs and check for status.
 */
globus_uuid_create(&uuid);
wsa_AttributedURI_init(&job_id);
job_id->base_value = globus_common_create_string("uuid:%s", uuid.text);

/* To subscribe to notifications at create time, add the consumer's EPR to
 * the input message. Otherwise, use the EPR created above in a
 * call to
 */
wsnt_SubscribeType_init(&input.Subscribe);
wsa_EndpointReferenceType_copy_contents(
    &input.Subscribe.ConsumerReference,
    &consumer_reference);

xsd_any_init(&input.Subscribe->TopicExpression.any);
&input.Subscribe->TopicExpression.any->any_info =
    &xsd_QName_contents_info;
xsd_QName_copy(
    (xsd_QName **) &input.Subscribe->TopicExpression.any->any.value,
    &ManagedJobPortType_state_rp_qname);

xsd_anyURI_copy_cstr(
    &input.Subscribe->TopicExpression._Dialect,
    "http://docs.oasis-open.org/wsn/2004/06/TopicExpression/Simple");
xsd_boolean_init(&input.Subscribe->UseNotify);

*(&input.Subscribe->UseNotify) = GLOBUS_TRUE;

/* Construct the EPR of the job factory */
wsa_EndpointReferenceType_init(&factory_epr);
wsa_AttributedURI_init_contents(&factory_epr->Address);
xsd_anyURI_init_contents_cstr(&factory_epr->Address.base_value,
    globus_common_create_string(
        "https://%s:%hu/wsrp/services/%s",
        factory_host,
        factory_port,
        MANAGEDJOBFACTORYSERVICE_BASE_PATH);
wsa_ReferenceParametersTypeinit(&factory_epr->ReferenceParameters);
reference_property = xsd_any_array_push(
    &factory_epr->ReferenceParameters.any);
reference_property->any_info = &xsd_string_info;
xsd_QName_copy(
    &reference_property->element,
    &factory_reference_id_qname);

xsd_string_copy_cstr(
    (xsd_string **) &reference_property->value,

```

```
    "Fork");

/* Submit the request to the service container */
ManagedJobFactoryPortType_createManagedJob_epr(
    factory_handle,
    factory_epr,
    input,
    &output,
    (ManagedJobFactoryPortType_createManagedJob_fault_t *) &fault_type,
    &fault);
```

If this is successful, then the `output` structure will be initialized with the results of the operation. Of particular interest is the `managedJobEndpoint` which contains the reference to the newly-created job resource.

1.7. Subscribing for job state notifications

In order to subscribe for job state change notifications to an existing job resource, initialize the `subscribe_input` used below in the same way as `input.Subscribe` was initialized above.

```
ManagedJobService_client_handle_t    job_handle;
wsnt_SubscribeType                    subscribe_input;
wsnt_SubscribeResponseType *          subscribe_response;

ManagedJobService_client_init(
    &job_handle,
    message_attr,
    NULL);

ManagedJobPortType_Subscribe_epr(
    job_handle,
    output->managedJobEndpoint,
    subscribe_input,
    &subscribe_response,
    (ManagedJobPortType_Subscribe_fault_t *) &fault_type,
    &fault);
```

1.8. Releasing any state holds (if necessary)

```
wsgram_ReleaseInputType              release;
wsgram_ReleaseOutputType *           release_response = NULL;

wsgram_ReleaseInputType_init_contents(&release);

ManagedJobPortType_release_epr(
    job_handle,
    output->managedJobEndpoint,
    &release,
    &release_response,
    (ManagedJobPortType_release_fault_t *) &fault_type,
    &fault);
```

1.9. Destroying resources

```

/* destroy subscription resource */
SubscriptionManagerService_client_init    subscription_handle;
wsnt_DestroyType                          destroy;
wsnt_DestroyResponseType *                destroy_response = NULL;

SubscriptionManagerService_client_init(
    &subscription_handle,
    message_attr,
    NULL);
/* if subscription done at job creation time, use
 * output->subscriptionEndpoint in place of
 * subscribe_response->SubscriptionReference,
 */
SubscriptionManager_Destroy_epr(
    subscription_handle,
    subscribe_response->SubscriptionReference,
    &destroy,
    &destroy_response,
    (SubscriptionManager_Destroy_fault_t *) &fault_type,
    &fault);

/* destroy the job resource */
jobPort.destroy(new Destroy());
ManagedJobPortType_Destroy_epr(
    job_handle,
    output->managedJobEndpoint,
    &destroy,
    &destroy_response,
    (ManagedJobPortType_Destroy_fault_t *) &fault_type,
    &fault);

```

1.10. Building a client

In order to build a client application, certain flags must be passed to the compiler and linker to enable them to be able to locate headers and libraries. The easiest way to do so is to generate a *makefile header*, which is a fragment of a Makefile which includes all of the necessary flags needed to build the application. To do this, issue the command:

```
% globus-makefile-header --flavor=gcc32dbg globus_c_gram_client_bindings > Makefile.inc
```

Then, write your makefile to include this file and use the GLOBUS_CC, GLOBUS_LD, GLOBUS_CFLAGS, GLOBUS_LDFLAGS, and GLOBUS_PKG_LIBS macros. For example:

```

GLOBUS_FLAVOR_NAME=gcc32dbg

include Makefile.inc

CC = $(GLOBUS_CC)
LD = $(GLOBUS_LD)

```

```
CFLAGS = $(GLOBUS_CFLAGS)
LDFLAGS = $(GLOBUS_LDFLAGS) $(GLOBUS_PKG_LIBS)
```

```
client: client.c
```

2. Java

The following is a general scenario for submitting a job using the Java stubs and APIs. Please consult the [Java WS Core API](#), [Delegation API](#), [Reliable File Transfer API](#), and [GRAM4 API](#) documentation for details on classes referenced in the code excerpts.

Also, it will probably be helpful to look at the [GramJob class source code](#)⁴ as a functioning example.

2.1. Class imports

The following imports will be needed for these examples:

```
import java.io.File;5
import java.io.FileInputStream;6
import java.net.URL;7
import java.util.LinkedList;8
import java.util.List;9
import java.util.Vector;10
import java.security.cert.X509Certificate;11
import javax.xml.rpc.Stub;12
import javax.xml.soap.SOAPElement;13
import org.apache.axis.components.uuid.UUIDGenFactory;14
import org.apache.axis.message.addressing.AttributedURI;15
import org.apache.axis.message.addressing.EndpointReferenceType;16
import org.globus.delegation.DelegationUtil;17
import org.globus.exec.generated.CreateManagedJobInputType;18
import org.globus.exec.generated.CreateManagedJobOutputType;19
import org.globus.exec.generated.ManagedJobFactoryPortType;20
import org.globus.exec.generated.ManagedJobPortType;21
```

⁴ http://viewcvs.globus.org/viewcvs.cgi/ws-gram/client/java/source/src/org/globus/exec/client/GramJob.java?rev=1.129.2.3&only_with_tag=globus_4_0_branch&content-type=text/vnd.viewcvs-markup

⁵ <http://java.sun.com/j2se/1.4.2/docs/api/java/io/File.html>

⁶ <http://java.sun.com/j2se/1.4.2/docs/api/java/io/FileInputStream.html>

⁷ <http://java.sun.com/j2se/1.4.2/docs/api/java/net/URL.html>

⁸ <http://java.sun.com/j2se/1.4.2/docs/api/java/util/LinkedList.html>

⁹ <http://java.sun.com/j2se/1.4.2/docs/api/java/util/List.html>

¹⁰ <http://java.sun.com/j2se/1.4.2/docs/api/java/util/Vector.html>

¹¹ <http://java.sun.com/j2se/1.4.2/docs/api/java/security/cert/X509Certificate.html>

¹² <http://java.sun.com/j2ee/1.4/docs/api/javax/xml/rpc/Stub.html>

¹³ <http://java.sun.com/j2ee/1.4/docs/api/javax/xml/soap/SOAPElement.html>

¹⁴ <http://ws.apache.org/axis/java/apiDocs/org/apache/axis/components/uuid/UUIDGenFactory.html>

¹⁵ <http://ws.apache.org/addressing/apidocs/org/apache/axis/message/addressing/AttributedURI.html>

¹⁶ <http://ws.apache.org/addressing/apidocs/org/apache/axis/message/addressing/EndpointReferenceType.html>

¹⁷ http://www.globus.org/api/javadoc-4.0/globus_wsrf_delegation_service_java/org/globus/delegation/DelegationUtil.html

¹⁸ http://www.globus.org/api/javadoc-4.0/globus_wsrf_gram_common_java/org/globus/exec/generated/CreateManagedJobInputType.html

¹⁹ http://www.globus.org/api/javadoc-4.0/globus_wsrf_gram_common_java/org/globus/exec/generated/CreateManagedJobOutputType.html

²⁰ http://www.globus.org/api/javadoc-4.0/globus_wsrf_gram_common_java/org/globus/exec/generated/ManagedJobFactoryPortType.html

²¹ http://www.globus.org/api/javadoc-4.0/globus_wsrf_gram_common_java/org/globus/exec/generated/ManagedJobPortType.html

```

import org.globus.exec.generated.ReleaseInputType;22
import org.globus.exec.utils.ManagedJobConstants;23
import org.globus.exec.utils.ManagedJobFactoryConstants;24
import org.globus.exec.utils.client.ManagedJobClientHelper;25
import org.globus.exec.utils.client.ManagedJobFactoryClientHelper;26
import org.globus.exec.utils.rsl.RSLHelper;27
import org.globus.wsrfl.NotificationConsumerManager;28
import org.globus.wsrfl.WSNConstants;29
import org.globus.wsrfl.encoding.ObjectDeserializer;30
import org.globus.wsrfl.impl.security.authentication.Constants;31
import org.globus.wsrfl.impl.security.authorization.Authorization;32
import org.globus.wsrfl.impl.security.authorization.HostAuthorization;33
import org.globus.wsrfl.impl.security.authorization.IdentityAuthorization;34
import org.globus.wsrfl.impl.security.authorization.SelfAuthorization;35
import org.globus.wsrfl.impl.security.descriptor.ClientSecurityDescriptor;36
import org.globus.wsrfl.impl.security.descriptor.GSISecureMsgAuthMethod;37
import org.globus.wsrfl.impl.security.descriptor.GSITransportAuthMethod;38
import org.globus.wsrfl.impl.security.descriptor.ResourceSecurityDescriptor;39
import org.gridforum.jgss.ExtendedGSSManager;40
import org.oasis.wsn.Subscribe;41
import org.oasis.wsn.SubscribeResponse;42
import org.oasis.wsn.SubscriptionManager;43
import org.oasis.wsn.TopicExpressionType;44
import org.oasis.wsn.WSBaseNotificationServiceAddressingLocator;45
import org.oasis.wsrfl.lifetime.Destroy;46
import org.oasis.wsrfl.properties.GetMultipleResourceProperties_Element;47
import org.oasis.wsrfl.properties.GetMultipleResourcePropertiesResponse;48
import org.oasis.wsrfl.properties.GetResourcePropertyResponse;49

```

²² http://www.globus.org/api/javadoc-4.0/globus_wsrfl_gram_common_java/org/globus/exec/generated/ReleaseInputType.html

²³ http://www.globus.org/api/javadoc-4.0/globus_wsrfl_gram_utils_java/org/globus/exec/utils/ManagedJobConstants.html

²⁴ http://www.globus.org/api/javadoc-4.0/globus_wsrfl_gram_utils_java/org/globus/exec/utils/ManagedJobFactoryConstants.html

²⁵ http://www.globus.org/api/javadoc-4.0/globus_wsrfl_gram_utils_java/org/globus/exec/utils/client/ManagedJobClientHelper.html

²⁶ http://www.globus.org/api/javadoc-4.0/globus_wsrfl_gram_utils_java/org/globus/exec/utils/client/ManagedJobFactoryClientHelper.html

²⁷ http://www.globus.org/api/javadoc-4.0/globus_wsrfl_gram_utils_java/org/globus/exec/utils/rsl/RSLHelper.html

²⁸ http://www.globus.org/api/javadoc-4.0/globus_java_ws_core/org/globus/wsrfl/impl/notification/ClientNotificationConsumerManager.html

²⁹ http://www.globus.org/api/javadoc-4.0/globus_java_ws_core/org/globus/wsrfl/WSNConstants.html

³⁰ http://www.globus.org/api/javadoc-4.0/globus_java_ws_core/org/globus/wsrfl/encoding/ObjectDeserializer.html

³¹ http://www.globus.org/api/javadoc-4.0/globus_java_ws_core/org/globus/wsrfl/impl/security/authentication/Constants.html

³² http://www.globus.org/api/javadoc-4.0/globus_java_ws_core/org/globus/wsrfl/impl/security/authorization/Authorization.html

³³ http://www.globus.org/api/javadoc-4.0/globus_java_ws_core/org/globus/wsrfl/impl/security/authorization/HostAuthorization.html

³⁴ http://www.globus.org/api/javadoc-4.0/globus_java_ws_core/org/globus/wsrfl/impl/security/authorization/IdentityAuthorization.html

³⁵ http://www.globus.org/api/javadoc-4.0/globus_java_ws_core/org/globus/wsrfl/impl/security/authorization/SelfAuthorization.html

³⁶ http://www.globus.org/api/javadoc-4.0/globus_java_ws_core/org/globus/wsrfl/impl/security/descriptor/ClientSecurityDescriptor.html

³⁷ http://www.globus.org/api/javadoc-4.0/globus_java_ws_core/org/globus/wsrfl/impl/security/descriptor/GSISecureMsgAuthMethod.html

³⁸ http://www.globus.org/api/javadoc-4.0/globus_java_ws_core/org/globus/wsrfl/impl/security/descriptor/GSITransportAuthMethod.html

³⁹ http://www.globus.org/api/javadoc-4.0/globus_java_ws_core/org/globus/wsrfl/impl/security/descriptor/ResourceSecurityDescriptor.html

⁴⁰ http://www.cogkit.org/release/4_1_2/api/jglobus/org/gridforum/jgss/ExtendedGSSManager.html

⁴¹ http://www.globus.org/api/javadoc-4.0/globus_java_ws_core/org/oasis/wsn/Subscribe.html

⁴² http://www.globus.org/api/javadoc-4.0/globus_java_ws_core/org/oasis/wsn/SubscribeResponse.html

⁴³ http://www.globus.org/api/javadoc-4.0/globus_java_ws_core/org/oasis/wsn/SubscriptionManager.html

⁴⁴ http://www.globus.org/api/javadoc-4.0/globus_java_ws_core/org/oasis/wsn/TopicExpressionType.html

⁴⁵ http://www.globus.org/api/javadoc-4.0/globus_java_ws_core/org/oasis/wsn/WSBaseNotificationServiceAddressingLocator.html

⁴⁶ http://www.globus.org/api/javadoc-4.0/globus_java_ws_core/org/oasis/wsrfl/lifetime/Destroy.html

⁴⁷ http://www.globus.org/api/javadoc-4.0/globus_java_ws_core/org/oasis/wsrfl/properties/GetMultipleResourceProperties_Element.html

⁴⁸ http://www.globus.org/api/javadoc-4.0/globus_java_ws_core/org/oasis/wsrfl/properties/GetMultipleResourcePropertiesResponse.html

⁴⁹ http://www.globus.org/api/javadoc-4.0/globus_java_ws_core/org/oasis/wsrfl/properties/GetResourcePropertyResponse.html

2.2. Loading the job description

```
File jobDescriptionFile = new File("myjobdesc.xml");
JobDescriptionType jobDescription = RSLHelper.readRSL(jobDescriptionFile);
```

The object `jobDescription` will be of sub-type `MultiJobDescriptionType` if the file contents is a multi-job description.

2.3. Creating the factory service stub

```
URL factoryUrl = ManagedJobFactoryClientHelper.getServiceURL(
    contactString).getURL();
String factoryType
    = ManagedJobFactoryConstants.FACTORY_TYPE.<factory type constant>;
EndpointReferenceType factoryEndpoint
    = ManagedJobFactoryClientHelper.getFactoryEndpoint(factoryUrl, factoryType);
ManagedJobFactoryPortType factoryPort
    = ManagedJobFactoryClientHelper.getPort(factoryEndpoint);
```

The format of `contactString` is `[protocol://]host[:port][/servicepath]`.

2.4. Loading a proxy from a file

- Default proxy file:

```
ExtendedGSSManager manager =
    (ExtendedGSSManager)ExtendedGSSManager.getInstance();

GSSCredential cred = manager.createCredential(
    GSSCredential.INITIATE_AND_ACCEPT);
```

- Specific proxy file:

```
File proxyFile = new File("proxy_file");
byte[] proxyData = new byte[(int)proxyFile.length];
FileInputStream inputStream = new FileInputStream(proxyFile);
inputStream.read(proxyData);
inputStream.close();

ExtendedGSSManager manager =
    (ExtendedGSSManager)ExtendedGSSManager.getInstance();

GSSCredential proxy = manager.createCredential(
    proxyData,
    ExtendedGSSCredential.IMPEXP_OPAQUE,
    GSSCredential.DEFAULT_LIFETIME,
    null,
    GSSCredential.ACCEPT_ONLY);
```

2.5. Setting stub security parameters

```
ClientSecurityDescriptor secDesc = new ClientSecurityDescriptor();
secDesc.setGSITransport(Constants.<protection level constant>);
secDesc.setAuthz(<Authorization sub-class instance>);
if (proxy != null) {
    secDesc.setGSSCredential(proxy);
}
((Stub) port)._setProperty(Constants.CLIENT_DESCRIPTOR, secDesc);
```

Use setGSISecureMsg() for GSI Secure Message.

2.6. Querying for factory resource properties

2.6.1. One at a time

```
GetResourcePropertyResponse response
    = factoryport.getResourceProperty(ManagedJobConstants.<RP constant>);

SOAPElement[] any = response.get_any();

... = ObjectDeserializer.toObject(any[0], <RP type>.class);
```

2.6.2. Many at a time

```
GetMultipleResourceProperties_Element rpRequest
    = new GetMultipleResourceProperties_Element();
rpRequest.setResourceProperty(new QName[] {
    ManagedJobFactoryConstants.<RP constant #1>,
    ManagedJobFactoryConstants.<RP constant #2>,
    ManagedJobFactoryConstants.<RP constant #N>
});
GetMultipleResourcePropertiesResponse response
    = factoryPort.getMultipleResourceProperties(rpRequest);

SOAPElement[] any = response.get_any();

... = ObjectDeserializer.toObject(any[0], <RP #1 type>.class);
... = ObjectDeserializer.toObject(any[0], <RP #2 type>.class);
... = ObjectDeserializer.toObject(any[0], <RP #N type>.class);
```

2.7. Delegating credentials (if needed)

```
X509Certificate certToSign = DelegationUtil.getCertificateChainRP(
    delegFactoryEndpoint, //EndpointReferenceType
    secDesc, //ClientSecurityDescriptor
)[0]; //first element in the returned array
```

```
EndpointReferenceType credentialEndpoint = DelegationUtil.delegate(
    delegFactoryurl,        //String
    credential,             //GlobusCredential
    certToSign,            //X509Certificate
    lifetime,              //int (seconds)
    fullDelegation,       //boolean
    secDesc);              //ClientSecurityDescriptor
```

There are three types of delegated credentials:

1. Credential used by the job to generate user-owned proxy:

```
jobDescription.setJobCredential(credentialEndpoint);
```

2. Credential used to contact RFT for staging and file clean up:

```
jobDescription.setStagingCredentialEndpoint(credentialEndpoint);
```

3. Credential used by RFT to contact GridFTP servers:

```
TransferRequestType stageOut = jobDescription.getFileStageOut();
stageOut.setTransferCredential(credentialEndpoint);
```

Do the same for fileStageIn and fileCleanUp.

2.8. Creating the job resource

```
CreateManagedJobInputType jobInput = new CreateManagedJobInputType();
jobInput.setJobID(new AttributeURI("uuid: " + UUIDGenFactory.getUUIDGen().nextUUID()));
jobInput.setInitialTerminationTime(<Calendar instance>);
if (multiJob) jobInput.setMultiJob(jobDescription) else jobInput.setJob(jobDescription);
if (subscribeOnCreate) jobInput.setSubscribe(subscriptionReq);
CreateManagedJobOutputType createResponse
    = factoryPort.createManagedJob(jobInput);
EndpointReferenceType jobEndpoint = createResponse.getManagedJobEndpoint();
```

2.9. Creating the job service stub

```
ManagedJobPortType jobPort = ManagedJobClientHelper.getPort(jobEndpoint);
```

You must set the appropriate security parameters for the job service stub (jobPort) as well.

2.10. Subscribing for job state notifications

```
NotificationConsumerManager notifConsumerManager
    = NotificationConsumerManager.getInstance();
```

```
notifConsumerManager.startListening();
List topicPath = new LinkedList();
```

```

topicPath.add(ManagedJobConstants.RP_STATE);

ResourceSecurityDescriptor resourceSecDesc = new ResourceSecurityDescriptor();
resourceSecDesc.setAuthz(Authorization.<authz type constant>);

Vector authMethods = new Vector();
authMethods.add(GSITransportAuthMethod.BOTH);
resourceSecDesc.setAuthMethods(authMethods);

EndpointReferenceType notificationConsumerEndpoint
    = notifConsumerManager.createNotificationConsumer(
        topicPath,
        this,
        resourceSecDesc);

Subscribe subscriptionReq = new Subscribe();
subscriptionReq.setConsumerReference(
    notificationConsumerEndpoint);

TopicExpressionType topicExpression = new TopicExpressionType(
    WSNConstants.SIMPLE_TOPIC_DIALECT,
    ManagedJobConstants.RP_STATE);
subscriptionReq.setTopicExpression(topicExpression);

EndpointReferenceType subscriptionEndpoint;

```

- Subscribe on creation

```
jobInput.setSubscribe(subscriptionReq);
```

- Subscribe after creation

```

SubscribeResponse subscribeResponse
    = jobPort.subscribe(subscriptionRequest);
subscriptionEndpoint = subscribeResponse.getSubscriptionReference();

```

2.11. Releasing any state holds (if necessary)

```
jobPort.release(new ReleaseInputType());
```

2.12. Destroying resources

```

/*destroy subscription resource*/
SubscriptionManager subscriptionManagerPort
    = new WSBaseNotificationServiceAddressingLocator()
        .getSubscriptionManagerPort(subscriptionEndpoint);

//set stub security parameters on subscriptionManagerPort

```

```
subscriptionManagerPort.destroy(new Destroy());
```

```
/*destroy the job resource*/  
jobPort.destroy(new Destroy());
```

Chapter 4. Tutorials

The following tutorials are available for GRAM4 developers:

- [GRAM4 Scheduler Interface Tutorial](#)¹

¹ scheduler-tutorial.html

Chapter 5. Architecture and design overview

The GRAM services in GT 4.2.1 are WSRF compliant. One of the key concepts in the WSRF¹ specification is the decoupling of a service with the public "state" of the service in the interface via the implied resource pattern². Following this concept, the data of GT 4.2.1 GRAM jobs is published as part of WSRF resources, while there is only one service to start jobs or query and monitor their state. This is different from the OGSI³ model of GT3 where each job was represented as a separate service. There is still a job factory service that can be called in order to create job instances (represented as WSRF resources). Each scheduling system that GRAM is interfaced with is represented as a separate factory resource. By making a call to the factory service while associating the call to the appropriate factory resource, the job submitting actor can create a job resource mapping to a job in the chosen scheduling system.

1. Job States

1.1. Overview

The *Managed Executable Job Service (MEJS)* relies on a state machine to handle state transitions. There are two sets of states: external and internal. The external states are those that the user gets in notifications and can be queried as a resource property. The internal states are those that are strictly used by the state machine to step through all the necessary internal tasks that need to be performed for a particular job.

The Managed Multi Job Service does not rely on a state machine, but instead makes judgements after receiving notifications from the sub-jobs about which external state it should be in. The external states for the *MMJS* are identical to the ones used by the MEJS.

1.2. External and Internal States of the Managed Job Services

1.2.1. External States of the Managed Job Services

- Unsubmitted
- StageIn
- Pending
- Active
- Suspended
- StageOut
- CleanUp
- Done

¹ <http://www.globus.org/wsrf/>

² <http://www.globus.org/wsrf/faq.php#wsrf12>

³ http://www.gridforum.org/ogsi-wg/drafts/draft-ggf-ogsi-gridservice-04_2002-10-04.pdf

- Failed
- UserTerminateDone
- UserTerminateFailed

1.2.2. Internal States of the Managed Executable Job Service

- None
- Start
- StageIn
- StageInHold
- StageInResponse
- Submit
- PendingHold
- WaitingForStateChanges
- Suspend
- Resume
- OpenStdout
- OpenStderr
- MergeStdout
- StageOut
- StageOutHold
- StageOutResponse
- CleanUp
- CleanUpHold
- FileCleanUp
- FileCleanUpResponse
- CacheCleanUp
- UserTerminate
- SystemTerminate
- FailureFileCleanUp
- FailureFileCleanUpResponse

- FailureCacheCleanUp
- FinalizeTermination
- Done
- Restart

1.3. Managed Executable Job Service Internal State

Here is a diagram illustrating the internal state transitions of the Managed Executable Job Service and how the external states are triggered within this progression:

Figure 5.1. Managed Executable Job Service Internal State Transition Diagram

Chapter 6. APIs

1. Programming Model Overview

This component consists abstractly of two interfaces: the Managed Job Factory Port Type (MJFPT) and the Managed Job Port Type (MJPT).

In actuality there are three service/resource implementations, two of which implement the basic MJPT. The first one is the service which actually talks to a particular local resource manager to execute a process on the remote computer or cluster. This one is called a *Managed Executable Job Service (MEJS)* and its resource is called the Managed Executable Job Resource (MEJR). The second is a special implementation which accepts a multi-job description, breaks the description up into single-job descriptions, and then submits each of these so-called "sub-jobs" to an MEJS. This implementation is called the *Managed Multi Job Service (MMJS)*. Its resource is called the Managed Multi-Job Resource (MMJR)

Because of the fact that these two job services use the same port type, the API for accessing both the MEJR and the MMJR are identical. The *MJFS* creates the appropriate job resource depending on the factory resource used to qualify the operation call. Most of the factory resources represent local resource managers used by the MEJS (*PBS*, *LSF*, *Condor*). There is a special Multi factory resource which represents an abstract multi-job resource manager. The appropriate *job description* type is required for the two different types of managed job.

2. Component API

Java API Documentation Links (Javadoc)

- [Client API](#)¹
- [Auto-Generated Service Stubs and Persistence Data Objects API](#)²
- [Service API](#)³
- [Utilities API](#)⁴
- [Job Monitoring API](#)⁵

C API Documentation Links

- [All C APIs](#)⁶
- [GRAM4 Client Bindings](#)⁷ [[noframes](#)⁸]
- [WS-Rendezvous Client Bindings](#)⁹ [[noframes](#)¹⁰]

¹ http://www.globus.org/api/javadoc-4.0.0/globus_wsrf_gram_client_java

² http://www.globus.org/api/javadoc-4.0.0/globus_wsrf_gram_common_java

³ http://www.globus.org/api/javadoc-4.0.0/globus_wsrf_gram_service_java

⁴ http://www.globus.org/api/javadoc-4.0.0/globus_wsrf_gram_utils_java

⁵ http://www.globus.org/api/javadoc-4.0.0/globus_wsrf_gram_job_monitoring_common_java

⁶ <http://www.globus.org/api/c-globus-4.0/>

⁷ http://www.globus.org/api/c-globus-4.0/globus_c_gram_client_bindings/html/index.html

⁸ http://www.globus.org/api/c-globus-4.0/globus_c_gram_client_bindings/html/modules.html

⁹ http://www.globus.org/api/c-globus-4.0/globus_c_rendezvous_client_bindings/html/index.html

¹⁰ http://www.globus.org/api/c-globus-4.0/globus_c_rendezvous_client_bindings/html/modules.html

- [GRAM4 Scheduler Event Generator](#)¹¹ [[noframes](#)¹²]

¹¹ http://www.globus.org/api/c-globus-4.0/globus_scheduler_event_generator/html/index.html

¹² http://www.globus.org/api/c-globus-4.0/globus_scheduler_event_generator/html/modules.html

Chapter 7. Services and WSDL

1. Protocol overview

GRAM4 allows for remote execution and management of programs through the creation of a managed job. The management of the job is taken care of primarily by core toolkit functionality (WS-ResourceLifetime and WS-BaseN implementations). Please see [Java WS Core](#) on notifications and resource lifetime (destruction) for more information.

1.1. Managed Job Factory Service (MJFS)

A single MJFS is used to create all jobs for all users. For each local resource manager, a dedicated Managed Job Factory Resource (MJFR) enables the MJFS to publish information about the characteristics of the compute resource, for example:

- host information
- GridFTP URL (for file staging and streaming)
- compute cluster size and configuration, and so on...

In addition, there is a special MJFR which is used for creating MMJRs.

1.2. Managed Executable Job Service (MEJS)

A single *MEJS* is used to manage all executable jobs for all users. Each Managed Executable Job Resource (MEJR) enables the MEJS to publish information about the individual job the MEJR represents. This information can be accessed by querying the MEJS for the resource properties of a given MEJR, such as the:

- current job state
- stdout location
- stderr location
- exit code, and so on.

1.3. Managed Multi Job Service (MMJS)

A single MMJS is used to manage all multi-jobs for all users. Each Managed Multi-Job Resource (MMJR) enables the MMJS to publish information about the individual multi-job the MMJR represents. This information can be accessed by querying the MMJS for the resource properties of a given MMJR, such as the:

- current overall job state
- list of sub-job EPRs

2. Operations

There are just two operations defined in the GRAM port types (not counting the Rendezvous port type which is used for MPI job synchronization): "createManagedJob" in the Managed Job Factory port type, and "release" in the Managed

Job port type. All other operations (such as canceling/killing the job and querying for resource properties) are provided by the underlying WSRF implementation of the toolkit.

2.1. ManagedJobFactoryPortType

- `createManagedJob`: This operation creates either a MEJR or MMJR, subscribes the client for notifications if requested, and replies with one or two endpoint references (EPRs). The input of this operation consists of a *job description*, an optional initial termination time for the job resource, and an optional state notification subscription request.

The first EPR:

- is qualified with the identifier to the newly created MEJR or MMJR
- points to either the MEJS or MMJS.

The second EPR:

- is only present if a notification subscription was requested
- is qualified with the identifier to the newly created subscription resource
- points to the subscription manager service.

Using the optional subscription request provides an efficient means of subscribing to the newly created MEJR or MMJR without additional round-trip messages. Clients who subscribe afterwards must check the current status of the job, since the inherent race-condition means some state-changes may have occurred prior to the separate subscription request. In any event, there is a slight risk of lost notifications due to the lack of reliability guarantees in the notification delivery mechanism from WS-BaseNotification.

The `ManagedJobFactoryPortType` also has all the operations and publishes all the resource properties (via the MJFR) defined in the following `WS-ResourceProperties`¹ port types:

- `GetResourceProperty`
- `GetMultipleResourceProperties`
- `QueryResourceProperties`

2.2. ManagedJobPortType

This port type does not define any new operations itself, but has all the operations and publishes all the resource properties defined in the following port types:

ReleaseManagedJob port type:

- `release`: This operation takes no parameters and returns nothing. Its purpose is to release a hold placed on a state through the use of the "holdState" field in the job description. See the [do-main-specific GRAM4 component documentation](#)² for more information on the "holdState" field.

TerminateManagedJob port type:

¹ <http://www.ibm.com/developerworks/library/ws-resource/ws-resourceproperties.pdf>

² [../schemas/gram_job_description.html](#)

- `terminate`: This operation terminates a job. Depending on arguments and the state of the job this may result in immediate destruction of the job resource or in starting of clean up steps and resource destruction after the clean up is done.

*WS-ResourceProperties*³ port types:

- `GetResourceProperty`
- `GetMultipleResourceProperties`
- `QueryResourceProperties`

*WS-ResourceLifetime*⁴ port types:

- `ScheduledResourceTermination`

*WS-BaseNotification*⁵ port type:

- `NotificationProducer`

2.3. Managed Executable Job Port Type

This port type does not define any new operations. See "Resources Properties" under [Services and WSDL](#).

2.4. Managed Multi-Job Port Type

This port type does not define any new operations. See "Resources properties" below.

3. Resource properties

3.1. Managed Job Factory Port Type

- `{http://www.globus.org/namespaces/2008/03/gram/job}condorArchitecture`
Condor architecture label.
- `{http://www.globus.org/namespaces/2008/03/gram/job}condorOS`
Condor OS label.
- `{http://www.globus.org/namespaces/2008/03/gram/job}delegationFactoryEndpoint`
The endpoint reference to the delegation factory used to delegated credentials to the job.
- `{http://mds.globus.org/glue/ce/1.1}GLUECE`
GLUE data
- `{http://mds.globus.org/glue/ce/1.1}GLUECESummary`
GLUE data summary

³ <http://www.ibm.com/developerworks/library/ws-resource/ws-resourceproperties.pdf>

⁴ <http://www.ibm.com/developerworks/library/ws-resource/ws-resourcelifetime.pdf>

⁵ <ftp://www6.software.ibm.com/software/developer/library/ws-notification/WS-BaseN.pdf>

- `{http://www.globus.org/namespaces/2008/03/gram/job}globusLocation`
The location of the Globus Toolkit installation that these services are running under.
- `{http://www.globus.org/namespaces/2008/03/gram/job}hostCPUType`
The job host CPU architecture (i686, x86_64, etc...)
- `{http://www.globus.org/namespaces/2008/03/gram/job}hostManufacturer`
The host manufacturer name. May be "unknown".
- `{http://www.globus.org/namespaces/2008/03/gram/job}hostOSName`
The host OS name (Linux, Solaris, etc...)
- `{http://www.globus.org/namespaces/2008/03/gram/job}hostOSVersion`
The host OS version.
- `{http://www.globus.org/namespaces/2008/03/gram/job}localResourceManager`
The local resource manager type (i.e. *Condor*, Fork, *LSF*, Multi, *PBS*, etc...)
- `{http://www.globus.org/namespaces/2008/03/gram/job}availableLocalResourceManager`
All local resource managers that are configured in this GRAM4 instance
- `{http://www.globus.org/namespaces/2008/03/gram/job}jobTTLAfterProcessing`
Time in seconds a job resource will stay alive after a job finished processing in GRAM4 (including fileStageOut, fileCleanUp). When this time elapsed the job resource is destroyed and no longer be available for a client. A negative values means that the job resource will never be destroyed.
- `{http://www.globus.org/namespaces/2008/03/gram/job}maxJobLifetime`
Max time in seconds a user can set as initial lifetime in job submission or in subsequent setTerminationTime calls. A negative value means that there is no limit.
- `{http://mds.globus.org/metadata/2005/02}ServiceMetaDataInfo`
service start time, Globus Toolkit(R) version, service type name
- `{http://www.globus.org/namespaces/2008/03/gram/job}scratchBaseDirectory`
The directory recommended by the system administrator to be used for temporary job data.
- `{http://www.globus.org/namespaces/2008/03/gram/job}stagingDelegationFactoryEndpoint`
The endpoint reference to the delegation factory used to delegated credentials to the staging service (RFT).

3.2. Managed Job Port Type

- `{http://www.globus.org/namespaces/2008/04/rendezvous}Capacity`

Used for Rendezvous.

- {http://docs.oasis-open.org/wsrf/r1-2}currentTime
Time of creation.
- {http://docs.oasis-open.org/wsrf/rp-2}QueryExpressionDialect
From the QueryResourceProperties port type.
- {http://www.globus.org/namespaces/2008/03/gram/job/faults}fault
Faults (if generated) that happen along job processing and that cause a job to fail.
- {http://www.globus.org/namespaces/2008/03/gram/job/types}holding
Indicates whether a hold has been placed on this job.
- {http://www.globus.org/namespaces/2008/03/gram/job/types}localUserId
The job owner's local user account name.
- {http://www.globus.org/namespaces/2008/04/rendezvous}RegistrantData
Used for Rendezvous.
- {http://www.globus.org/namespaces/2008/04/rendezvous}RendezvousCompleted
Used for Rendezvous.
- {http://www.globus.org/namespaces/2008/03/gram/job/description}service-
LevelAgreement
A wrapper around fields containing the single-job and multi-job descriptions or *RSLs*. Only one of these sub-fields shall have a non-null value.
- {http://www.globus.org/namespaces/2008/03/gram/job/types}state
The current state of the job.
- {http://docs.oasis-open.org/wsrf/r1-2}TerminationTime
Time when the resource expires.
- {http://www.globus.org/namespaces/2008/03/gram/job/types}userSubject
The GSI certificate DN of the job owner.

3.3. Managed Executable Job Port Type

- {http://docs.oasis-open.org/wsrf/r1-2}currentTime
Time of creation.
- {http://docs.oasis-open.org/wsrf/r1-2}TerminationTime
Time when the resource expires.

- `{http://www.globus.org/namespaces/2008/03/gram/job/exec}credentialPath`
The path (relative to the job process) to the file containing the user proxy used by the job to authenticate out to other services.
- `{http://www.globus.org/namespaces/2008/03/gram/job/types}exitCode`
The exit code generated by the job process.
- `{http://www.globus.org/namespaces/2008/03/gram/job/faults}fault`
The fault (if generated) indicating the reason for failure of the job to complete.
- `{http://www.globus.org/namespaces/2008/03/gram/job/types}holding`
Indicates whether a hold has been placed on this job.
- `{http://www.globus.org/namespaces/2008/03/gram/job/types}localUserId`
The job owner's local user account name.
- `{http://www.globus.org/namespaces/2008/03/gram/job/exec}localJobId`
The job id(s) of the job in the local resource manager. Note that for Fork jobs these id's are prefixed with the uuid of the job.
- `{http://www.globus.org/namespaces/2008/03/gram/job/description}service-LevelAgreement`
A wrapper around fields containing the single-job and multi-job descriptions or *RSLs*. Only one of these sub-fields shall have a non-null value.
- `{http://www.globus.org/namespaces/2008/03/gram/job/types}state`
The current state of the job.
- `{http://www.globus.org/namespaces/2008/03/gram/job/exec}stderrURL`
A GridFTP URL to the file generated by the job which contains the stderr.
- `{http://www.globus.org/namespaces/2008/03/gram/job/exec}stdoutURL`
A GridFTP URL to the file generated by the job which contains the stdout.
- `{http://www.globus.org/namespaces/2008/03/gram/job/types}userSubject`
The GSI certificate DN of the job owner.
- `{http://www.globus.org/namespaces/2008/04/rendezvous}Capacity`
Used for Rendezvous.
- `{http://www.globus.org/namespaces/2008/04/rendezvous}RegistrantData`
Used for Rendezvous.
- `{http://www.globus.org/namespaces/2008/04/rendezvous}RendezvousCompleted`

Used for Rendezvous.

- `{http://docs.oasis-open.org/wsrf/rp-2}QueryExpressionDialect`

From the QueryResourceProperties port type.

3.4. Managed Multi-Job Port Type

- `{http://docs.oasis-open.org/wsrf/r1-2}CurrentTime`

Time of creation.

- `{http://docs.oasis-open.org/wsrf/r1-2}TerminationTime`

Time when the resource expires.

- `{http://www.globus.org/namespaces/2008/03/gram/job/faults}fault`

The fault (if generated) indicating the reason for failure of the job to complete.

- `{http://www.globus.org/namespaces/2008/03/gram/job/types}holding`

Indicates whether a hold has been placed on this job.

- `{http://www.globus.org/namespaces/2008/03/gram/job/types}localUserId`

The job owner's local user account name.

- `{http://www.globus.org/namespaces/2008/03/gram/job/description}service-LevelAgreement`

A wrapper around fields containing the single-job and multi-job descriptions or *RSLs*. Only one of these sub-fields shall have a non-null value.

- `{http://www.globus.org/namespaces/2008/03/gram/job/types}state`

The current state of the job.

- `{http://www.globus.org/namespaces/2008/03/gram/job/multi}subJobEndpoint`

A set of endpoint references to the sub-jobs created by this multi-job.

- `{http://www.globus.org/namespaces/2008/03/gram/job/types}userSubject`

The GSI certificate DN of the job owner.

- `{http://www.globus.org/namespaces/2008/04/rendezvous}Capacity`

Used for Rendezvous.

- `{http://www.globus.org/namespaces/2008/04/rendezvous}RegistrantData`

Used for Rendezvous.

- `{http://www.globus.org/namespaces/2008/04/rendezvous}RendezvousCompleted`

Used for Rendezvous.

- `{http://docs.oasis-open.org/wsrf/rp-2}QueryExpressionDialect`

From the QueryResourceProperties port type.

4. WSDL and Schema Definition

WSDL links:

- [ManagedJobFactoryPortType](#)⁶
- [ManagedJobPortType](#)⁷
- [ReleaseManagedJobPortType](#)⁸
- [TerminateManagedJobPortType](#)⁹
- [ManagedExecutableJobPortType](#)¹⁰
- [ManagedMultiJobPortType](#)¹¹

Schema links:

- [CAS Types](#)¹²
- [File System Mapping Config Schema](#)¹³
- GLUE Schema
 - [Batch Providers](#)¹⁴
 - [Compute Element](#)¹⁵
 - [Metadata](#)¹⁶
- [Job Description Schema](#)¹⁷
- [Managed Job Faults Schema](#)¹⁸
- [Managed Job Types Schema](#)¹⁹
- [RFT Types Schema](#)²⁰

⁶ http://viewcvs.globus.org/viewcvs.cgi/ws-gram/common/schema/gram/jdd/managed_job_factory_port_type.wsdl?revision=1.6

⁷ http://viewcvs.globus.org/viewcvs.cgi/ws-gram/common/schema/gram/jdd/managed_job_port_type.wsdl?revision=1.9

⁸ http://viewcvs.globus.org/viewcvs.cgi/ws-gram/common/schema/gram/jdd/release_managed_job_provider_port_type.wsdl?revision=1.2

⁹ http://viewcvs.globus.org/viewcvs.cgi/ws-gram/common/schema/gram/jdd/terminate_managed_job_provider_port_type.wsdl?revision=1.3

¹⁰ http://viewcvs.globus.org/viewcvs.cgi/ws-gram/common/schema/gram/jdd/managed_executable_job_port_type.wsdl?revision=1.9

¹¹ http://viewcvs.globus.org/viewcvs.cgi/ws-gram/common/schema/gram/jdd/managed_multi_job_port_type.wsdl?revision=1.9

¹² [../schemas/cas_types.html](#)

¹³ [../schemas/gram_fs_map.html](#)

¹⁴ [../schemas/batchproviders.html](#)

¹⁵ [../schemas/glue_ce.html](#)

¹⁶ [../schemas/metadata.html](#)

¹⁷ [../schemas/gram_job_description.html](#)

¹⁸ [../schemas/mj_faults.html](#)

¹⁹ [../schemas/mj_types.html](#)

²⁰ [../schemas/rft_types.html](#)

- [WS Addressing Schema](#)²¹
- [WS Base Faults Schema](#)²²

²¹ ../schemas/ws_addressing.html
²² ../schemas/ws_base_faults.html

Chapter 8. Debugging

Log output from GRAM4 is a useful tool for debugging issues. Because GRAM4 is built on top of Java WS Core, developer debugging is the same as described in [Chapter 10, Debugging](#). For sys admin logging information, see [Chapter 10, Admin Debugging](#).

1. Development Logging in Java WS Core

The following information applies to Java WS Core and those services built on it.

Logging in the Java WS Core is based on the [Jakarta Commons Logging](#)¹ API. Commons Logging provides a consistent interface for instrumenting source code while at the same time allowing the user to plug-in a different logging implementation. Currently we use [Log4j](#)² as a logging implementation. Log4j uses a separate configuration file to configure itself. Please see Log4j documentation for details on the [configuration file format](#)³.

1.1. Configuring server side developer logs

Server side logging can be configured in `$GLOBUS_LOCATION/container-log4j.properties`, when the container is stand alone container. For tomcat level logging, refer to [Logging for Tomcat](#)⁴, . The logger `log4j.appender.A1` is used for developer logging and by default writes output to the system output. By default it is set for all warnings in the Globus Toolkit package to be displayed.

Additional logging can be enabled for a package by adding a new line to the configuration file. Example:

```
#for debug level logging from org.globus.package.FooClass
log4j.category.org.globus.package.name.FooClass=DEBUG
#for warnings from org.some.warn.package
log4j.category.org.some.warn.package=WARN
```

1.2. Configuring client side developer logs

Client side logging can be configured in `$GLOBUS_LOCATION/log4j.properties`. The logger `log4j.appender.A1` is used for developer logging and by default writes output to the system output. By default it is set for all warnings in the Globus Toolkit package to be displayed.

2. Enabling debug logging for GRAM classes

To turn on debug logging for the [Managed Executable Job Service \(MEJS\)](#), add the following entry to the `container-log4j.properties` file:

```
log4j.category.org.globus.exec.service.exec=DEBUG
```

¹ <http://jakarta.apache.org/commons/logging/>

² <http://logging.apache.org/log4j/>

³ [http://logging.apache.org/log4j/docs/api/org/apache/log4j/PropertyConfigurator.html#doConfigure\(java.lang.String,org.apache.log4j.spi.LoggerRepository\)](http://logging.apache.org/log4j/docs/api/org/apache/log4j/PropertyConfigurator.html#doConfigure(java.lang.String,org.apache.log4j.spi.LoggerRepository))

⁴ <http://tomcat.apache.org/tomcat-5.5-doc/logging.html>

To turn on debug logging for the delegated proxy management code, add the following entry to the `container-log4j.properties` file:

```
log4j.category.org.globus.exec.service.utils=DEBUG
```

To turn on debug logging for the *Managed Multi Job Service (MMJS)*, add the following entry to the `container-log4j.properties` file:

```
log4j.category.org.globus.exec.service.multi=DEBUG
```

To turn on debug logging for the *Managed Job Factory Service (MJFS)*, add the following entry to the `container-log4j.properties` file:

```
log4j.category.org.globus.exec.service.factory=DEBUG
```

To turn on debug logging for all GRAM code, add the following entry to the `container-log4j.properties` file:

```
log4j.category.org.globus.exec=DEBUG
```

Follow the pattern to turn on logging for other specific packages or classes.

3. Instrumented timings logging

Both the service and Java client API code contain special debugging statements which output certain timing data to help in determining performance bottlenecks.

The service code uses the `PerformanceLog` class to output the timings information. To turn on service timings logging without triggering full debug logging for the service code, add the following lines to the `container-log4j.properties` file:

```
log4j.category.org.globus.exec.service.factory.ManagedJobFactoryService.performance=DEBUG
log4j.category.org.globus.exec.service.exec.ManagedExecutableJobResource.performance=DEBUG
log4j.category.org.globus.exec.service.exec.StateMachine.performance=DEBUG
```

The Java client API has not been converted over to using the `PerformanceLog` class, so the debug statements are sent at the `INFO` level to avoid having to turn on full debug logging. To turn on client timings logging without triggering full debug logging for the client code, add the following line to the `container-log4j.properties` file:

```
log4j.category.org.globus.exec.client.e=INFO
```

There are two parsing scripts available in the source distribution that aren't distributed in any GPT package for summarizing the service and client timings data. They are located in `ws-gram/service/java/test/throughput/`, and are named `parse-service-timings.pl` and `parse-client-timings.pl`. They both simply take the path of the appropriate log file that contains the timing data. These scripts work fine with log files that have other logging statements mixed with the timing data.

4. Debugging script execution

It may be necessary to debug the *scheduler* scripts if jobs aren't being submitted correctly, and either no fault or a less-than-helpful fault is generated. Ideally we would like that this not be necessary; so if you find that you must resort to this, please file a bug report or let us know on the discuss e-mail list.

By turning on debug logging for the MEJS (see above), you should be able to search for "*Perl Job Description*" in the logging output to find the perl form of the *job description* that is sent to the scheduler scripts.

Also by turning on debug logging for the MEJS, you should be able to search for "*Executing command*" in the logging output to find the specific commands that are executed when the scheduler scripts are invoked from the service code. If you saved the perl job description from the previous paragraph, then you can use this to manually run these commands.

There is a perl job description attribute named `logfile` that isn't currently supported in the XML job description that can be used to print debugging info about the execution of the perl scripts. The value for this attribute is a path to a file that will be created. You can add this to the perl job description file that you created from the service debug logging before manually running the script commands.

Beyond the above advice, you may want to edit the perl scripts themselves to print more detailed information. For more information on the location and composition of the scheduler scripts, please consult the [GRAM4 Scheduler Interface Tutorial](#)⁵.

⁵ ../developer/scheduler-tutorial-perl.html

Chapter 9. Troubleshooting

For a list of common errors in GT, see [Error Codes](#).

For information about sys admin logging, see [Chapter 10, Admin Debugging](#) in the GRAM4 Admin Guide.

1. Troubleshooting tips

In case you run into problems you can do the following

- Check the GRAM4 documentation. Maybe you'll find hints here to solve your problem.
- Send e-mails to one of several Globus e-mail lists. You'll have to subscribe to a list before you can send an e-mail to it. See [here](#)¹ for general e-mail lists and information on how to subscribe to a list and [here](#)² for GRAM specific lists.

Probably the best lists for GRAM4-related problems are `gt-user@globus.org` and `gram-user@globus.org`

- Check the container log for errors.

In case you don't find anything suspicious you can increase the log-level of GRAM4 or other relevant components. Maybe the additional logging-information will tell you what's going wrong. General information about container logging can be found [Logging in Java WS Core](#) section.

To get debug information from GRAM4, un-comment the following line in `$GLOBUS_LOCATION/container-log4j.properties` by removing the leading '#' and restart the GT4 server.

```
# log4j.category.org.globus.exec=DEBUG
```

The logging output can either be found on the console if you started the container using `globus-start-container` (maybe with arguments) or in `$GLOBUS_LOCATION/var/container.log` in if you started the container using the command `globus-start-container-detached`

¹ http://dev.globus.org/wiki/Mailing_Lists

² http://dev.globus.org/wiki/GRAM#Mailing_Lists

2. Java WS Core Errors

Table 9.1. Java WS Core Errors

Error Code	Definition
Failed to acquire notification consumer home instance from registry	Caused by <code>javax.naming.NameNotFoundException</code> : Name <code>services</code> is not bound in
The WS-Addressing 'To' request header is missing	This warning is logged by the container if the request did not contain the necessary <i>WS-Addressing</i> headers, those headers at all or is somehow misconfigured.
java.io.IOException: Token length X > 33554432	If you see this error in the container log, it usually means you are trying to connect to HTTPS server using <code>https</code> specifies 8443 as a port number and <code>http</code> as the protocol name.
java.lang.NoSuchFieldError: DOCUMENT	This error usually indicates a mismatch between the version of Apache Axis that the code was compiled with and the version currently running with.
org.globus.wsrfl.InvalidResourceKeyException: Argument key is null / Resource key is missing	These errors usually indicate that a resource key was not passed with the request or that an invalid resource key was used (the element <code>QName</code> of the resource key did not match what the service expected).
Unable to connect to localhost:xxx	Cannot resolve localhost. The machine's <code>/etc/hosts</code> isn't set up correctly and/or you do not have DNS for
org.globus.common.ChainedIOException: Failed to initialize security context	This may indicate that the user's proxy is invalid.
Error: org.xml.sax.SAXException: Unregistered type: class xxx	This may indicate that an Axis generated XML type, defined by the WS RLS XSD, was not properly registered upon deployment without intervention by the user, sometimes they do not.
No socket factory for 'https' protocol	<p>When a client fails with the following exception:</p> <pre>java.io.IOException: No socket factory for 'https' protocol at org.apache.axis.transport.http.HTTPSender.getSocket(HTTPSender.java:100) org.apache.axis.transport.http.HTTPSender.writeToSocket(HTTPSender.java:110) org.apache.axis.transport.http.HTTPSender.invoke(HTTPSender.java:120)</pre> <p>FIXME - it may have happened because...</p>

Error Code	Definition
No client transport named 'https' found	<p>When a client fails with the following exception:</p> <pre>No client transport named 'https' found at org.apache.axis.client.AxisClient.invoke(AxisClient.java:170) at org.apache.axis.client.Call.invokeEngine(Call.java:2726)</pre> <p>The client is most likely loading an incorrect <code>client-config.wsdd</code> configuration file.</p>
ConcurrentModificationException in Tomcat 5.0.x	<p>If the following exception is visible in the Tomcat logs at startup, it might cause the HTTPSValve to fail:</p> <pre>java.util.ConcurrentModificationException at java.util.HashMap\$HashIterator.nextEntry(HashMap.java:782) at java.util.HashMap\$EntryIterator.next(HashMap.java:824) at java.util.HashMap.putAllForCreate(HashMap.java:424) at java.util.HashMap.clone(HashMap.java:656) at mx4j.server.DefaultMBeanRepository.clone(DefaultMBeanRepository.</pre> <p>The HTTPSValve might fail with the following exception:</p> <pre>java.lang.NullPointerException at org.apache.coyote.tomcat5.CoyoteRequest.setAttribute(CoyoteRequestFacade.java:100) at org.apache.coyote.tomcat5.CoyoteRequestFacade.setAttribute(CoyoteRequestFacade.java:100) at org.globus.tomcat.coyote.valves.HTTPSValve.expose(HTTPSVAlve.java:100)</pre> <p>These exceptions will prevent the transport security from working properly in Tomcat.</p>
java.net.SocketException: Invalid argument or cannot assign requested address	<p>FIXME - what causes this?</p>
GAR deploy/undeploy fails with container is running error	<p>A GAR file can only be deployed or undeployed locally while the container is off. However, GAR deployment fails with this error even if the container is off. This usually happens if the container has crashed or was stopped from cleaning up its state files.</p>

3. Errors

Table 9.2. GRAM4 Errors

Error Code	Definition	Possible Solutions
<p>globusrun-ws - error querying job state</p>	<p>During job submission, an error like this occurs: globusrun-ws failed: Delegating user credentials...Done. Submitting job...Done. Job ID: xxxx Termination time: xxxx Current job state: Unsubmitted globusrun-ws: Error querying job state globus_soap_message_module: Failed sending request ManagedJobPortType_GetMultipleResourceProperties. globus_xio: An end of file occurred</p>	<p>Periodically, globusrun-ws will query the GRAM service to check on the job state. The "End of file" indicates that the GRAM server dropped a connection when globusrun-ws tried to read a response. This could be caused by temporary network issues between the client and service, or possibly caused by an overloaded service host.</p>
<p>globusrun-ws - error querying job state</p>	<p>During job submission, an error like this occurs: globusrun-ws failed: Delegating user credentials...Done. Submitting job...Done. Job ID: xxxx Termination time: xxxx Current job state: Unsubmitted globusrun-ws: Error querying job state globus_soap_message_module: Failed sending request ManagedJobPortType_GetMultipleResourceProperties. globus_xio: System error in read: Connection reset by peer globus_xio: A system call failed: Connection reset by peer</p>	<p>Periodically, globusrun-ws will query the GRAM service to check on the job state. The System error in read: Connection reset by peer indicates that the GRAM server dropped the connection while trying to write the response. This could be caused by temporary network issues between the client and service, or possibly caused by an overloaded service host.</p>

Error Code	Definition	Possible Solutions
glo- bus- run- ws - error sub- mit- ting job	During job submission, an error like this occurs: globusrun-ws -Ft PBS -F ht- tps://host.teragrid.org:8444 -submit -b -f /tmp/wsgram.rsl -o /tmp/wsgram.epr failed: Submitting job...Failed. globusrun-ws: Error submitting job globus_soap_message_module: Failed sending request ManagedJobFactoryPortType_createManagedJob. globus_xio: Operation was canceled globus_xio: Operation timed out	The Operation timed out indicates that the GRAM service was not able to accept the job request and respond in time. This could be caused by temporary network issues between the client and service, or possibly caused by an overloaded service host.

Chapter 10. Related Documentation

No related documentation links have been determined at this time.

Chapter 11. Internal Components

Internal Components¹

¹ [internal-components.html](#)

Glossary

C

Condor A job scheduler mechanism supported by GRAM. See <http://www.cs.wisc.edu/condor/> for more information.

J

job description Term used to describe a GRAM4 job for GT4.

L

LSF A job scheduler mechanism supported by GRAM.
For more information, see <http://www.platform.com/Products/Platform.LSF.Family/Platform.LSF/>⁷.

M

Managed Executable Job Service (MEJS) [FIXME]

Managed Job Factory Service (MJFS) [FIXME]

Managed Multi Job Service (MMJS) [FIXME]

multijob A job that is itself composed of several executable jobs; these are processed by the MMJS subjob.
See also [MMJS subjob](#)¹⁰.

P

Portable Batch System (PBS) A job scheduler mechanism supported by GRAM. For more information, see <http://www.openpbs.org>.

R

Resource Specification Language (RSL) Term used to describe a GRAM job for GT2 and GT3. (Note: This is not the same as RLS - the Replica Location Service)

⁷ <http://www.platform.com/Products/Platform.LSF.Family/Platform.LSF/>
¹⁰ #mmjs-subjob

S

scheduler

Term used to describe a job scheduler mechanism to which GRAM interfaces. It is a networked system for submitting, controlling, and monitoring the workload of batch jobs in one or more computers. The jobs or tasks are scheduled for execution at a time chosen by the subsystem according to an available policy and availability of resources. Popular job schedulers include Portable Batch System (PBS), Platform LSF, and IBM LoadLeveler.

W

Web Services Addressing
(WSA)

The WS-Addressing specification defines transport-neutral mechanisms to address web services and messages. Specifically, it defines XML elements to identify web service endpoints and to secure end-to-end endpoint identification in messages. See the W3C WS Addressing Working Group¹⁴ for details.

¹⁴ <http://www.w3.org/2002/ws/addr/>

Index

A

- apis, 27
 - links, 27
 - overview, 27

C

- compatibility, 2
- containers, tested, 1

D

- debugging, 38
 - logging, 38
- dependencies, 2

E

- errors, 42, 45

F

- features, 1

L

- logging
 - debugging, 38

P

- platforms, tested, 1

R

- resource properties, 31
 - Managed Executable Job Port Type, 33
 - Managed Job Factory Port Type, 31
 - Managed Job Port Type, 32
 - Managed Multi-Job Port Type, 35

S

- services, 29

T

- troubleshooting, 41
 - check container log, 41
 - check documentation, 41
 - errors, 41
 - mailing lists, 41

W

- WSDL, 29

GT 4.2.1 Migrating Guide for GRAM4

GT 4.2.1 Migrating Guide for GRAM4

Abstract

The following provides available information about migrating from previous versions of the Globus Toolkit.

Table of Contents

1. Migrating GRAM from GT4.0	1
1. Admin - Migration Guide	1
2. User - Migration Guide	1
3. Developer - Migration Guide	3
2. Migrating GRAM from GT3	7
3. Migrating GRAM from GT2	9
1. Admin - Migration Guide	9
2. User - Migration Guide	11
3. Developer - API and RSL Migration Guide	12
Glossary	21

List of Tables

3.1. Command Line Option Comparison	12
3.2. C API Migration Table	13
3.3. RSL Migration Table	15
3.4. RSL Migration Examples	17

Chapter 1. Migrating GRAM from GT4.0

The 4.2.1 protocol has been changed to be WSRF 1.2, WSN 1.3 and WS Addressing 1.0 compliant. There is no backward compatibility between 4.2.1 and 4.0.

1. Admin - Migration Guide

1.1. Default local resource manager

In GRAM4 an administrator can configure a default local resource manager being used for job submission if the client did not specify any in a job submission request. Check section [Defining a default local resource manager](#) in the [System Administrator's Guide](#) for more information.

1.2. Audit Logging

The log4j configuration for audit logging in GRAM4 in 4.2 is slightly different compared to the 4.0 series. This involves log4j and database configuration changes. For the log4j configuration check section [Configure Log4j](#) in the [System Administrator's Guide](#) for more information. For the database related changes check [Configure the Database in JNDI](#) in the [System Administrator's Guide](#).

1.3. Job lifetime

An administrator can configure lifetime parameters for all jobs in GRAM4's JNDI configuration. In short these parameters restrict the maximum lifetime a client can set on a job, and the maximum time a job is kept in the container and thus in the persistence directory after the job had been fully processed. Check section [Job Lifetime](#) in the [Execution key concepts](#) and section [Job lifetime configuration](#) in the [System Administrator's Guide](#) for more information.

1.4. Local Java calls from GRAM4 to RFT

By default GRAM4 in the 4.0 series does Web Service invocations when calling RFT to perform staging and file cleanup. The default in GRAM4 in 4.2 is local java calls to significantly improve performance in jobs with staging and file cleanup. If GRAM4 is configured to use RFT in a separate machine local invocations are disabled. For more information check the section [GRAM4 - RFT interaction](#) in the [System Administrator's Guide](#)

1.5. Threads for SEG event processing

An administrator can configure the number of threads in the thread-pool that is responsible for handling scheduler events. The default size should be fine for all Scheduler Event Generators (SEGs) provided by Globus. It might be interesting for custom written SEGs. Check section [Configuring thread-pools](#) in the [System Administrator's Guide](#) for more information.

2. User - Migration Guide

2.1. Specification upgrade related changes

Due to the WSRF, WSN and WS-Addressing specification upgrade some namespaces changed. All occurrences of the following old 4.0 namespaces must be updated to the new 4.2 namespaces.

4.0	4.2
http://schemas.xmlsoap.org/ws/2004/03/addressing	http://www.w3.org/2005/08/addressing
http://www.globus.org/namespaces/2004/10/gram/job	http://www.globus.org/namespaces/2008/03/gram/job
http://www.globus.org/namespaces/2004/10/rft	http://www.globus.org/namespaces/2008/04/rft

Additionally, in a job description the ReferenceProperties element must be renamed to ReferenceParameters.

The following shows a factoryEndpoint element of a 4.0 job description and a 4.2 job description:

```
# Endpoint in 4.0
<factoryEndpoint
  xmlns:gram="http://www.globus.org/namespaces/2004/10/gram/job"
  xmlns:wsa="http://schemas.xmlsoap.org/ws/2004/03/addressing">
  <wsa:Address>
    https://grid-w.ncsa.teragrid.org:8444/wsrp/services/ManagedJobFactoryService
  </wsa:Address>
  <wsa:ReferenceProperties>
    <gram:ResourceID>Fork</gram:ResourceID>
  </wsa:ReferenceProperties>
</factoryEndpoint>

# Endpoint in 4.2
<factoryEndpoint
  xmlns:gram="http://www.globus.org/namespaces/2008/03/gram/job"
  xmlns:wsa="http://www.w3.org/2005/08/addressing">
  <wsa:Address>
    https://grid-w.ncsa.teragrid.org:8444/wsrp/services/ManagedJobFactoryService
  </wsa:Address>
  <wsa:ReferenceParameters>
    <gram:ResourceID>Fork</gram:ResourceID>
  </wsa:ReferenceParameters>
</factoryEndpoint>
```

2.2. Local resource manager

GRAM4 in 4.2 defines a default local resource manager which is used for job execution if the user did not explicitly specify one. A user however still can specify a local resource manager (LRM), e.g. if it's required that a job runs in a certain LRM. Check

- [Finding available local resource managers](#) to get information about how to get a list of available LRM's of a GRAM4 instance.
- [Finding the default local resource manager](#) to find out which LRM is the default LRM of a GRAM4 instance.
- [Specifying a local resource manager](#) to get information about how to submit to the default vs to a non-default LRM.

2.3. Job Lifetime

globusrun-ws in 4.0 sets a default lifetime of 24h on a job if a user does not explicitly set it. In 4.2 a job will run to completion in any event if no lifetime is specified, i.e. it will not be terminated after 24h.

However, there are server-side settings define limits on an explicitly requested lifetime, and that determine what will happen to a completed job if no lifetime had been set. Check section [Job Lifetime](#) in the [Execution key concepts](#) and section [Job Lifetime](#) in the [User's Guide](#) for more information.

3. Developer - Migration Guide

3.1. Specification upgrade related changes

Due to the WSRF, WSN and WS-Addressing specification upgrade namespaces of many components have changed. Check section [Specification upgrade related changes](#) and the [Java WS Core Migration guide](#) for details.

3.2. Job termination

GRAM4 does not extend from the ImmediateResourceTermination port type anymore, i.e. the Destroy() operation is no longer available to synchronously destroy job resources. Instead GRAM4 offers an asynchronous Terminate() method that lets clients terminate jobs and destroy a job resource. Check the TerminateManagedJobPortType PortType in the section 'WSDL and Schema Definition' of the [Developer's guide](#) for details and section [Job termination](#) for explanations of the new termination interface.

3.3. Job lifetime

The Java API GramJob in 4.0 sets a default lifetime of 24h on a job if a developer does not explicitly set it. In 4.2 it will not be set and a job will run to completion in any event if no lifetime is specified, i.e. it will not be terminated after 24h.

However, there are server-side settings define limits on an explicitly requested lifetime, and that determine what will happen to a completed job if no lifetime had been set. Check section [Job Lifetime](#) in the [Execution key concepts](#) and section [Job Lifetime](#) in the [User's Guide](#) for more information.

3.4. Subscription resources

In GRAM4 in the 4.0 series a client had to destroy subscription resources before destroying a job if it subscribed for notifications in order to cleanup state on the server-side. In GRAM4 in 4.2 all subscription resources of a job resource are destroyed on the server-side automatically if a job resource is destroyed. There's no need to keep track of subscription resources anymore.

3.5. Default local resource manager

In GRAM4 an administrator can configure a default local resource manager (LRM) being used for job submission if the client did not specify any in a job submission request. To submit a job to the default local resource manager just don't put a ReferenceParameters element in the EndpointReference used to contact GRAM4's job factory service. By querying resource properties of GRAM4's factory service a client can find out which local resource manager is the default, or which LRM's are available on the server-side.

3.6. Client-side generated UUIDS for MultiJobs

In GRAM4 in GT 4.0 the client-side generated submission being set in CreateManagedJobInputType.JobID had been used as server-side job id for multi-jobs. In GRAM4 in 4.2 a UUID is created on the server-side to manage the job and the client-side generated ID will only be stored to enable reliable job submission. A client cannot use the self-generated

id for any job operation besides a second submission call for multi-jobs anymore. See section [Client-Side Generated Submission ID](#) for more information about this.

3.7. API changes

3.7.1. Service

3.7.1.1. Namespace

All occurrences of `http://www.globus.org/namespaces/2004/10/gram/job` had been replaced by `http://www.globus.org/namespaces/2008/03/gram/job`.

3.7.1.2. ResourceProperties

- `{http://www.globus.org/namespaces/2008/03/gram/job/faults}fault` from the `ManagedJobPortType` is an array instead of a single element now. All faults that happen during job processing will be added to that array.
- New `{http://www.globus.org/namespaces/2008/03/gram/job}availableLocalResourceManager` All local resource managers that are configured in this GRAM4 instance.
- New `{http://www.globus.org/namespaces/2008/03/gram/job}jobTTLAfterProcessingTime` in seconds a job resource will stay alive after a job finished processing in GRAM4 (including `fileStageOut`, `fileCleanUp`). When this time elapsed the job resource is destroyed and no longer be available for a client. A negative values means that the job resource will never be destroyed.
- New `{http://www.globus.org/namespaces/2008/03/gram/job}maxJobLifetime` Max time in seconds a user can set as initial lifetime in job submission or in subsequent `setTerminationTime` calls. A negative value means that there is no limit.

3.7.1.3. State change notification topic

The QName of the topic that clients use to subscribe for job state change information changed to `{http://www.globus.org/namespaces/2008/03/gram/job}stateChangeNotificationMessage`. (In GRAM4 in 4.0 it was `{http://www.globus.org/namespaces/2004/10/gram/job}state`)

3.7.1.4. Job states

New values of the resource property `{http://www.globus.org/namespaces/2008/03/gram/job/types}state`: `UserTerminateDone` and `UserTerminateFailed`.

3.7.1.5. Fault types

- `{http://www.globus.org/namespaces/2008/03/gram/job/release}ResourceNotReleasedFaultType`: thrown when a resource cannot not be released but it still exists.
- `{http://www.globus.org/namespaces/2008/03/gram/job/terminate}ResourceNotTerminatedFaultType`: Thrown when a resource cannot not be terminated but still exists.
- `{http://www.globus.org/namespaces/2008/03/gram/job/faults}DelegatedCredentialDestroyFaultType`: Thrown when a job resource is terminated with request to destroy delegated credentials, and the delegated credential can't be destroyed. This fault is also added to the job RP `fault` and part of a notification message if a delegated credential cannot be destroyed in asynchronous termination.

- `{http://www.globus.org/namespaces/2008/03/gram/job/faults}JobResourceExpired-FaultType`: Added to the job RP fault and part of a notification message if a resource expired.
- `{http://www.globus.org/namespaces/2008/03/gram/job/faults}StagingTerminate-FaultType`: Added to the job RP fault and part of a notification message if termination of a running transfer at RFT as part of job termination failed.
- `{http://www.globus.org/namespaces/2008/03/gram/job/faults}LocalResourceManagerJobTerminateFaultType`: Added to the job RP fault and part of a notification message if termination of a job at the local resource manager as part of job termination failed.

3.7.1.6. Job destruction

GRAM4 does no longer extend from the `ImmediateResourceTermination` port type, i.e. does no longer offer the `Destroy` method. Instead jobs must be terminated. See above description.

3.7.2. Client

3.7.2.1. Java client API: `GramJob`

- There is a new method for job termination:

```
boolean terminate(  
    boolean destroyAfterCleanUp,  
    boolean continueNotifying,  
    boolean destroyDelegatedCredentials)  
    throws ResourceUnknownFaultType, DelegatedCredentialDestroyFaultType,  
           ResourceNotTerminatedFaultType
```

- The methods `destroy()`, `cancel()` are kept for convenience and implemented to do calls to `terminate()` with following settings: Resource destruction after cleanup, no notifications about success or failure of termination, destruction of delegated credentials if `GramJob` did the delegation. These methods are marked as deprecated and will be removed in 4.4.
- The method to set the duration of a job changed from

```
public void setDuration(Date duration)  
  
to  
  
public void setDuration(int hours, int minutes)
```

Check [bug 5806](http://bugzilla.globus.org/show_bug.cgi?id=5806)¹ for details.

- The default Axis stub timeout was extended from from 2min to 5min.
- No more default lifetime of a job: If a client does not explicitly set a lifetime `GramJob` does not set it. The job will run to completion then. If `GramJob` delegates credentials and no lifetime of a job has been specified, the lifetime of the delegated credentials will not explicitly be set but the `timeleft`-value of the proxy will be used. If a lifetime is specified for a job the lifetime of the delegated credentials will be the same like the lifetime of the job.
- The method `GramJob.getFault()` returns an array of `FaultType` objects instead of a single `FaultType` object.

¹ http://bugzilla.globus.org/show_bug.cgi?id=5806

- The method `isLocallyDestroyed()` was removed because it didn't make sense with the new job termination anymore.

3.7.2.2. C client API

Chapter 2. Migrating GRAM from GT3

Migrating to GT 4.2.1 from GT version 3.2:

- The 4.2.1 protocol has been changed to be WSRF compliant. There is no backward compatibility between 4.2.1 and 3.2.

API changes since GT 3.2:

- The *MJFS* `create` operation has become `createManagedJob` and, now provides the option to send a *uuid*. A client can use this uuid to recover a job EPR in the event that the reply message is not received. Given this new scheme, the `start` operation was removed. The `createManagedJob()` operation also allows a notification subscription request to be specified. This is the only way to reliably get all job state notifications.
- The *MJS* `start` operation has been removed. Its purpose was to ensure that the client had received the job EPR prior to the job being executed (and thus consuming resources), and is redundant with the `uuid` functionality.

New GRAM Client Submission Tool:

- *globusrun-ws* has replaced `managed-job-globusrun` as the GRAM4 client submission program. The main reason was performance. The cost of JVM startup for each job submission through `managed-job-globusrun` was too much. *globusrun-ws* is written in C and thus avoids the JVM startup cost. *globusrun-ws* is very similar in functionality to `managed-job-globusrun`, but you will need to become familiar with the arguments and options.

RSL Schema Changes Since GT 3.2:

- RSL Substitutions RSL substitution syntax has changed to allow for a simpler RSL schema that can be parsed by standard tools. In GT 3.2, applications could define arbitrary RSL substitutions within an RSL document and rely on the GRAM service to resolve them. In GT4 GRAM4, this feature is no longer present. In GT 4.2.1 there are 5 RSL variables: `${GLOBUS_USER_HOME}`, `${GLOBUS_USER_NAME}`, `${GLOBUS_SCRATCH_DIR}`, and `${GLOBUS_LOCATION}`.
- `executable` is now a single local file path. Remote URLs are no longer allowed. If executable staging is desired, it should be added to the `fileStageIn` directive.
- `stdin` is now a single local file path. Remote URLs are no longer allowed. If `stdin` staging is desired, it should be added to the `fileStageIn` directive.
- `stdout` is now a single local file path, instead of a list of remote URLs. If `stdout` staging is desired, it should be added to the `fileStageOut` directive.
- `stderr` is now a single local file path, instead of a list of remote URLs. If `stderr` staging is desired, it should be added to the `fileStageOut` directive.
- `scratchDirectory` has been removed.
- `gramMyJobType` has been removed. "Collective" functionality is always available if a job chooses to use it.
- `dryRun` has been removed. This is obsolete given the addition of the `holdState` attribute. setting `holdState` to "StageIn" should prevent the job from being submitted to the local *scheduler*. It can then be canceled once the StageIn-Hold state notification is received.
- `remoteIoUrl` has been removed. This was a hack for GRAM2 involved with staging via GASS, and has no relevancy in the current implementation.

- File Staging related RSL attributes have been replaced with RFT file transfer attributes/syntax.
- RSL substitution definitions and substitution references have been removed in order to be able to use standard XML parsing/serialization tools.
- RSL variables have been added. These are keywords denoted in the form of `${variable name}` that can be found in certain RSL attributes.
- Explicit credential references have been added, which, along with use of the new `DelegationFactory` service, replace the old implicit delegation model.

Fault changes since GT version 3.2:

- `CacheFaultType` was removed since there is no longer a GASS cache.
- `RepeatedlyStartedFaultType` was removed since there is no longer a `start` operation. Repeat creates with the same submission ID simply return the job EPR.
- `SLAFaultType` was changed to `ServiceLevelAgreementFaultType` for clarification.
- `StreamServiceCreationFaultType` was removed since there is no longer a stream service.
- `UnresolvedSubstitutionReferencesFaultType` was removed since there is no longer support for substitution definitions and references in the RSL.
- `DatabaseAccessFaultType` was removed since a database is no longer used to save job data.

Chapter 3. Migrating GRAM from GT2

1. Admin - Migration Guide

1.1. Installation / Deployment Differences

In GRAM2, jobs are submitted to a job manager process started by a Gatekeeper process. The Gatekeeper process is typically started out by an inetd server, which forks a new gatekeeper per job. In GRAM4, jobs are started by the ManagedExecutionJobService, which is a Java service implementation running within the globus service container.

The gatekeeper searches the \$GLOBUS_LOCATION/etc/grid-services directory to determine which services it will start. Typically there is one job manager service entry file in that directory per scheduler type.

1.2. Security Differences

1.2.1. Proxies and Delegation

In GRAM2, the GRAM client is required to delegate a proxy credential to the Gatekeeper so that the job manager can send authenticated job state change messages.

In GRAM4, delegation is done as needed using the DelegationFactoryService. Jobs may be passed references to delegated credentials as part of the job description.

1.2.2. Network Communication

In GRAM2, communication between the client and gatekeeper is done using GSI-wrapped messages. Communication between the the client and job manager are sent using SSL. The job manager uses the delegated credential for file streaming or staging as well. Mutual authentication is done on all connections. All communications consist of a single request-response pattern. Network connections and security contexts are never cached between messages.

In GRAM4, communication may be secured using TLS, ws secure messaging, or ws secure conversation, depending on service configuration. When doing authentication, the service will use the credentials of the container, or for secure message or conversation, a service-specific credential. It will not use a delegated credential when communicating with the client.

1.2.3. Root / Local Account Access

The gatekeeper process is started as a root service out of inetd. It then uses the grid-mapfile decide which local user it should setuid() to before starting the job manager process, based on the credential used to submit the job request. The user may optionally propose a non-default user-id by specifying it in the gatekeeper contact string. The job manager process runs entirely under the local user account.

In GRAM4, the job management service runs within a container shared with other services. The container is run under a non-privileged account. All commands which need to be run as a particular user (such as interactions with the *scheduler*) are started via *sudo*. Authorization is done via the globus-gridmap-and-execute program.

1.3. Scheduler Interaction Differences

In GRAM2, all file system and scheduler interactions occur within a perl module called by the globus-job-manager-script.pl program. Scheduler-specific perl modules implement a number of methods which are used by the job manager:

- submit
- poll
- cancel
- signal
- make_scratchdir
- remove_scratchdir
- stage_in
- stage_out
- cache_cleanup
- remote_io_file_create
- proxy_relocate
- proxy_update

Only a small set of these script methods are used in the GRAM4 implementation. The subset used is:

- submit
- poll (called only once per job and only for fork/condor jobs to merge output)
- cancel
- cache_cleanup

Some of the functionality has been moved into other services for reliability or performance reasons. Other functions have been removed altogether.

- poll: SEG
- signal: dropped
- make_scratchdir: rft
- remove_scratchdir: rft
- stage_in: rft
- stage_out: rft
- remote_io_file_create: rft or resource property queries
- proxy_relocate: delegation service
- proxy_update: delegation service

1.4. Local Node Impact

In GRAM2, each job submitted would cause the following processes to be created:

- gatekeeper (short lived)
- job manager (lives the duration of the job)
- perl script (short lived 4 or more instances depending on job type)
- perl script poll called periodically

In GRAM4, each job causes the following processes to be created

- sudo + perl script--(typically 2 times: submit, cache_cleanup)
- for fork jobs, one fork-starter process (blocked waiting for a signal) for the duration of the job

Additionally, there will be a per-scheduler instance of the *SEG* program, monitoring a log file for job state changes.

2. User - Migration Guide

2.1. Command Line Tools

Typical interactions with the GRAM2 service were done with either the *globusrun* or *command* or the *globus-job* suite of scripts (*globus-job-submit*, *globus-job-run*, *globus-job-get-output*, *globus-job-status*, *globus-job-clean*). The main difference between these sets of commands is that *globusrun* required a *job description* in *RSL* format, and the *globus-job-submit* and *globus-job-run* scripts would construct that based on command line options.

In GRAM4, the *globusrun-ws* command implements the functionality of *globusrun* using the XML Job Description language in place of the RSL format job description of GRAM2. It also allows specifying parts of the Job Description with simple command line arguments (for executable and arguments), similar to what one would do with *globus-job-run*. Like *globusrun*, the *globusrun-ws* program supports both the interactive and batch submission of GRAM jobs.

Table 3.1. Command Line Option Comparison

Description	GRAM2 globusrun option	GRAM4 globusrun-ws option
Interactive Multirequest Control	-i	NO EQUIVALENT
Job Description File Path	-f <rsl filename> -file <rsl filename>	-f <filename> -job-description-file <filename>
Quiet operation	-q -quiet	-q -quiet
File streaming of stdout and stderr *see note 1*	-o (Implies -q)	-s -streaming (Implies -q, sometimes -staging-delegate)
Enable embedded GASS Server	-s -server (Implies -o and -q)	NO EQUIVALENT
Enable writing to the embedded GASS Server	-w -write-allow (Implies -s and -q)	NO EQUIVALENT
Specify Service Contact	-r <resource-manager> -resource <resource-manager> (Specifies Gatekeeper contact)	-F, -Ft, or -Ff; Use either factory service contact (-F), Factory Type (-Ft) or Factory EPR file (-Ff)
Do not terminate job when SIGINT is received.	-n -no-interrupt	-n -no-cleanup
Destroy a job based on a job - contact	-k <job contact> -kill <job contact>	-kill -j <filename> -kill -job-epr-file <filename>
Get current job status	-status <job contact>	-status -j <filename> -status -job-epr-file <filename>
Batch mode job submission	-b -batch or -F -fast-batch	-batch -b
Refresh proxy	-refresh-proxy <job contact> -y <job contact>	NO EQUIVALENT
Stop a job manager process, saving state	-stop-manager <job contact>	NO EQUIVALENT
Validate job description without submitting job	-p -parse	-validate
Ping job manager	-a -authenticate-only	NO EQUIVALENT
Dryrun	-d -dryrun	NO EQUIVALENT

Note 1: In GRAM2, streaming is done using https connections from the job manager to a GASS server embedded in the globusrun program. In GRAM4, streaming is implemented by accessing a gridftp server configured to run along with the service container.

globusrun-ws has additional options to deal with file streaming, monitoring an existing job, authentication and authorization, http timeouts, default termination time, encryption, etc.

3. Developer - API and RSL Migration Guide

This table describes the migration path for applications which use the C language interface to GRAM2. This table covers the globus_gram_client API.

Table 3.2. C API Migration Table

GT2 API Command	GT4 API Command
globus_gram_client_callback_allow()	globus_notification_create_consumer()
globus_gram_client_register_job_request()	ManagedJobFactoryPortType_GetResourceProperty_epr_register()
globus_gram_client_job_request()	ManagedJobFactoryPortType_GetResourceProperty_epr()
globus_gram_client_register_job_cancel()	ManagedExecutableJobPortType_Destroy_epr_register()
globus_gram_client_job_cancel()	ManagedExecutableJobPortType_Destroy_epr()
globus_gram_client_job_status()	ManagedExecutableJobPortType_GetResourceProperty_epr() with the property name {http://www.globus.org/namespaces/2008/03/gram/job/types}state
globus_gram_client_register_job_status()	ManagedExecutableJobPortType_GetResourceProperty_epr_register() with the property name {http://www.globus.org/namespaces/2008/03/gram/job/types}state
globus_gram_client_job_refresh_credentials()	globus_delegation_client_util_delegate_epr
globus_gram_client_register_job_refresh_credentials()	globus_delegation_client_util_delegate_epr_register()
globus_gram_client_register_job_signal()	ManagedExecutableJobPortType_release_epr_register()
globus_gram_client_job_signal()	ManagedExecutableJobPortType_release_epr()
globus_gram_client_register_job_callback_registration()	ManagedExecutableJobPortType_Subscribe_epr_register()
globus_gram_client_job_callback_register()	ManagedExecutableJobPortType_Subscribe_epr()
globus_gram_client_register_job_callback_unregister()	SubscriptionManager_Destroy_epr_register()
globus_gram_client_job_callback_unregister()	SubscriptionManager_Destroy_epr()
globus_gram_client_callback_disallow()	globus_notification_destroy_consumer()
globus_gram_client_job_contact_free()	wsa_EndpointReferenceType_destroy()
globus_gram_client_error_string()	globus_error_get(result)
globus_gram_client_set_credentials()	globus_soap_message_handle_set_attr() with the property name GLOBUS_SOAP_MESSAGE_USER_CREDENTIAL_KEY and the value the gss_cred_id_t
globus_gram_client_ping()	XXX? Maybe factory get resource properties?
globus_gram_client_register_ping()	XXX? Maybe factory get resource properties?
globus_gram_client_debug()	set GLOBUS_SOAP_MESSAGE_DEBUG environment variable to MESSAGES to see XML messages sent/received
globus_gram_client_version()	NO EQUIVALENT
globus_gram_client_attr_init()	globus_soap_message_attr_init()
globus_gram_client_attr_destroy()	globus_soap_message_attr_destroy()
globus_gram_client_attr_set_credential()	globus_soap_message_handle_set_attr() with the property name GLOBUS_SOAP_MESSAGE_USER_CREDENTIAL_KEY and the value the gss_cred_id_t

GT2 API Command	GT4 API Command
globus_gram_client_attr_get_credential()	globus_soap_message_attr_get() with the property name GLOBUS_SOAP_MESSAGE_USER_CREDENTIAL_KEY. Migration from GRAM2 to GRAM4

GRAM2 uses a custom language for specifying a job description. GRAM4 uses an xml based language for this same purpose. In GRAM2, relations (such as count=5) can occur in any order within the RSL; in GRAM4, the relations must be in the order in the XML schema definition. The RSL attribute description below is in the order defined by the XML schema

Table 3.3. RSL Migration Table

GT2 RSL Attribute	GT4 job description element
(username = NAME)	<localUserId>NAME</localUserId>
(two_phase = TWO_PHASE_TIMEOUT) *See Note 1*	<holdState>Pending</holdState>
(executable = EXE)	<executable>EXE</executable>
(directory = DIR)	<directory>DIR</directory>
(arguments=ARG1 ... ARGN)	<argument>ARG1</argument> ... <argument>ARGN</argument>
(environment = (ENV_VAR_1 ENV_VAL_1) ... (ENV_VAR_N ENV_VAL_N))	<environment> <name>ENV_VAR_1</name> <value>ENV_VAL_1</value> ... <name>ENV_VAR_N</name> <value>ENV_VAL_N</value> </environment>
(stdin = LOCAL_FILE_PATH) *See Note 2*	<stdin>file:///LOCAL_FILE_PATH</stdin>
(stdout = LOCAL_FILE_PATH) *See Note 2*	<stdout>file:///LOCAL_FILE_PATH</stdout>
(stderr = LOCAL_FILE_PATH) *See Note 2*	<stderr>file:///LOCAL_FILE_PATH</stderr>
(count = NUMBER)	<count>NUMBER</count>
(library_path = PATH_ELEMENT_1 ... PATH_ELEMENT_N)	<libraryPath>PATH_ELEMENT_1</libraryPath> ... <libraryPath>PATH_ELEMENT_N</libraryPath>
(host_count = NUMBER)	<hostCount>NUMBER</hostCount>
(project = PROJECT)	<project>PROJECT</project>
(queue = QUEUE)	<queue>QUEUE</queue>
(max_time = MINUTES)	<maxTime>MINUTES</maxTime>
(max_wall_time = MINUTES)	<maxWallTime>MINUTES</maxWallTime>
(max_cpu_time = MINUTES)	<maxCpuTime>MINUTES</maxCpuTime>
(max_memory = MEGABYTES)	<maxMemory>MEGABYTES</maxMemory>
(min_memory = MEGABYTES)	<minMemory>MEGABYTES</minMemory>
(job_type = JOBTYP)	<jobType>JOBTYP</jobType>
(file_stage_in = (REMOTE_GRIDFTP_URL_1 LOCAL_FILE_PATH_1) ... (REMOTE_GRIDFTP_URL_N LOCAL_FILE_PATH_N)) *See Note 4*	<fileStageIn> <transfer> <sourceUrl>REMOTE_GRIDFTP_URL_1</sourceUrl> <destinationUrl>file:///LOCAL_FILE_PATH_1</destinationUrl> </transfer> <transfer> <sourceUrl>REMOTE_GRIDFTP_URL_N</sourceUrl> <destinationUrl>file:///LOCAL_FILE_PATH_N</destinationUrl> </transfer> </fileStageIn>
(file_stage_out = (LOCAL_FILE_PATH_1 REMOTE_GRIDFTP_URL_1) ... (LOCAL_FILE_PATH_N REMOTE_GRIDFTP_URL_N)) *See Note 4*	<fileStageOut> <transfer> <sourceUrl>file:///LOCAL_FILE_PATH_1</sourceUrl> <destinationUrl>REMOTE_GRIDFTP_URL_1</destinationUrl> </transfer> <transfer> <sourceUrl>file:///LOCAL_FILE_PATH_N</sourceUrl> <destinationUrl>REMOTE_GRIDFTP_URL_N</destinationUrl> </transfer> </fileStageOut>

Note 1: The globusrun-ws program will automatically release the hold after receiving the indicated hold state. To simulate the two-phase submit timeout, an application could set the initial termination time of the resource. A hold

state may be set for fileCleanUp state for two-phase commit end, but it is not possible to submit a job with both hold states.

Note 2: stdin, stdout, and stderr must only be a local file URL. Ftp and gridftp URLs can be handled by using a fileStageIn and fileStageOut elements (described below).

Note 3: Value job types for GRAM4 are multiple (the default), single, mpi, and condor.

Note 4: The GRAM4 service uses RFT to transfer files. This only supports gridftp and ftp file transfers. The local file path must be a mappable by an entry in the file system mapping file.

The following RSL attributes have no direct equivalent in GRAM4:

- `dry_run`: Similar behavior can be obtained by using a job hold state of Pending and then destroying the job resource without releasing the hold.
- `file_stage_in_shared`: No support for the GASS cache, hence this is gone. Applications may use RFT to transfer files before submitting a batch of jobs.
- `gass_cache`: GASS cache is not used by GRAM4, so there is no need for setting the cache path.
- `gram_my_job`: collective operations are enabled for every managed execution job service via rendezvous registration
- `proxy_timeout`: Delegated security proxies are handled via the DelegationFactory Service. Resource lifetime is controlled by the `wsrl:SetTerminationTime` operation
- `remote_io_url`: The GRAM4 service does not use GASS, so there is no equivalent to this.
- `restart`: There is no equivalent.
- `rsl_substitution`: The GRAM4 service does not support user-defined substitutions. Certain values may be referenced in some RSL values by a similar technique, but these are for system configuration parameters only. See the GRAM4 job description document for description of RSL variable syntax, values, and attributes where they may be used.
- `save_state`: All GRAM4 jobs are persistent, so there is no elements related to this.
- `scratch_dir`: This is now a deployment configuration option.
- `stderr_position`: Standard error streaming is now a feature of the globusrun-ws program instead of part of the GRAM4 service, so there is no equivalent element for restarting error streaming at a specific point.
- `stdout_position`: Standard output streaming is now a feature of the globusrun-ws program instead of part of the GRAM4 service, so there is no equivalent element for restarting output streaming at a specific point.

Here are some examples of converting some GRAM2 RSLs to GRAM4.

Table 3.4. RSL Migration Examples

GRAM2 RSL	GRAM4 Job Description
<pre>(* Simple Job Request With Arguments *) &(executable = /bin/echo) (arguments = Hello, Grid)</pre>	<pre><?xml version="1.0"?> <!-- Simple Job Request With Ar <job xmlns:ns1="http://www.glob <ns1:executable>/bin/echo</ns <ns1:argument>Hello,</ns1:arg <ns1:argument>Grid</ns1:argum </job></pre>

GRAM2 RSL	GRAM4 Job Description
<pre>(* Multijob Request *) +(&(executable = /bin/echo) (arguments = Hello, Grid From Subjob 1) (resource_manager_name = resource-manager-1.globus.org) (count = 1)) (&(executable = mpi-hello) (arguments = Hello, Grid From Subjob 2) (resource_manager_name = resource-manager-2.globus.org) (count = 2) (jobtype = mpi))</pre>	

GRAM2 RSL	GRAM4 Job Description
	<pre> <?xml version="1.0" <!-- Multijob Reque <multiJob <!-- namespace xmlns:gram="htt <!-- namespace xmlns:wsa="http <factoryEndpo <wsa:Address> <!-- URL fo https://res </wsa:Address <!-- Referenc <wsa:Referenc <!-- ID for <gram:Resou </wsa:Referen </factoryEndpoi <job> <factoryEndpo <wsa:Address <!-- UR https:/ </wsa:Addre <!-- Refere <wsa:Refere <!-- ID f <gram:Res </wsa:Refer </factoryEndp <executable>/ <argument>Hel <argument>Gri <argument>Fro <argument>Sub <argument>1</ <count>1</cou </job> <job> <factoryEndpo <wsa:Address <!-- UR https:/ </wsa:Addre <!-- Refere <wsa:Refere <!-- ID f <gram:Res </wsa:Refer </factoryEndp </pre>

GRAM2 RSL	GRAM4 Job Description
	<pre data-bbox="1321 243 1624 554"><executable>m <argument>Hel <argument>Gri <argument>Fro <argument>Sub <argument>2</ <count>2</cou <jobType>mpi< </job> </multiJob></pre>

Glossary

G

globusrun-ws A command line program used to submit jobs to a GRAM4 service. See the the GRAM4 Commandline page.

J

job description Term used to describe a GRAM4 job for GT4.

M

Managed Job Factory Service (MJFS) [FIXME]

R

Resource Specification Language (RSL) Term used to describe a GRAM job for GT2 and GT3. (Note: This is not the same as RLS - the Replica Location Service)

S

scheduler Term used to describe a job scheduler mechanism to which GRAM interfaces. It is a networked system for submitting, controlling, and monitoring the workload of batch jobs in one or more computers. The jobs or tasks are scheduled for execution at a time chosen by the subsystem according to an available policy and availability of resources. Popular job schedulers include Portable Batch System (PBS), Platform LSF, and IBM LoadLeveler.

Scheduler Event Generator (SEG) The Scheduler Event Generator (SEG) is a program which uses scheduler-specific monitoring modules to generate job state change events. Depending on scheduler-specific requirements, the SEG may need to run with privileges to enable it to obtain scheduler event notifications. As such, one SEG runs per scheduler resource. For example, on a host which provides access to both PBS and fork jobs, two SEGs, running at (potentially) different privilege levels will be running. One SEG instance exists for any particular scheduled resource instance (one for all homogeneous PBS queues, one for all fork jobs, etc). The SEG is implemented in an executable called the globus-scheduler-event-generator, located in the Globus Toolkit's libexec directory.

superuser do (sudo) Allows a system administrator to give certain users (or groups of users) the ability to run some (or all) commands as root or another user while logging the commands and arguments. See <http://www.courtesan.com/sudo/> for more information.

U

Universally Unique Identifier (UUID) Identifier that is immutable and unique across time and space.

GT 4.2.1 GRAM4: Quality Profile

Table of Contents

1. Test coverage reports	1
2. Code analysis reports	1
3. Outstanding bugs	1
4. Bug Fixes	4
5. Performance reports	5

<titleabbrev>Quality Profile</titleabbrev>

1. Test coverage reports

- [4.2.1 Testing Status Report](#)¹

2. Code analysis reports

- No code analysis reports have been generated at this time.

3. Outstanding bugs

- [Bug 3384](#):² Inconsistent jobType/count parameter semantics
- [Bug 3529](#):³ setup/postinstall fatal errors should be warnings
- [Bug 3575](#):⁴ SEG dependent on GLOBUS_LOCATION env var
- [Bug 3748](#):⁵ WS-GRAM Pluggable Resource Manager Backend
- [Bug 3803](#):⁶ Default scratchDirectory doesn't exist
- [Bug 3892](#):⁷ Out of date performance data?
- [Bug 3948](#):⁸ Service must release all of its resources on deactivation
- [Bug 4181](#):⁹ Allow File Staging To/From globusrun-ws application without an external server
- [Bug 4182](#):¹⁰ Improve Condor/Fork Job Monitoring for reliability and security

¹ ../reports/tests/

² http://bugzilla.globus.org/bugzilla/show_bug.cgi?id=3384

³ http://bugzilla.globus.org/bugzilla/show_bug.cgi?id=3529

⁴ http://bugzilla.globus.org/bugzilla/show_bug.cgi?id=3575

⁵ http://bugzilla.globus.org/bugzilla/show_bug.cgi?id=3748

⁶ http://bugzilla.globus.org/bugzilla/show_bug.cgi?id=3803

⁷ http://bugzilla.globus.org/bugzilla/show_bug.cgi?id=3892

⁸ http://bugzilla.globus.org/bugzilla/show_bug.cgi?id=3948

⁹ http://bugzilla.globus.org/bugzilla/show_bug.cgi?id=4181

¹⁰ http://bugzilla.globus.org/bugzilla/show_bug.cgi?id=4182

- [Bug 4513](#):¹¹ LD_LIBRARY_PATH should not be set if no library_path is specified
- [Bug 4550](#):¹² Multijob code not checking for existence of job credential
- [Bug 4684](#):¹³ Loading persisted jobs with expired delegation resources causes stacktraces
- [Bug 4719](#):¹⁴ globus runs /usr/bin/env without checking for \u
- [Bug 4734](#):¹⁵ Missing wsa:Action for GRAM4 rendezvous register operations
- [Bug 4761](#):¹⁶ Scheduler Tutorial is missing WS-GRAM setup package
- [Bug 4778](#):¹⁷ WS-Fork job manager doesn't set environment up for mpi jobs
- [Bug 4787](#):¹⁸ no lifetime management for WS Rendezvous
- [Bug 4817](#):¹⁹ Condor OS and ARCH do not have dynamic defaults
- [Bug 4864](#):²⁰ environment variables containing '=' get escaped
- [Bug 4918](#):²¹ user account details are cached even for unknown users
- [Bug 4944](#):²² Multijob resources never yield to memory pressure and can't be destroyed
- [Bug 5012](#):²³ Container in livelock state for an incorrectly mapped DN
- [Bug 5017](#):²⁴ gram[24] tests that need to be updated
- [Bug 5397](#):²⁵ GRAM4 recovery of persisted job resources needs to be reviewed
- [Bug 5402](#):²⁶ "invalid password" error messages
- [Bug 5433](#):²⁷ public interface doc lists non-public/internal APIs
- [Bug 5471](#):²⁸ GRAM Jobs Hang in Unsubmitted State
- [Bug 5484](#):²⁹ Review and Update 4.2 GRAM doc
- [Bug 5515](#):³⁰ Destruction of subscription resources in a container shutdown/restart

¹¹ http://bugzilla.globus.org/bugzilla/show_bug.cgi?id=4513

¹² http://bugzilla.globus.org/bugzilla/show_bug.cgi?id=4550

¹³ http://bugzilla.globus.org/bugzilla/show_bug.cgi?id=4684

¹⁴ http://bugzilla.globus.org/bugzilla/show_bug.cgi?id=4719

¹⁵ http://bugzilla.globus.org/bugzilla/show_bug.cgi?id=4734

¹⁶ http://bugzilla.globus.org/bugzilla/show_bug.cgi?id=4761

¹⁷ http://bugzilla.globus.org/bugzilla/show_bug.cgi?id=4778

¹⁸ http://bugzilla.globus.org/bugzilla/show_bug.cgi?id=4787

¹⁹ http://bugzilla.globus.org/bugzilla/show_bug.cgi?id=4817

²⁰ http://bugzilla.globus.org/bugzilla/show_bug.cgi?id=4864

²¹ http://bugzilla.globus.org/bugzilla/show_bug.cgi?id=4918

²² http://bugzilla.globus.org/bugzilla/show_bug.cgi?id=4944

²³ http://bugzilla.globus.org/bugzilla/show_bug.cgi?id=5012

²⁴ http://bugzilla.globus.org/bugzilla/show_bug.cgi?id=5017

²⁵ http://bugzilla.globus.org/bugzilla/show_bug.cgi?id=5397

²⁶ http://bugzilla.globus.org/bugzilla/show_bug.cgi?id=5402

²⁷ http://bugzilla.globus.org/bugzilla/show_bug.cgi?id=5433

²⁸ http://bugzilla.globus.org/bugzilla/show_bug.cgi?id=5471

²⁹ http://bugzilla.globus.org/bugzilla/show_bug.cgi?id=5484

³⁰ http://bugzilla.globus.org/bugzilla/show_bug.cgi?id=5515

- [Bug 5516](#).³¹ Destruction of subscription resources in a container shutdown/restart
- [Bug 5611](#).³² GramJob API changes to improve performace and efficiency
- [Bug 5698](#).³³ Allow a prologue/epilogue script for 'mpi' and 'multiple' jobs
- [Bug 5712](#).³⁴ Gram auditing: local_job_id format variations
- [Bug 5713](#).³⁵ GRAM auditing: Failed database connection loses audit records
- [Bug 5714](#).³⁶ GRAM Auditing: additional data in audit records
- [Bug 5725](#).³⁷ Gram auditing: housekeeping for the auditRecords database
- [Bug 5770](#).³⁸ GRAM4 auditing: inconsistent data in job_description column of DB
- [Bug 5776](#).³⁹ GRAM4 auditing: Need for an INFO log message
- [Bug 5777](#).⁴⁰ GRAM2 auditing: database connection times out
- [Bug 5778](#).⁴¹ GRAM2 auditing: no error message on db update failure
- [Bug 5805](#).⁴² Change threadpools from Gram custom implementation to java.util.concurrent
- [Bug 5820](#).⁴³ Improve Condor Logfile Processing in GRAM
- [Bug 5843](#).⁴⁴ Swap custom threadpool with ExecutorServices from java.util.concurrent
- [Bug 5853](#).⁴⁵ create automated tests for globus-job-*-ws programs
- [Bug 5859](#).⁴⁶ Java GramJob getExitCode() always returns 0
- [Bug 5969](#).⁴⁷ GRAM Job submission failed!!!
- [Bug 6002](#).⁴⁸ globusrun-ws hangs indefinitely
- [Bug 6019](#).⁴⁹ CEDPS: Add executable path to log statement
- [Bug 6043](#).⁵⁰ Confusing JobID in timeout message

³¹ http://bugzilla.globus.org/bugzilla/show_bug.cgi?id=5516

³² http://bugzilla.globus.org/bugzilla/show_bug.cgi?id=5611

³³ http://bugzilla.globus.org/bugzilla/show_bug.cgi?id=5698

³⁴ http://bugzilla.globus.org/bugzilla/show_bug.cgi?id=5712

³⁵ http://bugzilla.globus.org/bugzilla/show_bug.cgi?id=5713

³⁶ http://bugzilla.globus.org/bugzilla/show_bug.cgi?id=5714

³⁷ http://bugzilla.globus.org/bugzilla/show_bug.cgi?id=5725

³⁸ http://bugzilla.globus.org/bugzilla/show_bug.cgi?id=5770

³⁹ http://bugzilla.globus.org/bugzilla/show_bug.cgi?id=5776

⁴⁰ http://bugzilla.globus.org/bugzilla/show_bug.cgi?id=5777

⁴¹ http://bugzilla.globus.org/bugzilla/show_bug.cgi?id=5778

⁴² http://bugzilla.globus.org/bugzilla/show_bug.cgi?id=5805

⁴³ http://bugzilla.globus.org/bugzilla/show_bug.cgi?id=5820

⁴⁴ http://bugzilla.globus.org/bugzilla/show_bug.cgi?id=5843

⁴⁵ http://bugzilla.globus.org/bugzilla/show_bug.cgi?id=5853

⁴⁶ http://bugzilla.globus.org/bugzilla/show_bug.cgi?id=5859

⁴⁷ http://bugzilla.globus.org/bugzilla/show_bug.cgi?id=5969

⁴⁸ http://bugzilla.globus.org/bugzilla/show_bug.cgi?id=6002

⁴⁹ http://bugzilla.globus.org/bugzilla/show_bug.cgi?id=6019

⁵⁰ http://bugzilla.globus.org/bugzilla/show_bug.cgi?id=6043

- [Bug 6065](#).⁵¹ Deleting delegated credential does not work
- [Bug 6069](#).⁵² JDD documentation issues
- [Bug 6072](#).⁵³ specification of RAM per process in parallel jobs
- [Bug 6091](#).⁵⁴ GRAM4 JDD substitution detection is too broad
- [Bug 6192](#).⁵⁵ Apply relevant VDT patches
- [Bug 6203](#).⁵⁶ audit record not inserted for RSL argument containing single quotes
- [Bug 6204](#).⁵⁷ Drain Mode for the GRAM service
- [Bug 6279](#).⁵⁸ document streaming overhead in globusrun-ws
- [Bug 6289](#).⁵⁹ ws-gram multiJob submissions fail with extensions element
- [Bug 6350](#).⁶⁰ Infinite loop in org.globus.exec.service.utils.UserProxyCreator
- [Bug 6351](#).⁶¹ Infinite loop in org.globus.exec.service.utils.UserProxyCreator
- [Bug 6376](#).⁶² WS GRAM container becomes non responsive
- [Bug 6387](#).⁶³ add service security descriptor checks in job submission
- [Bug 6400](#).⁶⁴ Fix audit logging problems in Gram2 and Gram4 in globus_4_0_branch
- [Bug 6402](#).⁶⁵ Throughput tester destroying shared delegation credentials

4. Bug Fixes

- [Bug 5617](#).⁶⁶ GRAM4 seg hangs with fork jobs
- [Bug 5831](#).⁶⁷ Changes in GramJob in Gram 4.2
- [Bug 5977](#).⁶⁸ Add user's DN to INFO logging statement for each job submission
- [Bug 5982](#).⁶⁹ GRAM handles lack of local username badly(java.lang.NullPointerException)

⁵¹ http://bugzilla.globus.org/bugzilla/show_bug.cgi?id=6065

⁵² http://bugzilla.globus.org/bugzilla/show_bug.cgi?id=6069

⁵³ http://bugzilla.globus.org/bugzilla/show_bug.cgi?id=6072

⁵⁴ http://bugzilla.globus.org/bugzilla/show_bug.cgi?id=6091

⁵⁵ http://bugzilla.globus.org/bugzilla/show_bug.cgi?id=6192

⁵⁶ http://bugzilla.globus.org/bugzilla/show_bug.cgi?id=6203

⁵⁷ http://bugzilla.globus.org/bugzilla/show_bug.cgi?id=6204

⁵⁸ http://bugzilla.globus.org/bugzilla/show_bug.cgi?id=6279

⁵⁹ http://bugzilla.globus.org/bugzilla/show_bug.cgi?id=6289

⁶⁰ http://bugzilla.globus.org/bugzilla/show_bug.cgi?id=6350

⁶¹ http://bugzilla.globus.org/bugzilla/show_bug.cgi?id=6351

⁶² http://bugzilla.globus.org/bugzilla/show_bug.cgi?id=6376

⁶³ http://bugzilla.globus.org/bugzilla/show_bug.cgi?id=6387

⁶⁴ http://bugzilla.globus.org/bugzilla/show_bug.cgi?id=6400

⁶⁵ http://bugzilla.globus.org/bugzilla/show_bug.cgi?id=6402

⁶⁶ http://bugzilla.globus.org/bugzilla/show_bug.cgi?id=5617

⁶⁷ http://bugzilla.globus.org/bugzilla/show_bug.cgi?id=5831

⁶⁸ http://bugzilla.globus.org/bugzilla/show_bug.cgi?id=5977

⁶⁹ http://bugzilla.globus.org/bugzilla/show_bug.cgi?id=5982

- [Bug 6102](#).⁷⁰ GRAM4 throughput tester for 4.2
- [Bug 6172](#).⁷¹ Bad error message for "file not found"
- [Bug 6271](#).⁷² Proxy cleanup doesn't check for authz callouts, uses grid-mapfile check
- [Bug 6272](#).⁷³ Proxy cleanup doesn't check for authz callouts, uses grid-mapfile check
- [Bug 6283](#).⁷⁴ globusrun-ws doesn't resolve IP addresses to host names
- [Bug 6320](#).⁷⁵ Audit logging with PostgreSQL fails in 4.2
- [Bug 6327](#).⁷⁶ Add support for Derby
- [Bug 6333](#).⁷⁷ Provide support for a common user home for testing purposes
- [Bug 6341](#).⁷⁸ Infinite loop and blocking in org.globus.exec.service.utils.DelegatedCredential
- [Bug 6349](#).⁷⁹ Infinite loop and blocking in org.globus.exec.service.utils.DelegatedCredential
- [Bug 6357](#).⁸⁰ Changes in current Gram4 audit logging

5. Performance reports

- No code performance reports have been generated at this time.

⁷⁰ http://bugzilla.globus.org/bugzilla/show_bug.cgi?id=6102

⁷¹ http://bugzilla.globus.org/bugzilla/show_bug.cgi?id=6172

⁷² http://bugzilla.globus.org/bugzilla/show_bug.cgi?id=6271

⁷³ http://bugzilla.globus.org/bugzilla/show_bug.cgi?id=6272

⁷⁴ http://bugzilla.globus.org/bugzilla/show_bug.cgi?id=6283

⁷⁵ http://bugzilla.globus.org/bugzilla/show_bug.cgi?id=6320

⁷⁶ http://bugzilla.globus.org/bugzilla/show_bug.cgi?id=6327

⁷⁷ http://bugzilla.globus.org/bugzilla/show_bug.cgi?id=6333

⁷⁸ http://bugzilla.globus.org/bugzilla/show_bug.cgi?id=6341

⁷⁹ http://bugzilla.globus.org/bugzilla/show_bug.cgi?id=6349

⁸⁰ http://bugzilla.globus.org/bugzilla/show_bug.cgi?id=6357

GT 4.2.1 Release Notes: GRAM4

Table of Contents

1. Component Overview	1
2. Feature summary	1
3. Summary of Changes in GRAM4	2
4. Bug Fixes	2
5. Known Problems	3
6. Technology dependencies	6
7. Tested platforms	7
8. Backward compatibility summary	7
9. Associated Standards	7
10. For More Information	7
Glossary	7

<titleabbrev>Release Notes</titleabbrev>

1. Component Overview

The Web Services Grid Resource Allocation and Management (GRAM4) component comprises a set of WSRF-compliant Web services to locate, submit, monitor, and cancel jobs on Grid computing resources. GRAM4 is not a *job scheduler*, but rather a set of services and clients for communicating with a range of different batch/cluster job schedulers using a common protocol. GRAM4 is meant to address a range of jobs where reliable operation, stateful monitoring, credential management, and file staging are important.

Note

The GRAM server is typically deployed in conjunction with the [Delegation](#) and [RFT](#) services to address data staging, delegation of proxy credentials, and computation monitoring and management in an integrated manner.

2. Feature summary

New Features new since 4.0.x

- New terminate method in the client-side GramJob API
- Improved job lifetime management for users and admins
- Added configuration for "default" Local Resource Managers

Other Standard Supported Features

- Remote job execution and management
- Uniform and flexible interface to batch scheduling systems
- File staging before and after job execution
- File / directory clean up after job execution (after file stage out)

- Service auditing for each submitted

Deprecated Features

- With the addition of the new terminate method in the GramJob API, the destroy method is no longer necessary. For backward compatibility, the destroy method was left in the GramJob API, but it simply calls the terminate method. During the 4.2.x series, clients using the destroy method should change to instead use terminate. In GT 4.4, the plan is to remove the destroy method.

3. Summary of Changes in GRAM4

There are significant enhancements in this release to improve reliability of GRAM4 audit logging. Database errors are handled better. If the audit database is unavailable, job records are written to local disk and later inserted. In addition, there are a number of other bug fixes.

4. Bug Fixes

- [Bug 5617](#):¹ GRAM4 seg hangs with fork jobs
- [Bug 5831](#):² Changes in GramJob in Gram 4.2
- [Bug 5977](#):³ Add user's DN to INFO logging statement for each job submission
- [Bug 5982](#):⁴ GRAM handles lack of local username badly(java.lang.NullPointerException)
- [Bug 6102](#):⁵ GRAM4 throughput tester for 4.2
- [Bug 6172](#):⁶ Bad error message for "file not found"
- [Bug 6271](#):⁷ Proxy cleanup doesn't check for authz callouts, uses grid-mapfile check
- [Bug 6272](#):⁸ Proxy cleanup doesn't check for authz callouts, uses grid-mapfile check
- [Bug 6283](#):⁹ globusrun-ws doesn't resolve IP addresses to host names
- [Bug 6320](#):¹⁰ Audit logging with PostgreSQL fails in 4.2
- [Bug 6327](#):¹¹ Add support for Derby
- [Bug 6333](#):¹² Provide support for a common user home for testing purposes
- [Bug 6341](#):¹³ Infinite loop and blocking in org.globus.exec.service.utils.DelegatedCredential

¹ http://bugzilla.globus.org/bugzilla/show_bug.cgi?id=5617
² http://bugzilla.globus.org/bugzilla/show_bug.cgi?id=5831
³ http://bugzilla.globus.org/bugzilla/show_bug.cgi?id=5977
⁴ http://bugzilla.globus.org/bugzilla/show_bug.cgi?id=5982
⁵ http://bugzilla.globus.org/bugzilla/show_bug.cgi?id=6102
⁶ http://bugzilla.globus.org/bugzilla/show_bug.cgi?id=6172
⁷ http://bugzilla.globus.org/bugzilla/show_bug.cgi?id=6271
⁸ http://bugzilla.globus.org/bugzilla/show_bug.cgi?id=6272
⁹ http://bugzilla.globus.org/bugzilla/show_bug.cgi?id=6283
¹⁰ http://bugzilla.globus.org/bugzilla/show_bug.cgi?id=6320
¹¹ http://bugzilla.globus.org/bugzilla/show_bug.cgi?id=6327
¹² http://bugzilla.globus.org/bugzilla/show_bug.cgi?id=6333
¹³ http://bugzilla.globus.org/bugzilla/show_bug.cgi?id=6341

- [Bug 6349](#):¹⁴ Infinite loop and blocking in org.globus.exec.service.utils.DelegatedCredential
- [Bug 6357](#):¹⁵ Changes in current Gram4 audit logging

5. Known Problems

The following problems and limitations are known to exist for GRAM4 at the time of the 4.2.1 release:

5.1. Limitations

- [list limitations]

5.2. Outstanding bugs

- [Bug 3384](#):¹⁶ Inconsistent jobType/count parameter semantics
- [Bug 3529](#):¹⁷ setup/postinstall fatal errors should be warnings
- [Bug 3575](#):¹⁸ SEG dependent on GLOBUS_LOCATION env var
- [Bug 3748](#):¹⁹ WS-GRAM Plugable Resource Manager Backend
- [Bug 3803](#):²⁰ Default scratchDirectory doesn't exist
- [Bug 3892](#):²¹ Out of date performance data?
- [Bug 3948](#):²² Service must release all of its resources on deactivation
- [Bug 4181](#):²³ Allow File Staging To/From globusrun-ws application without an external server
- [Bug 4182](#):²⁴ Improve Condor/Fork Job Monitoring for reliability and security
- [Bug 4513](#):²⁵ LD_LIBRARY_PATH should not be set if no library_path is specified
- [Bug 4550](#):²⁶ Multijob code not checking for existence of job credential
- [Bug 4684](#):²⁷ Loading persisted jobs with expired delegation resources causes stacktraces
- [Bug 4719](#):²⁸ globus runs /usr/bin/env without checking for \u
- [Bug 4734](#):²⁹ Missing wsa:Action for GRAM4 rendezvous register operations

¹⁴ http://bugzilla.globus.org/bugzilla/show_bug.cgi?id=6349

¹⁵ http://bugzilla.globus.org/bugzilla/show_bug.cgi?id=6357

¹⁶ http://bugzilla.globus.org/bugzilla/show_bug.cgi?id=3384

¹⁷ http://bugzilla.globus.org/bugzilla/show_bug.cgi?id=3529

¹⁸ http://bugzilla.globus.org/bugzilla/show_bug.cgi?id=3575

¹⁹ http://bugzilla.globus.org/bugzilla/show_bug.cgi?id=3748

²⁰ http://bugzilla.globus.org/bugzilla/show_bug.cgi?id=3803

²¹ http://bugzilla.globus.org/bugzilla/show_bug.cgi?id=3892

²² http://bugzilla.globus.org/bugzilla/show_bug.cgi?id=3948

²³ http://bugzilla.globus.org/bugzilla/show_bug.cgi?id=4181

²⁴ http://bugzilla.globus.org/bugzilla/show_bug.cgi?id=4182

²⁵ http://bugzilla.globus.org/bugzilla/show_bug.cgi?id=4513

²⁶ http://bugzilla.globus.org/bugzilla/show_bug.cgi?id=4550

²⁷ http://bugzilla.globus.org/bugzilla/show_bug.cgi?id=4684

²⁸ http://bugzilla.globus.org/bugzilla/show_bug.cgi?id=4719

²⁹ http://bugzilla.globus.org/bugzilla/show_bug.cgi?id=4734

- [Bug 4761](#).³⁰ Scheduler Tutorial is missing WS-GRAM setup package
- [Bug 4778](#).³¹ WS-Fork job manager doesn't set environment up for mpi jobs
- [Bug 4787](#).³² no lifetime management for WS Rendezvous
- [Bug 4817](#).³³ Condor OS and ARCH do not have dynamic defaults
- [Bug 4864](#).³⁴ environment variables containing '=' get escaped
- [Bug 4918](#).³⁵ user account details are cached even for unknown users
- [Bug 4944](#).³⁶ Multijob resources never yield to memory pressure and can't be destroyed
- [Bug 5012](#).³⁷ Container in livelock state for an incorrectly mapped DN
- [Bug 5017](#).³⁸ gram[24] tests that need to be updated
- [Bug 5397](#).³⁹ GRAM4 recovery of persisted job resources needs to be reviewed
- [Bug 5402](#).⁴⁰ "invalid password" error messages
- [Bug 5433](#).⁴¹ public interface doc lists non-public/internal APIs
- [Bug 5471](#).⁴² GRAM Jobs Hang in Unsubmitted State
- [Bug 5484](#).⁴³ Review and Update 4.2 GRAM doc
- [Bug 5515](#).⁴⁴ Destruction of subscription resources in a container shutdown/restart
- [Bug 5516](#).⁴⁵ Destruction of subscription resources in a container shutdown/restart
- [Bug 5611](#).⁴⁶ GramJob API changes to improve performace and efficiency
- [Bug 5698](#).⁴⁷ Allow a prologue/epilogue script for 'mpi' and 'multiple' jobs
- [Bug 5712](#).⁴⁸ Gram auditing: local_job_id format variations
- [Bug 5713](#).⁴⁹ GRAM auditing: Failed database connection loses audit records

³⁰ http://bugzilla.globus.org/bugzilla/show_bug.cgi?id=4761

³¹ http://bugzilla.globus.org/bugzilla/show_bug.cgi?id=4778

³² http://bugzilla.globus.org/bugzilla/show_bug.cgi?id=4787

³³ http://bugzilla.globus.org/bugzilla/show_bug.cgi?id=4817

³⁴ http://bugzilla.globus.org/bugzilla/show_bug.cgi?id=4864

³⁵ http://bugzilla.globus.org/bugzilla/show_bug.cgi?id=4918

³⁶ http://bugzilla.globus.org/bugzilla/show_bug.cgi?id=4944

³⁷ http://bugzilla.globus.org/bugzilla/show_bug.cgi?id=5012

³⁸ http://bugzilla.globus.org/bugzilla/show_bug.cgi?id=5017

³⁹ http://bugzilla.globus.org/bugzilla/show_bug.cgi?id=5397

⁴⁰ http://bugzilla.globus.org/bugzilla/show_bug.cgi?id=5402

⁴¹ http://bugzilla.globus.org/bugzilla/show_bug.cgi?id=5433

⁴² http://bugzilla.globus.org/bugzilla/show_bug.cgi?id=5471

⁴³ http://bugzilla.globus.org/bugzilla/show_bug.cgi?id=5484

⁴⁴ http://bugzilla.globus.org/bugzilla/show_bug.cgi?id=5515

⁴⁵ http://bugzilla.globus.org/bugzilla/show_bug.cgi?id=5516

⁴⁶ http://bugzilla.globus.org/bugzilla/show_bug.cgi?id=5611

⁴⁷ http://bugzilla.globus.org/bugzilla/show_bug.cgi?id=5698

⁴⁸ http://bugzilla.globus.org/bugzilla/show_bug.cgi?id=5712

⁴⁹ http://bugzilla.globus.org/bugzilla/show_bug.cgi?id=5713

- [Bug 5714](#).⁵⁰ GRAM Auditing: additional data in audit records
- [Bug 5725](#).⁵¹ Gram auditing: housekeeping for the auditRecords database
- [Bug 5770](#).⁵² GRAM4 auditing: inconsistent data in job_description column of DB
- [Bug 5776](#).⁵³ GRAM4 auditing: Need for an INFO log message
- [Bug 5777](#).⁵⁴ GRAM2 auditing: database connection times out
- [Bug 5778](#).⁵⁵ GRAM2 auditing: no error message on db update failure
- [Bug 5805](#).⁵⁶ Change threadpools from Gram custom implementation to java.util.concurrent
- [Bug 5820](#).⁵⁷ Improve Condor Logfile Processing in GRAM
- [Bug 5843](#).⁵⁸ Swap custom threadpool with ExecutorServices from java.util.concurrent
- [Bug 5853](#).⁵⁹ create automated tests for globus-job-*-ws programs
- [Bug 5859](#).⁶⁰ Java GramJob getExitCode() always returns 0
- [Bug 5969](#).⁶¹ GRAM Job submission failed!!!
- [Bug 6002](#).⁶² globusrun-ws hangs indefinitely
- [Bug 6019](#).⁶³ CEDPS: Add executable path to log statement
- [Bug 6043](#).⁶⁴ Confusing JobID in timeout message
- [Bug 6065](#).⁶⁵ Deleting delegated credential does not work
- [Bug 6069](#).⁶⁶ JDD documentation issues
- [Bug 6072](#).⁶⁷ specification of RAM per process in parallel jobs
- [Bug 6091](#).⁶⁸ GRAM4 JDD substitution detection is too broad
- [Bug 6192](#).⁶⁹ Apply relevant VDT patches

⁵⁰ http://bugzilla.globus.org/bugzilla/show_bug.cgi?id=5714

⁵¹ http://bugzilla.globus.org/bugzilla/show_bug.cgi?id=5725

⁵² http://bugzilla.globus.org/bugzilla/show_bug.cgi?id=5770

⁵³ http://bugzilla.globus.org/bugzilla/show_bug.cgi?id=5776

⁵⁴ http://bugzilla.globus.org/bugzilla/show_bug.cgi?id=5777

⁵⁵ http://bugzilla.globus.org/bugzilla/show_bug.cgi?id=5778

⁵⁶ http://bugzilla.globus.org/bugzilla/show_bug.cgi?id=5805

⁵⁷ http://bugzilla.globus.org/bugzilla/show_bug.cgi?id=5820

⁵⁸ http://bugzilla.globus.org/bugzilla/show_bug.cgi?id=5843

⁵⁹ http://bugzilla.globus.org/bugzilla/show_bug.cgi?id=5853

⁶⁰ http://bugzilla.globus.org/bugzilla/show_bug.cgi?id=5859

⁶¹ http://bugzilla.globus.org/bugzilla/show_bug.cgi?id=5969

⁶² http://bugzilla.globus.org/bugzilla/show_bug.cgi?id=6002

⁶³ http://bugzilla.globus.org/bugzilla/show_bug.cgi?id=6019

⁶⁴ http://bugzilla.globus.org/bugzilla/show_bug.cgi?id=6043

⁶⁵ http://bugzilla.globus.org/bugzilla/show_bug.cgi?id=6065

⁶⁶ http://bugzilla.globus.org/bugzilla/show_bug.cgi?id=6069

⁶⁷ http://bugzilla.globus.org/bugzilla/show_bug.cgi?id=6072

⁶⁸ http://bugzilla.globus.org/bugzilla/show_bug.cgi?id=6091

⁶⁹ http://bugzilla.globus.org/bugzilla/show_bug.cgi?id=6192

- [Bug 6203](#):⁷⁰ audit record not inserted for RSL argument containing single quotes
- [Bug 6204](#):⁷¹ Drain Mode for the GRAM service
- [Bug 6279](#):⁷² document streaming overhead in globusrun-ws
- [Bug 6289](#):⁷³ ws-gram multiJob submissions fail with extensions element
- [Bug 6350](#):⁷⁴ Infinite loop in org.globus.exec.service.utils.UserProxyCreator
- [Bug 6351](#):⁷⁵ Infinite loop in org.globus.exec.service.utils.UserProxyCreator
- [Bug 6376](#):⁷⁶ WS GRAM container becomes non responsive
- [Bug 6387](#):⁷⁷ add service security descriptor checks in job submission
- [Bug 6400](#):⁷⁸ Fix audit logging problems in Gram2 and Gram4 in globus_4_0_branch
- [Bug 6402](#):⁷⁹ Throughput tester destroying shared delegation credentials

6. Technology dependencies

GRAM depends on the following GT components:

- Java WS Core
- Transport-Level Security
- Delegation Service
- RFT
- GridFTP
- MDS - internal libraries
- The XML::Parser Perl module is required <http://search.cpan.org/~msergeant/XML-Parser/Parser.pm>

Other scheduler adapters available for GT 4.2.1 release:

- [SGE scheduler adapter interface](#)⁸⁰
- IBM LoadLeveler (As of release 3.3.1). For more information see "What's new" in the [LoadLeveler product documentation](#)⁸¹

⁷⁰ http://bugzilla.globus.org/bugzilla/show_bug.cgi?id=6203

⁷¹ http://bugzilla.globus.org/bugzilla/show_bug.cgi?id=6204

⁷² http://bugzilla.globus.org/bugzilla/show_bug.cgi?id=6279

⁷³ http://bugzilla.globus.org/bugzilla/show_bug.cgi?id=6289

⁷⁴ http://bugzilla.globus.org/bugzilla/show_bug.cgi?id=6350

⁷⁵ http://bugzilla.globus.org/bugzilla/show_bug.cgi?id=6351

⁷⁶ http://bugzilla.globus.org/bugzilla/show_bug.cgi?id=6376

⁷⁷ http://bugzilla.globus.org/bugzilla/show_bug.cgi?id=6387

⁷⁸ http://bugzilla.globus.org/bugzilla/show_bug.cgi?id=6400

⁷⁹ http://bugzilla.globus.org/bugzilla/show_bug.cgi?id=6402

⁸⁰ <http://www.lesc.ic.ac.uk/projects/SGE-GT4.html>

⁸¹ <http://publib.boulder.ibm.com/infocenter/clresctr/index.jsp?topic=/com.ibm.cluster.infocenter.doc/library.html>

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

7. Tested platforms

Tested platforms for GRAM4:

- Linux
 - Fedora Core 1 i686
 - Fedora Core 3 i686
 - Fedora Core 3 yup xeon
 - RedHat 7.3 i686
 - RedHat 9 x86
 - Debian Sarge x86
 - Debian 3.1 i686

Tested containers for GRAM4:

- Java WS Core container

8. Backward compatibility summary

Protocol changes since GRAM4 in the GT4.0 series:

- The Java WS Core Framework has been updated from the draft versions of the WSRF/WSN and WS Addressing specifications to the final versions WSRF 1.2, WSN 1.3 and WS Addressing 1.0. There is no backward compatibility between this version and any previous versions.

9. Associated Standards

[See the Java WS Core related standards](#)

10. For More Information

See [GRAM4](#) for more information about this component.

Glossary

J

job scheduler See the term [scheduler](#)⁴.

⁴ #scheduler