

GT 4.2.0 GRAM4: User's Guide

DRAFT

GT 4.2.0 GRAM4: User's Guide

Introduction

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

DRAFT

Table of Contents

1. Using GRAM4	1
1. Generating a valid proxy	1
2. Delegating credentials	1
3. Local resource managers interfaced by a GRAM4 installation	1
2. Job description overview	3
1. Job Description Schema Reference	3
2. Single-Job Description	3
3. Multi-Job Description	3
4. Staging Directives	4
5. Substitution Variables	5
6. Extensions	5
3. Submitting jobs	10
1. Simple interactive job	10
2. Simple batch job	10
3. Using a contact string	11
4. Streaming output	11
5. Using a job description	11
6. Using a contact string in the job description	12
7. Specifying a local resource manager	13
8. Job with staging	14
9. Specifying a local user id in the job description	15
10. Using substitution variables	16
11. Using custom job description extensions	16
12. Multi-Job	16
4. Getting information about jobs	19
1. Resource properties of a job	19
2. List of all jobs in a GRAM4 instance	21
5. Terminating jobs	22
1. Jobs in batch mode	22
2. Jobs in interactive mode	22
6. Job lifetime	23
1. Server-side settings that impact job lifetime	23
2. Client-side information	24
7. Job hold and release	25
8. Client-Side Generated Submission ID	27
9. Specifying SoftEnv keys in the job description	28
10. Job and process rendezvous	30
I. GRAM4 Commands	31
globusrun-ws	32
11. Troubleshooting	39
1. Troubleshooting tips	39
2. Java WS Core Errors	40
3. Errors	43
12. Known Problems in GRAM4	46
1. Known Problems	46
13. Usage statistics collection by the Globus Alliance	48
1. GRAM4-specific usage statistics	48
Glossary	49
Index	51

List of Tables

11.1. Java WS Core Errors	41
11.2. GRAM4 Errors	44

DRAFT

Chapter 1. Using GRAM4

1. Generating a valid proxy

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

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

2. Delegating credentials

There are three different uses of delegated credentials:

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

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

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

3. Local resource managers interfaced by a GRAM4 installation

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

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

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

² [../schemas/rft_types.html](#)

3.1. Finding available local resource managers

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

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

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

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

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

3.2. Finding the default local resource manager

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

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

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

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

Chapter 2. Job description overview

1. Job Description Schema Reference

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

2. Single-Job Description

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

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

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

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

3. Multi-Job Description

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

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

¹ ../schemas/gram_job_description.html

```

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

```

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

```

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

```

4. Staging Directives

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

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

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

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

5. Substitution Variables

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

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

Valid variables and their description:

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

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

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

6. Extensions

To allow adding features to GRAM4 while avoiding breaking compatibility between versions, an extensibility point was included in the job description schema. This appears as the `<extensions>` element at the bottom of a job description document. Starting with version 4.2.0 of the Globus Toolkit, GRAM4 will support both a number of specific

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

6.1. Supported Extensions

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

6.1.1. Condor specific parameters

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

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

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

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

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

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

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

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

6.1.2. PBS Node Selection Parameters

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

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

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

```

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

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

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

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

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

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

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

```

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

Here are some examples of using `resourceAllocationGroup`:

```

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

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

```

```

        <cpuCount>10</cpuCount>
    </resourceAllocationGroup>
    <resourceAllocationGroup>
        <hostType>ia64-compute</hostType>
        <hostCount>5</hostCount>
        <cpusPerHost>2</cpusPerHost>
    </resourceAllocationGroup>
</extensions>

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

```

6.2. Additional Extension Constructs

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

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

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

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

6.2.1. Simple String Parameters

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

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

```

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

```

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

```
'yahoo!'
```

6.2.2. String Array Parameters

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

For example:

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

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

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

6.2.3. Name/Value Parameters

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

For example:

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

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

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

6.3. Supporting custom extensions in the Perl adapter modules.

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

Chapter 3. Submitting jobs

1. Simple interactive job

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

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

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

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

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

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

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

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

2. Simple batch job

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

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

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

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

Again the job had been submitted to localhost

3. Using a contact string

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

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

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

4. Streaming output

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

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

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

5. Using a job description

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

Here is an example of a simple job description:

```
<job>  
  <executable>/bin/echo</executable>  
  <argument>this is an example_string </argument>
```

```

    <argument>Globus was here</argument>
    <stdout>${GLOBUS_USER_HOME}/stdout</stdout>
    <stderr>${GLOBUS_USER_HOME}/stderr</stderr>
</job>

```

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

```

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

```

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

Here is an example with more job description parameters:

```

<?xml version="1.0" encoding="UTF-8"?>
<job>
  <executable>/bin/echo</executable>
  <directory>/tmp</directory>
  <argument>12</argument>
  <argument>abc</argument>
  <argument>34</argument>
  <argument>this is an example_string </argument>
  <argument>Globus was here</argument>
  <environment>
    <name>PI</name>
    <value>3.141</value>
  </environment>
  <stdin>/dev/null</stdin>
  <stdout>stdout</stdout>
  <stderr>stderr</stderr>
  <count>2</count>
</job>

```

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

```

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

```

6. Using a contact string in the job description

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

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

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

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

Note

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

7. Specifying a local resource manager

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

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

7.1. Submitting to the default local resource manager

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

7.2. Submitting to a non-default local resource manager

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

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

Example: the following job will be submitted to Condor:

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

Or with a job description that contains a factoryEndpoint:

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

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

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

8. Job with staging

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

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

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

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

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

```
<job>
  <executable>my_echo</executable>
  <directory>${GLOBUS_USER_HOME}</directory>
  <argument>Hello</argument>
  <argument>World!</argument>
  <stdout>${GLOBUS_USER_HOME}/stdout</stdout>
  <stderr>${GLOBUS_USER_HOME}/stderr</stderr>
```

¹ ../schemas/rft_types.html

```

<fileStageIn>
  <transfer>
    <sourceUrl>gsiftp://job.submitting.host:2811/bin/echo</sourceUrl>
    <destinationUrl>file:///${GLOBUS_USER_HOME}/my_echo</destinationUrl>
  </transfer>
</fileStageIn>
<fileStageOut>
  <transfer>
    <sourceUrl>file:///${GLOBUS_USER_HOME}/stdout</sourceUrl>
    <destinationUrl>gsiftp://job.submitting.host:2811/tmp/stdout</destinationUrl>
  </transfer>
</fileStageOut>
<fileCleanup>
  <deletion>
    <file>file:///${GLOBUS_USER_HOME}/my_echo</file>
  </deletion>
</fileCleanup>
</job>

```

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

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

9. Specifying a local user id in the job description

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

```

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

```

10. Using substitution variables

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

For example:

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

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

11. Using custom job description extensions

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

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

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

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

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

12. Multi-Job

The job description XML schema allows for specification of a *multijob* i.e. a job that is itself composed of several executable jobs, which we will refer to as *subjobs* (*note*: subjobs cannot be multijobs, so the structure is not recursive).

This is useful for instance in order to bundle a group of jobs together and submit them as a whole to a remote GRAM installation.

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

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

Here is an example of a multijob description:

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

  <job>
    <factoryEndpoint>
      <wsa:Address>https://localhost:8443/wsrf/services/ManagedJobFactoryService</wsa:A
    </factoryEndpoint>
    <executable>/bin/date</executable>
    <stdout>${GLOBUS_USER_HOME}/stdout.p1</stdout>
    <stderr>${GLOBUS_USER_HOME}/stderr.p1</stderr>
    <count>2</count>
  </job>

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

Submit the multi-job with the following command:

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

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

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

**Note**

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

**Note**

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

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

DRAFT

Chapter 4. Getting information about jobs

1. Resource properties of a job

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

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

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

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

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

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

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

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

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

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

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

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

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

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

```
[martin@osg-test1 ~]$ wsrf-get-properties -e myJob.epr \
  {http://www.globus.org/namespaces/2008/03/gram/job/types}localUserId \
  {http://www.globus.org/namespaces/2008/03/gram/job/exec}localJobId

<ns1:localUserId xmlns:ns1="http://www.globus.org/namespaces/2008/03/gram/job/types">
  feller
</ns1:localUserId>
<ns2:localJobId xmlns:ns2="http://www.globus.org/namespaces/2008/03/gram/job/exec">
  51370.osg-test1.unl.edu
</ns2:localJobId>
```

Get the whole resource property document of a job:

```
[martin@osg-test1 ~]$ wsrf-query -e myJob.epr "/*"

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

```

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

```

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

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

2. List of all jobs in a GRAM4 instance

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

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

Chapter 5. Terminating jobs

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

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

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

1. Jobs in batch mode

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

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

2. Jobs in interactive mode

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

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

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

Chapter 6. Job lifetime

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

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

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

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

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

1. Server-side settings that impact job lifetime

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

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

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

Getting the value of the RP maxLifetime:

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

The result (in this case 1 year) is:

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

Getting the value of the RP jobTTLAfterProcessing:

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

The result (in this case 24h) is:

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

2. Client-side information

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

2.1. Specifying no lifetime in submission

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

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

2.2. Specifying a lifetime in submission

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

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

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

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

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

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

2.3. Setting a new lifetime on an existing job

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

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

The output could be something like this:

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

Chapter 7. Job hold and release

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

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

Valid hold states are:

- StageIn
- StageOut
- CleanUp
- Pending

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

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

Submitting the job in batch mode:

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

Checking for status:

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

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

Releasing the job:

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

Checking for status after the release:

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

Removing the job:

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

DRAFT

Chapter 8. Client-Side Generated Submission ID

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

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

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

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

Chapter 9. Specifying SoftEnv keys in the job description

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

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

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

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

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

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

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

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

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

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

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

Note

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

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

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

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

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

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

Note

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

Chapter 10. Job and process rendezvous

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

¹ ../../wsrendezvous/

GRAM4 Commands

DRAFT

Name

globusrun-ws -- Official job submission client for GRAM4

```

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

```

Description

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

Command options

Quiet mode

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

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

Debug mode

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

Protocol Options

Service authorization

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

- | | |
|--|--|
| -host, -host-authz | The GSI "host authorization" rule is used to verify that the service is using a host credential appropriate for the underlying service address information. This is the default. |
| -self, -self-authz | The GSI "self authorization" rule is used to verify that the service is using a (proxy) credential derived from the same identity as the client's. |
| -subject, -subject-authz <u>subject name</u> | The service must be using a credential with the exact subject name provided by this option. |

Security Protocol

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

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

Timeouts

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

Signal handling

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

Submit options

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

Output Mode

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

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

Streaming Options

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

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

Factory information

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

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

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

Default values form the following contact information if not overridden:

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

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

Job description

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

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

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

Submission ID

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

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

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

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

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

Job EPR output

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

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

Delegation

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

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

- J, -job-delegate If supplied AND the job description does not already provide a jobCredential element, globusrun-ws will delegate the client credential to GRAM4 and introduce the corresponding element to the submission input.

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

This option is implied by -streaming

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

note: for multijob descriptions, only the top level jobCredential will be copied into.

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

note: this option is ignored for multijobs.

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

note: this option is ignored for multijobs.

Lifetime

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

-term, -termination mm/dd/yyyy Set an absolute termination time.
HH:MM

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

Validate options

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

Job description

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

Monitor options

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

Job

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

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

Output mode

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

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

Status options

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

See the Job options for the -monitor command.

Kill options

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

Help options

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

Usage options

-usage Outputs brief usage information for the command.

Version options

-version Outputs version information for the command.

Job Handling

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

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

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

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

Environment

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

Exit Codes

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

Chapter 11. Troubleshooting

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

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

1. Troubleshooting tips

In case you run into problems you can do the following

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

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

- Check the container log for errors.

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

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

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

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

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

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

2. Java WS Core Errors

DRAFT

Table 11.1. Java WS Core Errors

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

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

DRAFT

Table 11.2. GRAM4 Errors

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

Chapter 12. Known Problems in GRAM4

1. Known Problems

The following problems and limitations are known to exist for GRAM4 at the time of the 4.2.0 release:

1.1. Limitations

- [list limitations]

1.2. Outstanding bugs

- [Bug 2250](http://bugzilla.globus.org/globus/show_bug.cgi?id=2250):¹delegation required resource property
- [Bug 2578](http://bugzilla.globus.org/globus/show_bug.cgi?id=2578):²reliable state change notification
- [Bug 2579](http://bugzilla.globus.org/globus/show_bug.cgi?id=2579):³reliable state change notification
- [Bug 2623](http://bugzilla.globus.org/globus/show_bug.cgi?id=2623):⁴service summary/diagnostics
- [Bug 2624](http://bugzilla.globus.org/globus/show_bug.cgi?id=2624):⁵Multiple job hold states and parameterized release operation
- [Bug 2629](http://bugzilla.globus.org/globus/show_bug.cgi?id=2629):⁶performance timings in globusrun-ws
- [Bug 2734](http://bugzilla.globus.org/globus/show_bug.cgi?id=2734):⁷non-shared FS scheduler file list
- [Bug 3088](http://bugzilla.globus.org/globus/show_bug.cgi?id=3088):⁸Default Job Environment
- [Bug 3242](http://bugzilla.globus.org/globus/show_bug.cgi?id=3242):⁹Software selection thru WS GRAM RSL
- [Bug 3569](http://bugzilla.globus.org/globus/show_bug.cgi?id=3569):¹⁰Selectable jobType default per factory
- [Bug 3575](http://bugzilla.globus.org/globus/show_bug.cgi?id=3575):¹¹SEG dependent on GLOBUS_LOCATION env var
- [Bug 3948](http://bugzilla.globus.org/globus/show_bug.cgi?id=3948):¹²Service must release all of its resources on deactivation
- [Bug 4009](http://bugzilla.globus.org/globus/show_bug.cgi?id=4009):¹³Use pbsdsh if available
- [Bug 4181](http://bugzilla.globus.org/globus/show_bug.cgi?id=4181):¹⁴Allow File Staging To/From globusrun-ws application without an external server

¹ http://bugzilla.globus.org/globus/show_bug.cgi?id=2250

² http://bugzilla.globus.org/globus/show_bug.cgi?id=2578

³ http://bugzilla.globus.org/globus/show_bug.cgi?id=2579

⁴ http://bugzilla.globus.org/globus/show_bug.cgi?id=2623

⁵ http://bugzilla.globus.org/globus/show_bug.cgi?id=2624

⁶ http://bugzilla.globus.org/globus/show_bug.cgi?id=2629

⁷ http://bugzilla.globus.org/globus/show_bug.cgi?id=2734

⁸ http://bugzilla.globus.org/globus/show_bug.cgi?id=3088

⁹ http://bugzilla.globus.org/globus/show_bug.cgi?id=3242

¹⁰ http://bugzilla.globus.org/globus/show_bug.cgi?id=3569

¹¹ http://bugzilla.globus.org/globus/show_bug.cgi?id=3575

¹² http://bugzilla.globus.org/globus/show_bug.cgi?id=3948

¹³ http://bugzilla.globus.org/globus/show_bug.cgi?id=4009

¹⁴ http://bugzilla.globus.org/globus/show_bug.cgi?id=4181

- [Bug 4791](#):¹⁵ Job state history RP
- [Bug 5017](#):¹⁶ gram[24] tests that need to be updated
- [Bug 5402](#):¹⁷ invalid password error messages
- [Bug 5433](#):¹⁸ public interface doc lists non-public/internal APIs
- [Bug 5484](#):¹⁹ Review and Update 4.2 GRAM doc
- [Bug 5510](#):²⁰ Reduction of notification consumers in WS-GRAM RFT interaction
- [Bug 5745](#):²¹ Allow users to specify JDD in a different order than RSL schema
- [Bug 5805](#):²² Change threadpools from Gram custom implementation to java.util.concurrent
- [Bug 5853](#):²³ create automated tests for globus-job-*-ws programs
- [Bug 6019](#):²⁴ CEDPS: Add executable path to log statement
- [Bug 6024](#):²⁵ GRAM Audit v2
- [Bug 6102](#):²⁶ GRAM4 throughput tester for 4.2
- [Bug 6109](#):²⁷ Add ability for setuid programs to be plugged into GRAM4
- [Bug 6110](#):²⁸ Add comments to GRAM4 service code for audit v2
- [Bug 6172](#):²⁹ Bad error message for file not found

¹⁵ http://bugzilla.globus.org/globus/show_bug.cgi?id=4791

¹⁶ http://bugzilla.globus.org/globus/show_bug.cgi?id=5017

¹⁷ http://bugzilla.globus.org/globus/show_bug.cgi?id=5402

¹⁸ http://bugzilla.globus.org/globus/show_bug.cgi?id=5433

¹⁹ http://bugzilla.globus.org/globus/show_bug.cgi?id=5484

²⁰ http://bugzilla.globus.org/globus/show_bug.cgi?id=5510

²¹ http://bugzilla.globus.org/globus/show_bug.cgi?id=5745

²² http://bugzilla.globus.org/globus/show_bug.cgi?id=5805

²³ http://bugzilla.globus.org/globus/show_bug.cgi?id=5853

²⁴ http://bugzilla.globus.org/globus/show_bug.cgi?id=6019

²⁵ http://bugzilla.globus.org/globus/show_bug.cgi?id=6024

²⁶ http://bugzilla.globus.org/globus/show_bug.cgi?id=6102

²⁷ http://bugzilla.globus.org/globus/show_bug.cgi?id=6109

²⁸ http://bugzilla.globus.org/globus/show_bug.cgi?id=6110

²⁹ http://bugzilla.globus.org/globus/show_bug.cgi?id=6172

Chapter 13. Usage statistics collection by the Globus Alliance

1. GRAM4-specific usage statistics

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

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

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

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

¹ ../../Usage_Stats.html

Glossary

C

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

G

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

J

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

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

L

LSF A job scheduler mechanism supported by GRAM.

For more information, see <http://www.platform.com/Products/Platform.LSF.Family/Platform.LSF/>⁷.

M

Managed Executable Job Service (MEJS) [FIXME]

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

See also [MMJS subjob](#)¹⁰.

P

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

R

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

⁴ #scheduler

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

¹⁰ #mmjs-subjob

S

scheduler

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

U

Universally Unique Identifier (UUID)

Identifier that is immutable and unique across time and space.

W

Web Services Addressing (WSA)

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

¹⁴ <http://www.w3.org/2002/ws/addr/>

Index

B

bugs
outstanding, 46

C

command line client
globusrun-ws, 32

E

errors, 40, 43

J

jobs
hold and release, 25
lifetime, 23
client-side settings, 24
server-side settings, 23
preparing
delegating credentials, 1
finding available LRMs, 2
finding default LRM, 2
generate valid proxy, 1
querying
all jobs in a GRAM4 instance, 21
resource properties, 19
rendezvous, 30
SoftEnv keys, 28
submission ID, 27
submitting
multijob, 16
request streaming output, 11
simple batch, 10
simple interactive, 10
specifying LRM, default, 13
specifying LRM, non-default, 13
using a job description, 11
using a job description with contact string, 12
using contact string, 11
using custom job description extensions, 16
using substitution variables, 16
with job description, specifying local user, 15
with staging, 14
terminating, 22
batch mode, 22
interactive mode, 22

L

limitations, 46

S

submitting jobs
globusrun-ws, 32

T

troubleshooting, 39
check container log, 39
check documentation, 39
errors, 39
mailing lists, 39

U

usage statistics, 48