

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.
-----------------------	--

jobTTLOnlyAfterProcessing	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 + jobTTLOnlyAfterProcessing)

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>

- `Char()` 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 `$self->{CDATA}`.
- `StartTag()` Responsible for collecting the attributes associated with the element. It also increments the counter, which keeps track of the number of child elements to the current extension element, and pushes the current element name onto the `@scope` queue for later use.
- `EndTag()` Takes the CDATA collected by `Char()` and creates new Perl job description hash entries. This is most likely where you will need to do most of your work when adding support for new extension elements. Two useful variables are `$currentScope` and `$parentScope`. 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 `@scope` 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.
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 passed (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 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 fail 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. 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