GridGateWay 1.0 Documentation: Installation and Configuration Guide

GridGateWay 1.0 Documentation: Installation and Configuration Guide

Published March, 2007

Table of Contents

List of Tables

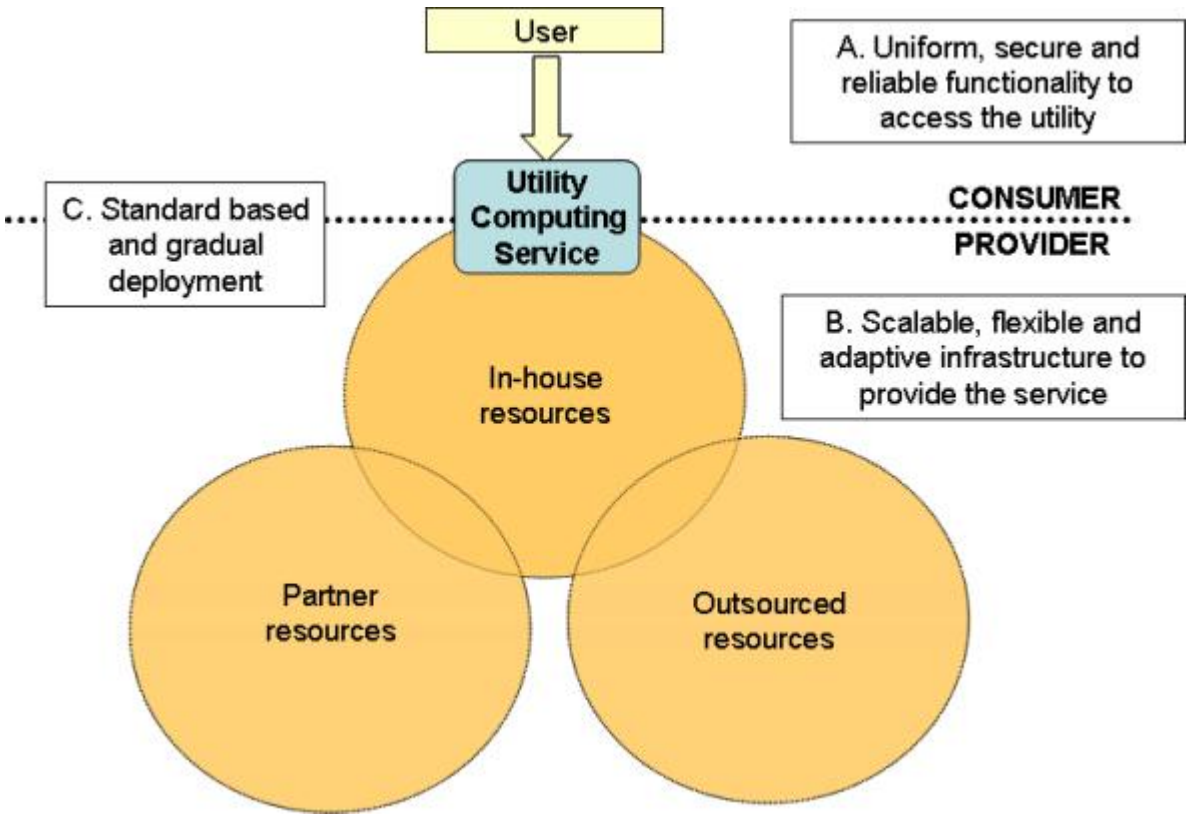
Chapter 1. Utility Computing key concepts

What is Utility Computing?

Utility is a computing term related to a new paradigm for information technology (IT) provision. A utility computing solution should provide access to the latest computing platform and technology and still be flexible enough to adjust capacity as required without needing to purchase costly hardware. In doing so information technology costs are transferred from fixed to variable as the organization gets charged for how much they use. Such pay-as-you-use paradigm exhibits several potential benefits for an organization: reducing fixed costs, treating information technology as a variable cost, providing access to unlimited computational capacity and improving flexibility, thereby making resource provision more agile and adaptive.

The deployment of a utility computing solution involves a full separation between the provider and the consumer. The consumer requires a uniform, secure and reliable functionality to access the utility computing service and the provider requires a scalable, flexible and adaptive infrastructure to provide the service. The solution should be based on standards and allow a gradual deployment in order to obtain a favourable response from the application developers and the IT staff.

Figure 1-1. Utility Computing solution



What is a Grid infrastructure?

Several Distributed Resource Management (DRM) systems have been developed to provide workload management of applications. These systems first appeared in the late eighties, coinciding with the advent of parallel and distributed platforms, such as high performance computing servers and clusters. There are a number of commercial and open source local workload management systems available today, such as PBS, SGE, LSF or Condor, and each one is used for different underlying computer architectures and execution profiles. In any case their benefits in cost minimization and performance maximization are mainly due to greater utilization of underlying resources.

Even though DRM systems share many capabilities, mainly batch queueing, job scheduling and resource management; they do not provide a common interface and security framework, and so their integration is not possible. Such lack of interoperability involves the existence within an organization of independent computational platforms (vertical silos) responsible for distinct functions that require specialized administration skills and generates over-provisioning and global load unbalance. Moreover, such technologies are also unsuitable to build computational infrastructures where resources are scattered across several administrative domains, each with its own security policy and DRM system. A grid infrastructure offers a common layer to integrate these non-interoperable computational platforms by defining a consistent set of abstraction and interfaces for access to, and management of, shared resources.

What is Globus?

The Globus Toolkit (http://www.globus.org/), a de facto standard in grid computing, is open source software that implements a collection of high level services at the grid infrastructure layer. These services include, among others, monitoring and discovery services (MDS), resource allocation & management (GRAM), a security infrastructure (GSI), and file transfer services (RFT). The Globus Toolkit meets most of the abstract requirements set forth in Open Grid Services Architecture (OGSA (http://forge.gridforum.org/projects/ogsa-wg/)). OGSA, under development by the Global Grid Forum (GGF), aims to define a common, standard, and open architecture for the grid. The Web Services Resource Framework (WSRF (http://www.globus.org/wsrf/)), developed by OASIS (http://www.oasis-open.org/), defines a family of specifications based on standard Web Services to access the stateful resources that OGSA needs. In fact, the Globus Toolkit is implemented on top of the WSRF standard. Several open-source software components (http://www.globus.org/grid_software/computation/) are available to deploy Globus-based grid solutions that address the requirements of the new grid projects.

Figure 1-2. Globus Grid infrastructure

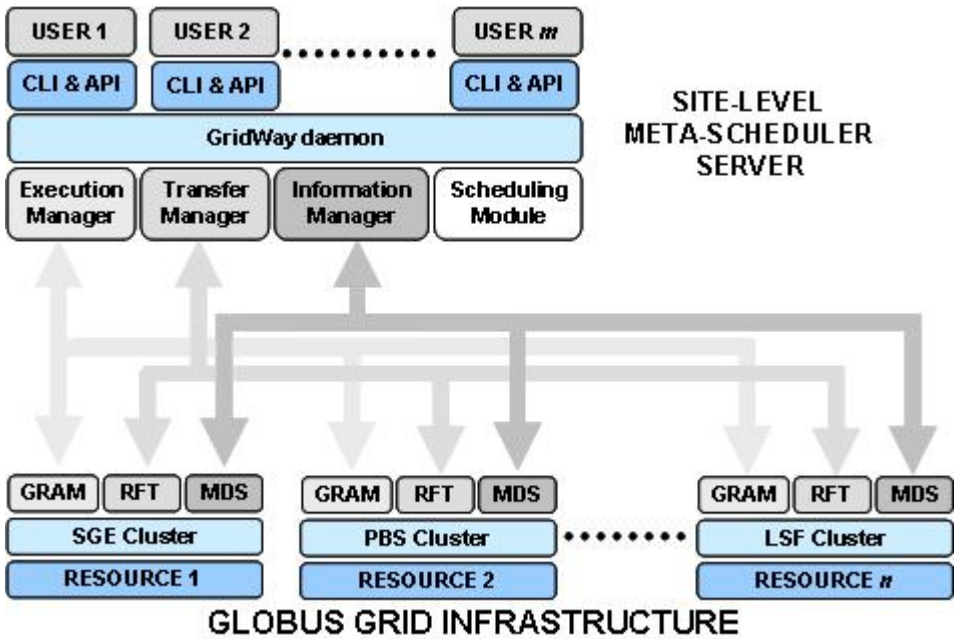
What is the GridWay Metascheduler?

GridWay is a metascheduling technology that gives end users, application developers and managers of

Globus infrastructures a scheduling functionality similar to that found on local distributed resource management systems:

- Advanced scheduling capabilities on a grid consisting of Globus services
- Detection and recovery from remote and local job execution failure situations
- DRM-like commands to submit, monitor, synchronize and control single, array and interdependent jobs; and monitor Globus resources and users
- Full support for C and JAVA DRMAA GGF standard for the development of distributed applications on Globus services
- Straightforward deployment that does not require new services apart from those provided by the Globus Toolkit
- Modular architecture that allows an easy incorporation of new grid services and interoperability between different grid infrastructures (Globus Toolkit WS, Globus Toolkit pre-WS and EGEE)

Figure 1-3. Globus Grid infrastructure with GridWay as site-level metascheduler

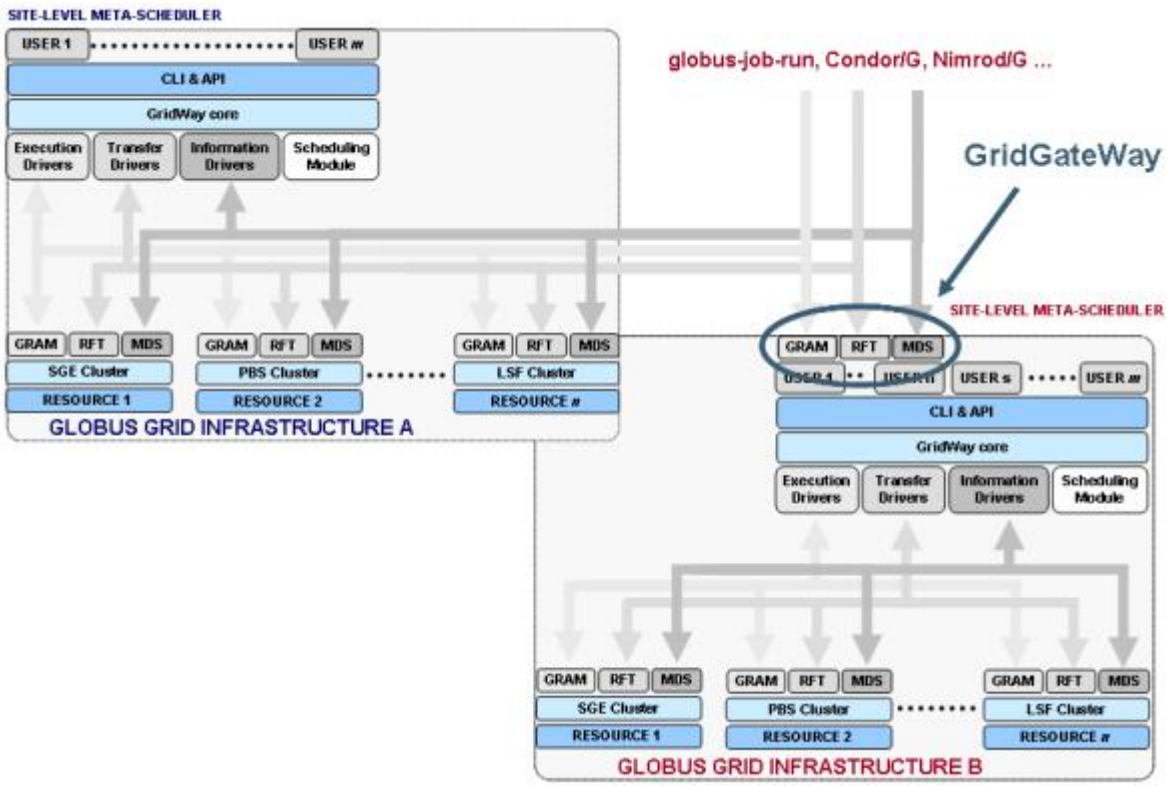


What is our proposal?

Grid technology overcomes utility computing challenges by means of its standard functionality for flexible integration of diverse distributed resources. The technological feasibility of the utility model for computing services can be established by using a grid infrastructure based on Globus Toolkit components and the GridWay metascheduler. It is well known that the GRAM service included in Globus Toolkit provides a uniform, secure and reliable interface to heterogeneous computing platforms managed by different local resource management systems. The main feature of our model is the use of

the GRAM, MDS and RFT services to recursively interface to the services available in a Globus based grid. A GRAM4 service hosting a GridWay metascheduler provides the standard functionality required to implement a gateway to a federated grid (a *GridGateWay*).Such a combination allows the required virtualization technology to be created in order to provide a powerful abstraction of the underlying grid resource management services. Such components are only a first step as other additional components for negotiation, service level agreement, accounting and billing may be required to deploy production-level utility solutions.

Figure 1-4. Deployment of GridGateWays



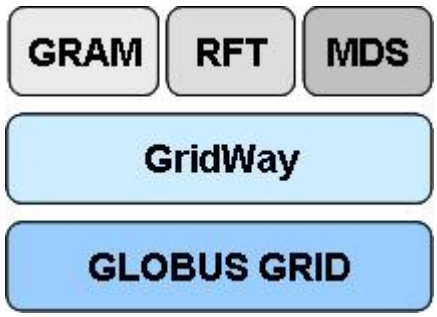
The grid hierarchy in our utility computing model is clear. An enterprise grid, managed by the IT Department, can access a GridGateWay to an outsourced grid, managed by a utility computing service provider. The outsourced grid provides pay-per-use computational power when local resources are overloaded; while security, accounting and billing are locally managed at the gateway of the outsourced grid, transparently to the end user. This hierarchical grid organization may be extended recursively to federate a higher number of partner or outsourced grid infrastructures with consumer/provider relationships.

Chapter 2. Installation guide

GridGateWay architecture

A GridGateWay follows the same architecture as typical Globus job managers, exposing GridWay functionality as a local DRM system.

Figure 2-1. Components of a GridGateWay.



The architecture of a GridGateWay consists of the following components:

- *GRAM4 job manager adapter* is a perl library providing an interface to GridWay.
- *GRAM4 scheduler event generator* checks GridWay logs to trigger events related to job state changes.
- *MDS4 scheduler information provider* checks GridWay status and reports it to the MDS.

Required software

GridGateWay is distributed as several source packages, required software to install it:

- Globus Toolkit 4.0, with the following active services:
  - GRAM4
  - MDS4
  - RFT (requires also GridFTP)
  - Delegation service (optional)
- Globus Packaging Tools (GPT)
- GridWay Metascheduler (multiple-user configuration is recommended)

> **Note:** From GridGateWay 1.0.4 onwards, GRAM2 (pre-WebServices) compatibility has been introduced, although it is not completely tested. Installation steps are the same as for GRAM4. If you find any problem interacting with GRAM2 and the GridGateWay, please send your feedback to *gridway-user@globus.org*.

Platform notes

Debian Testing

No known issues.

Fedora Core

No known issues.

Mac OS X 10.4

No known issues.

Solaris 10

No known issues.

Verifying Globus Toolkit installation

Since GridGateWay relies on Globus WS services, it is assumed that a Globus grid infrastructure has been installed and configured. You can perform the following tests to verify your Globus WS installation in the machine that will host GridGateWay:

1. Submission test:

   ```
   $ globusrun-ws -submit -F <GGW host> -s -c /bin/uname -a
   ```

   You should see the output of the **/bin/uname -a"** command (along with other information about submission progress).

2. File transfer test:

   ```
   $ globus-url-copy file://etc/hosts gsiftp://<GGW host>/tmp/hosts1
   ```

   ```
   $ globus-url-copy gsiftp://<GGW host>/tmp/hosts1 file://tmp/hosts2
   ```

   The contents of files /etc/hosts, /tmp/hosts1 and /tmp/hosts2 should be identical.

3. Information retrieval test:

   ```
   $ wsrf-query -x -s https://<GGW host>/wsrf/services/DefaultIndexService
   ```

   You should see a lot of information in XML format.

Change *<GGW host>* to the name of the host your want to test.

Verifying GridWay Metascheduler installation

In order to test the GridWay installation in the machine hosting the GridGateWay, please refer to "GridWay 5 Installation & Configuration guide".

> **Note:** GridWay multiple-user configuration in the GridGateWay is recommended, as single-user configuration needs all users to be mapped to the owner of the `gwd` process. For multiple-user configurations, make sure that local (grid-mapped) users can execute GridWay commands and user `globus` can execute the **gwhost** command and have read access to GridWay logs in `$GW_LOCATION/var`.

Installation

Log in to the machine that will host the GridGateWay as `globus` user account and follow these steps:

1. Download the distribution to the `globus` user `$HOME` directory (for example).
2. Unpack the distribution file and change to `ggw` directory (change {version} accordingly, i.e 1.0.0. This ggw directory is the source code directory and can be disposed of when the installation completes.):

```
$ tar xzf ggw-{version}.tar.gz
$ cd ggw
```

3. Setup Globus environment (if needed):

```
$ export GLOBUS_LOCATION=<path_to_Globus_location>
$ export GPT_LOCATION=<path_to_GPT_location>
$ . globus-user-env.sh
$ . globus-devel-env.sh
```

4. Setup GridWay environment (if needed):

```
$ export GW_LOCATION=<path_to_GridWay_location>
$ export PATH=$GW_LOCATION/bin:$PATH
$ export CLASSPATH=$GW_LOCATION/bin:$CLASSPATH
```

5. Build all GridGateWay packages:

```
$ gpt-build -force gcc32dbg globus_gram_job_manager_setup_gw.tar.gz \
    globus_scheduler_event_generator_gw.tar.gz \
    globus_scheduler_provider_setup_gw.tar.gz \
    globus_wsrf_gram_service_java_setup.tar.gz
```

6. **Optionally**, install the following Globus package (not included in the tarball, please find the one that corresponds with your installed Globus version) in GridGateWay web page) needed in order to support transparent file staging without using RSL/JDD extensions (see the Section called *File staging support* in Chapter 4 for more details):

```
$ gpt-build -force gcc32dbg globus_wsrf_gram_service_java.tar.gz
```

> **Note:** This is *not* needed for this from GT4.0.6 onwards, since the patch has been directly applied in the Globus Toolkit distribution.

7. Perform the post-installation steps:

```
$ gpt-postinstall -force
```

8. Restart the globus container.

Note: Proxy certificates for local (grid-mapped) users must be somehow created on the machine hosting the GridGateWay, since each GridWay command is executed by that user. This can be done manually (issuing a **grid-proxy-init** command on the GridGateWay host), with MyProxy or by delegating a full credential.

Note: To get delegation working in some distributions the sudoers configuration needs to be told to preserve the $HOME variable. Therefore, the following line should be added to /etc/sudoers:

```
Defaults        always_set_home
```

Verifying a GridGateWay installation

In order to test the GridGateWay installation, by means of full credential delegation, login as your user account and follow the steps listed below:

1. Delegate a full credential to the GridGateWay:

   ```
   $ globus-credential-delegate -h <GGW host> deleg.epr
   ```

   The EPR of the fully delegated credential is now stored in `deleg.epr`.

2. Submit a simple job to the GridGateWay using the previously delegated credential:

   ```
   $ globusrun-ws -submit -Jf deleg.epr -F <GGW host> -Ft GW -s -c /bin/uname -a
   ```

   You should see the output of the **/bin/uname -a** command (along with other information about submission progress).

Change *<GGW host>* to the name of the host your want to test.

Both delegation and submission can be achieved in the same step using the next command:

```
$ globusrun-ws -submit -J -F <GGW host> -Ft GW -s -c /bin/uname -a
```

Chapter 3. Configuration guide

Configuring the GridGateWay

Since most of the functionality of GridGateWay is provided by the GridWay metascheduler, please refer to the "GridWay 5 Installation & Configuration Guide". In this section we will primary disscus the configuration of the client hosts.

It is possible to make modifications in some scripts to adapt their behaviour to your expectations. For example, you can adapt the `globus-scheduler-provider-gw` shell script in `$GLOBUS_LOCATION/libexec`, in order to provide the desired information to MDS, or the `gw.pm` perl script, in order to modify the submission or cancellation actions.

The GridGateWay can be used in three different ways:

Using Globus Client Tools and API

In this case, the submission hosts have the same requirements that those needed to interact with any other GRAM service, in terms of security, file staging... So no additional configuration steps are needed

When using the Globus client interface, file staging can be effectively handled through RSL/JSDD extensions. However, in this case you may be interested in installing a GRAM patch to provide legacy support for RSL descriptions (i.e. use standard RSLs with the GridGateWay without any modification).

Using GridWay

GridWay provides specific support for GridGateWays, and it will generate tuned job descriptions when interacting with a GridGateWay. In this case, there are no additional requirements in the GridWay server-side, as the GateWay is used just as another GRAM service.

If you are primarily using the GridGateWay with GridWay you do not need to install the GRAM patch

Using GridGateWay Tools

Under development! The GridGateWay CLI allows a simple and straightforward use of the GRAM service by automatically generating RSL job descriptions and performing input/output file staging

Configuring GridWay to access a GridGateWay

Login as `gwadmin` user account (or the user account you use to administer GridWay) and follow these steps:

1. Configure an Information MAD to perform the discovery of the GridGateWay. For example:

   ```
   IM_MAD = static:gw_im_mad_static:-s <GGW host>:gridftp:ws
   ```

2. Configure an Execution MAD to access WS GRAM:

```
EM_MAD = ws:gw_em_mad_ws::rsl2
```

3. Configure a Transfer MAD to access GridFTP:

```
TM_MAD = gridftp:gw_tm_mad_ftp:
```

Chapter 4. User guide

As described previously, the GridGateWay can be used as any other GRAM service. In this section we will describe how to use the GridGateWay with the Globus Client API and tools; and in conjunction with GridWay.

File staging support

Different file transfer patterns can be implemented with the GridGateWay. The following list describes the alternatives:

Two-hop file transfers

In this case, files are transferred from the client host -or from any file server for all that matters- to the GridGateWay. Once the files are in the GridGateWay server, GridWay forwards them to the target resource. When the job finishes the output files are transferred back to the client (or file server) in the same way: first to the GridGateWay, and then to the client.

The way of handling file staging in GRAM4 doesn't suit GridGateWay needs, since it strips out the file staging information before it reaches the perl job manager (client-GridGateWay file staging is performed via RFT). Therefore is not possible for the GridGateWay to gather this information (needed to forward the input and output files the virtualized grid) in a standard way.

Figure 4-1. Two-hop file staging

There are two possibilities to solve this problem using GridGateWay:

1. One possibility is to use it with no changes at all in the Globus Toolkit installation. This will mean that all the jobs submitted to GRAM4 must convey the staging information using extensions (see the Section called *Extensions*).
2. Another possibility is to install the modified GRAM4 package. This will enable to use GRAM4 exactly as before with an encapsulated GridGateWay. GRAM4 will rename the file staging information in the perl job description instead of removing it. It will therefore create a `filestagein` and a `filestageout` in the perl job description to allow GridGateWay to access this information.

Note: In any case, GridWay will always work out-of-the-box because it will use the extensions anyway.

One-hop file transfers

In this case, files are transferred from the client host -or from any file server- directly to the target resource. All the staging information must be included in the extensions section of the RSL.

Important: This operation mode is only available through the Globus Client Tools, and NOT with GridWay.

Figure 4-2. One-hop file staging

Extensions

GridWay extends the RSL/JDD information for a job using the extensions mechanism provided. When GridWay detects that the underlying resource manager in a GRAM interface is "GW", it uses RSL extensions to embed information of the job template. Basically, whatever is between the <*gw*> and </*gw*> tags are going to be interpreted as name and value pairs for GridWay's job template. Please refer to GridWay's user guide (Chapter 3) for a list of valid job template variables that you can use in this extensions.

> **Note:** RSL stands for *Resource Specification Language*, which is the language to specify jobs in GRAM2 using LDAP filter sintax. JDD stands for *Job Description Document*, which is the language to specify jobs in GRAM4 using XML syntax. In this document we use RSL/JDD for historical reasons, although JDD is more appropriate.

You can try GridWay RSL/JDD extensions by submitting a Globus job to a GridGateWay. The following lines are an example of gw extensions:

```
<extensions>
<gw>
CPULOAD_THRESHOLD=50
SUSPENSION_TIMEOUT=900
REQUIREMENTS=HOSTNAME="*.es"
INPUT_FILES=input_file1.txt, input_file2.txt
OUTPUT_FILES=output_file.txt
</gw>
</extensions>
```

> **Note:** It is always important to bear in mind when constructing JDD documents that the end result has to be a XML document. This enforces us to be careful about certain symbols (&,<,>), that should be escaped as in all XML documents.

Logging

Since GridGateWay reporting and accounting facilities are provided by GridWay, please refer to "GridWay 5 Installation & Configuration Guide".

GridGateWay Limitations

The WS-GRAM interface to GridWay limits its functionality. There is an ongoing effort to tackle and solve this limitations, but GridGateWay this release doesn't allow:

- *Workflows.* There is no support in GRAM RSL to declare dependencies between jobs. Possible line of action would be to use the concept of multi-job.
- *Array jobs.* Same as above

Credential Handling

Whenever a job is sent to the GridGateWay with credential delegation (see the Section called *Verifying a GridGateWay installation* in Chapter 2 for more details on how to do this), the job manager performs the following operations to ensure a correct handling:

1. The certificate is copied to .globus/gwdelegcred.pem, this being relative to the mapped user home directory. This happens even if there was a previously present delegated credential.
2. A reference to this gwdelegcred.pem credential is added in the users *.gwrc* file, as the value for the environment variable X509_USER_PROXY. This ensures that the MADs loaded for this particular user is going to use this particular credential. You can find more information on MAD environment setup in the GridWay documentation.

Is worth noting that there is no possibility to send a job with credential delegation using gridway (when used as a client of the GridGateWay), you need to use the *globusrun-ws* command in order to do this. To avoid the necessity of logging into the GridGateWay server and renewing the credential by hand a recommended solution is to first send a job using the following command and then use gridway:

```
globusrun-ws -submit -J -F <GGW host> -Ft GW -s -c /bin/true
```

The GridGateWay will then use the credential delegated by the previous command.

Equivalence RSL - Job Template

GridGateWay needs to meet two different syntaxes, the one used by Globus in their RSL specification, and the one used by GridWay in its job template definition. There are subtle differences, mostly steaming from the fact that GridWay uses implicit staging when the path is relative in the EXECUTABLE and STD* fields, and Globus doesn't in their equivalent fields. The following table shows a one-to-one equivalence between Globus RSL and GridWay Job Template's significant fields. Some useful notes:

- In GridWay Job Template, files specified with absolute paths don't get staged.
- In the equivalence table, wDIR(some_tag) means that the content of some_tag in the original RSL (assuming it is a gsiftp url) gets stripped off its protocol, host and port bit, i.e., it is transformed to the absolute path where the file is supposed to exist in the remote host.
- In the same line, BASE(some_tag) means the basename of the content of some_tag in the original RSL, i.e., the filename.

Table 4-1. Field Equivalence RSL - JT

<directory> (all the relative paths are calculated w.r.t this one)	It is used to define the GridGateWay job directory in the GridGateWay server. Doesn't have an explicit value in GridWay's Job Template. If the job comes from GridWay, then it's the GridWay job experiment directory ($HOME/.gw_user_jobid) in the GridGateWay server. Alternatively, if it comes from a Globus client and declared explicitly, it is the path of the job directory in the GridGateWay server. If not declared it is set to {GLOBUS_USER_HOME}/.globus/{globus-job-id}.
`<fileStageIn> <transfer> <sourceURL>srcfile</sourceURL> <destinationURL>dstfile</destinationURL> </transfer> </fileStageIn>`	The `srcfile` is transferred by RFT to `dstfile` in the GridGateWay host. Each staging directive in this form is transform to: `INPUT_FILES=file://<directory>/<destinationURL>`
`<fileStageOut> <transfer> <sourceURL>srcfile</sourceURL> <destinationURL>dstfile</destinationURL> </transfer> </fileStageOut>`	The `srcfile` is transferred by RFT to `dstfile` in the client host. Each staging directive in this form is transform to: `OUTPUT_FILES=BASE(srcfile) URLDIR(srcfile)`. Where BASE() is the basename of the source URL, and URLDIR is the path in the source URL.
`<executable>executable_file</executable>`	There are two possibilities: if there is explicit stage in of the executable, then we have EXECUTABLE=BASE(`executable_file`). Conversely, if it is not present in the staging, then we have EXECUTABLE=`executable_file` directly.
`<stdin>stdin_file</stdin>`	If it is set and it is not "/dev/null" then `STDIN_FILE=stdin_file`. Otherwise it doesn't get set.
`<stdout>stdout_file</stdout>`	`STDOUT_FILE=stdout_file`
`<stderr>stderr_file</stderr>`	`STDERR_FILE=stderr_file`
`<argument>arg</argument>`	`ARGUMENTS=arg`
`<environment> <name>env_var</name> <value>env_val</value> </environment>`	`ENVIRONMENT=var="val".`

Examples

Before executing any joba, you need to delegate your credentials. This can be done once with the

`globus-credential-delegate` or each time you submit a job using the `-J` option of the `globusrun-ws` command. Please note that delegating a credential takes time, so if your going to submit several jobs use the first option.

To execute a program, without staging directives just use the following RSL:

```
<job>
<executable>/bin/echo</executable>
<argument>Hello World</argument>
<stdout>${GLOBUS_USER_HOME}/stdout</stdout>
<stderr>${GLOBUS_USER_HOME}/stderr</stderr>
</job>
```

The output should look like:

```
> globusrun-ws -submit -Jf deleg_cred.epr -F cepheus.dacya.ucm.es -Ft GW -s -f hello1.rsl
Submitting job...Done.
Job ID: uuid:7aae7eee-2b15-11dc-8945-000fea311626
Termination time: 07/06/2007 16:33 GMT
Current job state: Pending
Current job state: Active
Current job state: CleanUp-Hold
Hello World
Current job state: CleanUp
Current job state: Done
Destroying job...Done.
```

Now, if you want to get the standard output and error streams include the following staging directives. Note that you need to patch the GRAM service as described in the previous sections.

```
<job>
<executable>/bin/echo</executable>
<argument>Hello World</argument>
<stdout>${GLOBUS_USER_HOME}/stdout</stdout>
<stderr>${GLOBUS_USER_HOME}/stderr</stderr>

<fileStageOut>
 <transfer>

<sourceUrl>gsiftp://cepheus.dacya.ucm.es:2811/${GLOBUS_USER_HOME}/stdout&
lt;/ sourceUrl>
  <destinationUrl>gsiftp://cygnus.dacya.ucm.es:2811/home/ruben/stdout</
destinationUrl>
 </transfer>
</fileStageOut>
</job>
```

You should get

```
> globusrun-ws -submit -Jf deleg_cred.epr -F cepheus.dacya.ucm.es -Ft GW -s  -f hello2.rsl
Delegating user credentials...Done.
Submitting job...Done.
Job ID: uuid:2e0d5b22-2b16-11dc-ad34-000fea311626
Termination time: 07/06/2007 16:38 GMT
Current job state: Pending
Current job state: Active
Current job state: StageOut
```

```
Hello World
Current job state: CleanUp-Hold
Current job state: CleanUp
Current job state: Done
Destroying job...Done.
Cleaning up any delegated credentials...Done.
```

If you want to do the previous example using one-hop transfers, use the following RSL:

```
<job>
<executable>/bin/echo</executable>
<argument>Hello World</argument>
<extensions>
<gw>
STDOUT_FILE=gsiftp://cygnus.dacya.ucm.es:2811/home/ruben/stdout
</gw>
</extensions>
```

The output should be:

```
> globusrun-ws -submit -Jf deleg_cred.epr -F cepheus.dacya.ucm.es -Ft GW   -f hello3.rsl
Submitting job...Done.
Job ID: uuid:b879204a-2b19-11dc-a098-000fea311626
Termination time: 07/06/2007 17:03 GMT
Current job state: Pending
Current job state: Active
Current job state: CleanUp
Current job state: Done
Destroying job...Done.
```

Finally let us suppose a job that involves some file transfers. If you have installed the GRAM patch you can just use the RSL in a stantdard way:

```
<job>
<executable>/bin/gzip</executable>
<argument>passwd</argument>
<stdout>${GLOBUS_USER_HOME}/stdout</stdout>
<stderr>${GLOBUS_USER_HOME}/stderr</stderr>

<fileStageIn>
 <transfer>
  <sourceUrl>gsiftp://cygnus.dacya.ucm.es:2811/etc/passwd</sourceUrl>
  <destinationUrl>file:///${GLOBUS_USER_HOME}/passwd</destinationUrl>
 </transfer>
</fileStageIn>


<fileStageOut>

 <transfer>
  <sourceUrl>gsiftp://cepheus.dacya.ucm.es:2811/${GLOBUS_USER_HOME}/stdout</sourceUrl>
  <destinationUrl>gsiftp://cygnus.dacya.ucm.es:2811/home/ruben/stdout</destinationUrl>
 </transfer>

 <transfer>
  <sourceUrl>gsiftp://cepheus.dacya.ucm.es:2811/${GLOBUS_USER_HOME}/stderr</sourceUrl>
```

```
  <destinationUrl>gsiftp://cygnus.dacya.ucm.es:2811/home/ruben/stderr</destinationUrl>
 </transfer>

 <transfer>
  <sourceUrl>gsiftp://cepheus.dacya.ucm.es:2811/${GLOBUS_USER_HOME}/passwd.gz</sourceUrl>
  <destinationUrl>gsiftp://cygnus.dacya.ucm.es:2811/home/ruben/passwd.gz</destinationUrl>
 </transfer>

</fileStageOut>
</job>
```

The previous example using one-hop transfers (or if you did not installed the GRAM patch) can be written as:

```
<job>
<executable>/bin/gzip</executable>
<argument>passwd</argument>

<extensions>
<gw>
STDOUT_FILE=gsiftp://cygnus.dacya.ucm.es:2811/home/ruben/stdout
STDERR_FILE=gsiftp://cygnus.dacya.ucm.es:2811/home/ruben/stderr

INPUT_FILES=gsiftp://cygnus.dacya.ucm.es:2811/etc/passwd
OUTPUT_FILES=passwd.gz gsiftp://cygnus.dacya.ucm.es:2811/home/ruben/passwd.gz
</gw>
</extensions>

</job>
```

The output should be:

```
> globusrun-ws -submit -Jf deleg_cred.epr -F cepheus.dacya.ucm.es -Ft GW  -s  -f gzip2.rsl
Submitting job...Done.
Job ID: uuid:afd61ef4-2b1c-11dc-8a25-000fea311626
Termination time: 07/06/2007 17:25 GMT
Current job state: Pending
Current job state: Active
Current job state: CleanUp-Hold
Current job state: CleanUp
Current job state: Done
Destroying job...Done.
```

Troubleshooting

If you are having problems with a GridGateWay, check the errors listed below.

Moreover, since most functionality of GridGateWay is provided by GridWay, please refer also to "GridWay 5 Installation & Configuration Guide".

Job creation failed while submitting a job to the GridGateWay.

```
Submitting job...Failed.
globusrun-ws: Error submitting job
globus_soap_message_module: SOAP Fault
```

```
Fault code: soapenv:Server.userException
Fault string: java.rmi.RemoteException: Job creation failed.; nested exception is:
        org.globus.wsrf.NoSuchResourceException
```

The GridWay's GRAM service is not correctly configured. Look for errors when you build and install the `globus_wsrf_gram_service_java_setup` package.

The job failed when the job manager attempted to run it.

```
GRAM Job failed because the job failed when the job manager
attempted to run it (error code 17)
```

The GridWay's GRAM job manager adapter is not properly working. Look for errors when you build and install the `globus_gram_job_manager_setup_gw` package.