

Globus Toolkit 5.0 Migration Guide

A Guide for Migrating GT4.0 services in D-Grid to GT5.0

Leibniz Rechenzentrum

Authors:

Table of Contents

Overview	3
1 Globus Components	4
1.1 Job Submission	4
1.1.1 GRAM	4
1.1.2 WS-GRAM	5
1.1.3 Job Submission Languages (RSL).....	5
1.1.4 Job Submission through Gridway	5
1.2 File Transfer	6
1.2.1 GridFTP	6
1.2.2 RFT	7
1.3 GSI-OpenSSH	9
1.4 MDS	9
2 Web Services	10
3 Installation of Globus 5.0	11
3.1 Compilation from source.....	11
3.2 GRAM5.....	12
3.2.1 PBS or Torque/Maui	12
3.3 GridFTP	12
3.4 GSI-OpenSSH	13
4 Other TOOLS/APIs	14
4.1 JavaGAT	14
4.2 GWES	14
4.3 Java CoG-Kit.....	14
5 References	15
Acknowledgments	16
Appendix	17
Job description language translation script:.....	17

Overview

This document endeavours to provide a picture of the status of Globus 5.0 and its compatibility with the various Globus related components for the D-Grid resource providers and the D-Grid communities. Based on the questionnaire sent to the D-Grid communities we have tried to address a number of issues that can be addressed at the moment. Since Globus 5 is being actively developed, this guide will be updated in the future to reflect the latest updates in order to facilitate even more the migration of the communities.

For any comments/requests you can contact the authors.

1 Globus Components

1.1 Job Submission

1.1.1 GRAM

The job submission service for Globus (GRAM) has been updated and currently called GRAM5. This is known to be backwards compatible with GRAM2 and follows a similar structure. A gatekeeper service is deployed for authentication and setting up of the Globus Job Management Service. Below we provide differences as per the migration guide provided by Globus here:

<http://www.globus.org/toolkit/docs/5.0/5.0.3/execution/gram5/mig/> [1]

1.1.1.1 Differences From GRAM2

In GRAM2, jobs are submitted to a job manager process started by a Gatekeeper process. In GRAM5, the job-submit protocol is the same; however, all jobs for a particular user and LRM run in the same job manager process.

1.1.1.2 Security Differences

1.1.1.2.1 Proxies and Delegation

In GRAM2, the GRAM client is required to delegate a proxy credential to the Gatekeeper so that the job manager can send authenticated job state change messages. Because each job is monitored by a separate job-manager in GRAM2, each job-manager has access to a different delegated credential. In GRAM5, the shared job manager uses the delegated credential with the latest expiration time as its credential.

In GRAM2, the client can control the job manager proxy timeout by setting the value of `proxy_timeout` RSL attribute to a time interval in seconds indicating when the job manager will exit if the proxy is about to expire. In GRAM5, this is a job manager-wide setting and the `proxy_timeout` RSL attribute is ignored.

1.1.1.3 Network Communication

In GRAM2, the standard output and standard error streams of a job may be sent in near real time to a file server such as the GASS server embedded in a globusrun execution. In GRAM5 the same RSL syntax is used to name output and error stream destinations, but those are not sent until after the job execution is complete.

In GRAM2, the job-manager implements intra-job communication via DUCT and task synchronization via DUROC. These features have been dropped in GRAM5. GRAM5 adds various protocol extensions to the GRAM2 protocol. This is done in a way such that the existing GRAM2 protocol processors will ignore the extensions to the messages. Details about the protocol and its extensions can be found in the [GRAM5 public interface document](#) [2].

1.1.2 WS-GRAM

WS-GRAM, corresponds to the Web Services based job submission. In GT4 it was possible to submit jobs to the container, which handled authentication delegation and job submission. These web service based services are no longer available to the users, as the container is gone. Job submission to WS-GRAM was performed in the following ways:

1.1.2.1 The globus command-line tools for job submission

This kind of use case can easily be migrated to its GT5 counterpart. Below we present a mapping between two common commands `globusrun-ws` (web services based job submission) and `globusrun` (`gram5` job submission).

Command	Example
<code>globusrun-ws</code>	<code>globusrun-ws -Ft <factory> -submit -F <resource> -s -f job.xml</code>
<code>globusrun</code>	<code>globusrun -o -r <resource> -f job.rsl</code>

The resource is a URL describing the location of the service to submit the job to. In GT4 this was something like `lxgt2.lrz-muenchen.de` and the specific resource to submit the job to was defined by the “-Ft” argument -e.g. “-Ft SGE”. In GRAM5 this can be directly added in the resource URL. For example “`lxgt2.lrz.de/jobmanager-sge`”. The differences between the job submission syntaxes from GRAM4 to GRAM5 are described in section 1.1.2.3.

1.1.2.2 Job Submission through Gridway

GRAM5 is already supported in GridWay. GridWay allows unattended, reliable, and efficient execution of single, array, or complex jobs, both sequential and parallel, on heterogeneous and dynamic grids. The work necessary to allow for the successful cooperation between Globus 5 and Gridway is undertaken by IGE. In order to find out more please contact: support@ige-project.eu.

1.1.2.3 Job Submission using external tools

There were several external ways to submit jobs to Globus resources in D-Grid. These are covered in Section 4.

1.1.3 Job Submission Languages (RSL)

The job submission language for GRAM4 is an XML-based language. It differs syntactically from the RSL language used in GRAM2, although many of the elements are common. In GRAM5 the RSL language is identical to GRAM2 and compatible. GRAM5 RSL offers similar functionality but currently lacks support for a few GRAM4 relevant elements, so a direct translation may not always be possible. Some of the elements that are not compatible at the moment of this writing can be found in Table 1.

GRAM4 elements not in GRAM5	
factoryEndpoint	Not available. Submission endpoint only through command line.
jobCredentialEndpoint	Not available
multijob	Not supported
serviceLevelAgreement	Not available
stagingCredentialEndpoint	Not available

Table 1: Incompatible elements between GRAM4 and GRAM5 job description languages

To aid the transition, a script has been compiled that translates from the XML-based language to GRAM5 RSL and informs the user of any problems or issues that are related to the translation. The script is provided in the Appendix. A brief usage note is available simply by using the ‘-h’ argument, as shown below.

```

:~> jdl2rsl -h
usage: jdl2rsl [options]

options:
  -h, --help                show this help message and exit
  -j JDL_FILE, --jdl-file=JDL_FILE
                             File containing the GT4 job description. Use '-'
for
                             stdin
  -n, --no-extensions       Do not translate extensions.

```

The script will try to match all GRAM4 elements to their GRAM5 counterparts and in case any unsupported elements are found, the user is informed via stderr about the status. The ‘multijob’ elements are also translated and a multijob RSL file is created. Currently though, it is not possible to submit multijob requests to GRAM5 (although the RSL will be considered correct).

In case there are extensions in the RSL file, those extensions are also translated by default. If this is not desired, the translation can be disabled using the ‘-n’ flag. Finally if an output file is not given as an argument (via the ‘-f’ flag) the RSL file is output to stdout.

1.2 File Transfer

1.2.1 GridFTP

The GridFTP service has been updated and greatly improved in GT5. The protocol and APIs have not changed so any combination of old/new server/client will work. The latest version of the GridFTP server (bundled with GT5.0.4) includes the following improvements:

- globus-url-copy can recover from server and network failures (-rst options)
- Improved restart capabilities for file transfers:
 - Restart transfers from checkpoints.
- Stall detection (-st option)
 - Specify how long before cancelling/restarting transfers without data movement
- Client-Side host aliasing
 - Allows concurrent transfers to be extended to multiple different hosts specified in an alias file.

- Synchronization capabilities
- Multiple credential support
- New Data Channel Security Context (DCSC). This command is useful for 3rd party transfers between GridFTP servers with different CA certificates.
- GridFTP server chrooting. This allows the server to run under chroot which restricts the directories the gridftp server process can access.
- GWFTP (GridFTP where there is FTP) to allow simple FTP clients to connect to GridFTP servers.

1.2.2 RFT

The RFT service has been discontinued and no clients/servers are included in Globus Toolkit 5. In D-Grid we have found that the communities where using RFT in the following ways:

1.2.2.1 Via attributes in the job description languages

Most of the projects in D-Grid make use of RFT via the fileStageIn/fileStageOut attributes in the job description files. The migration of these cases poses no great challenge as the RSL used in GRAM5 can handle the aforementioned attributes by passing the transfers to GridFTP. Using the converter presented in the previous chapter (section 1.1.2.3) the transfers will be handled without any further user intervention.

1.2.2.2 Via the Globus 4 Java CoG-Kit

For the projects that use the Java CoG-Kit file transfer capabilities are still available using GridFTP. See section 4.3 for more information.

1.2.2.3 Via the globus command-line tools

Those making use of the command line tool 'rft' will not be able to access this command anymore. Possible solutions for reliable file transfer include:

1.2.2.3.1 The 'globus-url-copy' command

The 'globus-url-copy' command has been updated in GT5 to use the various options that offer higher reliability of GridFTP transfers. The command can be used with the following reliability options (taken from the user's guide):

-rst | -restart

Restarts failed FTP operations.

-rst-retries <retries>

Specifies the maximum number of times to retry the operation before giving up on the transfer.

Use 0 for infinite.

The default value is 5.

-rst-interval <seconds>

Specifies the interval in seconds to wait after a failure before retrying the transfer.

Use 0 for an exponential backoff.

The default value is 0.

-rst-timeout <seconds>

Specifies the maximum time after a failure to keep retrying.
Use 0 for no timeout.
The default value is 0.

-df <filename> | -dumpfile <filename>

Specifies path to the file where untransferred urls will be saved for later restarting. The resulting file is the same format as the -f input file. If the file exists, it will be read and all other url input will be ignored.

-do <filename> | -dump-only <filename>

Perform no write operations on the destination. Instead, all files that would be transferred are enumerated and dumped to the specified file. Resulting file is the same format as the -f input file. Note: if you intend to use this file as input for a future transfer, the -create-dest option will be required if any destination directories do not already exist.

-stall-timeout | -st <seconds>

Specifies how long before cancelling/restarting a transfer with no data movement. Set to 0 to disable. Default is 600 seconds.

The aforementioned options can be used in small scripts in order to provide much better reliability for transfers. An example, also provided in the User's guide:

```
#!/bin/sh
STATEFILE=/path/to/statefile;
while [ ! -e $STATEFILE -o -s $STATEFILE ];
do
  globus-url-copy -rst -p 4 -cc 4 -cd -vb -r -df $STATEFILE
  gsiftp://srchost/srcdirpath/ gsiftp://dsthost/dstdirpath/;
  sleep 10;
done;
```

The above would allow you to initiate a transfer, which will continue automatically where it was left if something goes wrong by checking the state file created. When the state file is empty the transfer can finish.

1.2.2.4 Globus Online

Globus Online is a new service provided by Globus whose target is to provide robust and reliable file transfers. Transfers can be initiated between endpoints (i.e. GridFTP servers that the user has access to, or his own PC using Globus Connect^{1,2}). To use the service one can use a web browser or the command line by connecting to the available command-line-interface at 'cli.globusonline.org'. Globus Online also offers the possibility to use the Transfer REST API to initiate and monitor transfers and other aspects of the Globus Online functionality.

Some key features:

- Performance optimization: Globus Online tries to optimize the transfer based on the number and size of files involved.
- Error handling and retry: The transfers are restarted automatically, when the connection is broken or no data has been sent for a while.
- Monitoring: The user can monitor the activity and performance of the transfer on demand.

¹ Globus Connect is an application available for Mac, Windows and Linux that allows the https://www.globusonline.org/globus_connect/

² Transfers between two Globus Connects are not possible.

- **Endpoint Management:** Currently only GridFTP endpoints are supported but further protocols will be added in the future. The users can also define what credentials to use with their endpoints.
- **Conditional Transfers and Integrity Checking**
- **Synchronization:** Only files that have changed will be uploaded. The user can select the criteria for the synchronization feature (exist, size, timestamp, checksum).
- **Notifications:** The user can be notified on transfer completion, expiration of deadline or expiration of credentials.

There are numerous tutorials and ample documentation online. For more information please refer to: <https://www.globusonline.org/resources/>.

1.3 GSI-OpenSSH

Globus has been using GSI-OpenSSH which is a patched version of the OpenSSH tool to provide interactive access to resources using the GSI (Grid Security Infrastructure). The GSI-OpenSSH has been updated with various security fixes from GT4 to GT5 but remains compatible. There should be no migration concerns for GSI-OpenSSH.

1.4 MDS

In GT4 the important task of resource monitoring was taken by the MDS component. This component was also discontinued and so far there is no available component in production to replace it. The Globus alliance has been working on a new component called IIS (Integrated Information Services), which targets to undertake some of the operations of MDS. IIS is essentially a registry of Metadata representations of resources, which can be updated frequently and exported in various formats (including GLUE). These registries are XML-based and can be accessed via a REST interface. Its main target is to facilitate service advertisement and discovery and this using a distributed architecture. Currently IIS is in alpha stage, its latest version at the time of this writing is 0.3.0 and can be found in the main webpage:

<http://www.ci.uchicago.edu/wiki/bin/view/IIS/WebHome>

It is recommended that communities making heavy use of MDS contact us about their specific requirements, so that we can forward them to the developers in order to understand the possibilities available for migration.

From the questionnaire it seems that there are four kinds of users for MDS:

4. **Web Browser to view resources:** This is by accessing WebMDS. There is no component available as a replacement for this. IIS could be used to provide this information to a web application, which would visualize it for the user. IIS can provide XML formats of the registries and therefore it should be relatively easy to create such a web application.
4. **Command-line tool to access and further process the XML data:** This seems to be the least complicated case to migrate as IIS can provide this XML data to a command-line tool through the REST interface. It would be perhaps necessary to modify the data processing, as it is likely that the schema would differ from the old one.
4. **GridWay:** GridWay does not depend on MDS to retrieve job information for Globus5. Currently it supports job monitoring via synchronous or asynchronous

polling of GT5 resources. Information about the execution environment is provided through a static host file.

4. **Other tool:** Currently only AstroGrid is using Stellaris to communicate with MDS. The possibility of Stellaris using IIS will be investigated in the near future.

2 Web Services

In Globus 5 there are no more web services components available. The web service core components are gone along with web service authorization and authentication. The globus container is also not part of the Globus 5.0 release. Due to the above, it is not possible to deploy Globus-based web services. Currently it is possible to use GSI (Grid Security Infrastructure) with more modern Java security and web services stacks such as JAX-WS/Apache CXF using jGlobus 2.0. Other aspects and possibilities of service deployment are currently under investigation.

3 Installation of Globus 5.0

The installation of Globus 5.0 is somewhat simpler compared to its predecessor. Here we will provide instructions for a typical installation of the Globus 5.0 components that will be necessary for D-Grid. The contents of this section are likely to become further reduced with the 5.2 releases where the Globus middleware will be provided packaged for various Linux distributions and will follow the filesystem hierarchy standard.

The topics covered below assume that the host and CA certificates reside under ‘/etc/grid-security’. The initial section 3.1, should be used as an initial step and further sections regarding components assume that the commands in the aforementioned section have been successful.

In any case it is recommended to read carefully the D-Grid operational concept.

3.1 Compilation from source

A first step is to download the latest source installer from here:

<http://globus.org/toolkit/downloads/latest-stable/>

[3]

At the time of this writing the latest-stable version is 5.0.4 and the downloaded installer is called ‘gt5.0.4-all-source-installer.tar.bz2’. We perform the following steps:

```
1) mkdir /opt/globus-5.0.4
2) tar -jxf gt.5.0.4 -C /opt/globus-5.0.4
3) export GLOBUS_LOCATION=/opt/globus-5.0.4
4) cd globus-5.0.4-all-source-installer
5) ./configure -prefix=$GLOBUS_LOCATION --with-gsiopensshargs="--with-pam --with-tcp-wrappers" && make && make install
```

This should take care of the initial steps of the installation. For more information about specific distributions please refer to the installation guide ([4]). In order to have services available for D-Grid users the necessary ports should be open on the resources. These should adhere to the following port recommendation:

- Globus 5 Gatekeeper: Port 2119/tcp
- Globus 5 GridFTP: Port 2811/tcp
 - For file transfers: Port range 20000-25000
- Globus 5 GSI-OpensSSH: Port 2222

Also it is important that the following environment variables are set for all users allowed to access the Globus resources and for the invocation of the startup scripts:

- GLOBUS_LOCATION
- GLOBUS_TCP_PORT_RANGE
- LD_LIBRARY_PATH: Should contain the \$GLOBUS_LOCATION/lib

One possibility is to set these variables in a file in the folder ‘/etc/profile.d’.

3.2 GRAM5

The job submissions are handled by the globus gatekeeper service. For GRAM5 the gatekeeper service runs by default on port 2119, which differs from the GRAM4 port (8443) and thus installation can be done in parallel without problems. The GRAM5 gatekeeper runs on the same port as the GT4 gatekeeper but as it is backwards compatible replacing the service with GRAM5 should pose no problems.

3.2.1 PBS or Torque/Maui

To enable the gatekeeper to work with PBS, the environment variable PBS_HOME and relevant paths should be set. From the build folder above run the following commands:

```
% make gram5-pbs
...
% make install
...
```

This will install and configure the connectivity between the job submission service and the batch system. The endpoint for the service is defined by appending ‘/jobmanager-pbs’ to the host URL.

To setup automatic startup/shutdown of the service there are several ways. We describe below how to use xinetd for this purpose.

- 1) Make sure that the entry ‘gsigatekeeper 2119/tcp’ exists in the ‘/etc/services’ file.
- 2) Create a file in ‘/etc/xinetd.d’ called gsigatekeeper with the following contents:

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

3.3 GridFTP

GridFTP in GT5 is backwards compatible and thus no special care is needed for the migration to the new service.

- 1) Make sure that the entry ‘gsiftp 2811/tcp’ exists in the ‘/etc/services’ file.
- 2) Create a file in ‘/etc/xinetd.d’ called gsiftp with the following contents:

```
service gsiftp
{
    socket_type = stream
    protocol = tcp
    wait = no
    user = root
    env = LD_LIBRARY_PATH=GLOBUS_LOCATION/lib
    server = GLOBUS_LOCATION/sbin/globus-gridftp-server
```

```
server_args = -i
log_on_success = DURATION
nice = 10
disable = no
}
```

3.4 GSI-OpenSSH

GIS-OpenSSH is also backwards compatible. The administrator only needs to take care of the startup scripts. One can use as a default the script 'sbin/SXXsshd' under \$GLOBUS_LOCATION. It is recommended that the port is changed to 2222 and that a link is created:

\$GLOBUS_LOCATION/sbin/sshd to point to \$GLOBUS_LOCATION/sbin/gsisshd
This will make it easier to distinguish messages logged by the GSI-OpenSSH server from those logged by the OpenSSH server.

The following options for the server are recommended and can be set in:
\$GLOBUS_LOCATION/etc/ssh/sshd_config

```
PasswordAuthentication no
PubkeyAuthentication no
ChallengeResponseAuthentication no
UsePAM yes
UsePrivilegeSeparation yes
Port 2222
```

4 Other TOOLS/APIs

4.1 JavaGAT

Thanks to the help of Cerial Jacobs, the latest JavaGAT has been tested on the IGE testbed resources successfully. To be more precise:

- The globus resource broker adaptor tests were successful.
- The gridftp file adaptor tests were successful.
- The wsgt4new and gt42 adaptors will not work, since the web services are not available.

4.2 GWES

The GT4 WS-based plugins for GWES will not work with Globus 5.0 resources. There are plugins available that use the Pre-WS components, and through a collaborative work between IGE and Pneumogrid, SHIWA these plugins are going to be tested on the IGE testbed. Thanks to Andreas Hoheisel, new plugins supporting Globus 5.0 resources have been implemented. These new plugins will be tested as well on the IGE testbed.

4.3 Java CoG-Kit

According to the 5.0.4 release notes [5], the GRAM development team is using Java CoG-Kit 1.6 to perform their tests. Essentially the jGlobus APIs which are part of the CoG-Kit can be used with GT5 resources.

5 References

- 1 Globus Alliance. GT 5.0.3 Migrating Guide for GRAM5. [Internet]. Available from: <http://www.globus.org/toolkit/docs/5.0/5.0.3/execution/gram5/mig/#gram5Mig>.
- 2 Globus Alliance. GT 5.0.3 Component Guide to Public Interfaces: GRAM5. [Internet]. [cited 2011 March 28]. Available from: <http://www.globus.org/toolkit/docs/5.0/5.0.3/execution/gram5/pi/#gram5-interface-protocol>.
- 3 Globus Alliance. Latest Stable Globus Release. [Internet]. 2011 Available from: <http://globus.org/toolkit/downloads/latest-stable/>.
- 4 Globus Alliance. Installing GT 5.0.4. [Internet]. 2011 Available from: <http://www.globus.org/toolkit/docs/latest-stable/admin/install/>.
- 5 Globus Alliance. GT 5.0.4 Release Notes. [Internet]. 2011 Available from: <http://globus.org/toolkit/docs/5.0/5.0.4/rn/#java-api-download>.

Acknowledgments

Many people were kind enough to contribute for the results presented in this document. We are grateful for their input.

Alexander Beck-Ratzka
Ismael Marín Carrión
Steve Crouch
Harry Enke
Anton Frank
Helmut Heller
Matthias Hofmann
Andreas Hoheisel
Eduardo Huedo
Ceriël Jacobs
Raj Kettimuthu
Dagmar Krefting
Matteo Lanatti
Siew Hoon Leong
Ilya Saverchenko
Steve Tuecke
Florian Zrenner

Appendix

Job description language translation script:

```
#!/usr/bin/python
# TODO: Multijob top declaration elements should be passed on to jobs
# TODO: Include unsupported gt4rsl elements and mention in stderr that they
# are not
# supported
# TODO: Enable rsl_substitutions
# TODO: Better error checking
import xml.dom.minidom
import time
import datetime
from datetime import datetime
from datetime import timedelta
import string
import os
import sys
import subprocess
import tempfile
import csv
import re
import copy
from optparse import OptionParser

#Global variable for extensions
TRANSLATE_EXTENSIONS=1

#class job_types:
#    single="single"
#    multiple="multiple"
#    mpi="mpi"
#    condor="condor"
#    none=0

VARIABLE_PERMITTING_ELEMENTS=[
"executable",
"directory",
"argument",
"environment",
"stdin",
"stdout",
"stderr",
"libraryPath",
"fileStageIn",
"fileStageOut",
"fileCleanUp"]

UNSUPPORTED_ELEMENTS_FOUND=[]
READ_FROM_STDIN=0

MULTIJOB=0

RSL_INTERNAL_VARIABLES=["GLOBUS_USER_HOME", "GLOBUS_USER_NAME", "GLOBUS_LOCATION"]

RSL_VARIABLE_MAPPINGS={
    '\${GLOBUS_USER_HOME}': '$(HOME)',
    '\${GLOBUS_USER_NAME}': '$(LOGNAME)',
    '\${GLOBUS_LOCATION}': '$(GLOBUS_LOCATION)'
}
```

```

default_job_environment={
    'name':'',
    'value':''
}

default_extension={
    'tag_name':'',
    'value':''
}

default_file_transfer={
    'source':'',
    'destination':''
}

default_job_elements={
    'arguments':[],
    'count':'',
    'directory':'',
    'environment':[],
    'executable':'',
    'extension_list':[],
    'resource_manager_name_list':[],
    'file_clean_up':[],
    'file_stage_in':[],
    'file_stage_out':[],
    'host_count':'',
    'job_type':'',
    'library_path':'',
    'max_cpu_time':'', #in minutes,
    'max_memory':'', #in Megabytes,
    'max_time':'', # in minutes,
    'max_wall_time':'',
    'min_memory':'',
    'project':'',
    'queue':'',
    'stderr':'',
    'stdin':'',
    'stdout':''
}

def string_prepare(datastr):
    newStr=datastr
    for i in RSL_VARIABLE_MAPPINGS:
        newStr=re.sub(i,RSL_VARIABLE_MAPPINGS[i],newStr)
        if i in ['\${GLOBUS_USER_HOME}','\${GLOBUS_LOCATION}']:
            if newStr!=datastr and 'file:/' in newStr:
                newStr=re.sub('file:\/\/','',newStr)

    return string.strip(newStr)

def trans_arguments(jdl_dom,job_elements):
    for node in jdl_dom.childNodes:
        job_elements['arguments'].append(string_prepare(node.data))
    return job_elements

def trans_count(jdl_dom,job_elements):
    for node in jdl_dom.childNodes:
        job_elements['count']=string.strip(node.data)
    return job_elements

def trans_directory(jdl_dom,job_elements):
    for node in jdl_dom.childNodes:
        job_elements['directory']=string_prepare(node.data)
    return job_elements

def trans_environment(jdl_dom,job_elements):
    new_env_obj=copy.copy(default_job_environment)

```

[GLOBUS TOOLKIT 5.0 MIGRATION GUIDE]

```
env_name=jdl_dom.getElementsByTagName('name')
new_env_obj['name']=string_prepare(env_name[0].firstChild.data)
env_value=jdl_dom.getElementsByTagName('value')
new_env_obj['value']=string_prepare(env_value[0].firstChild.data)
job_elements['environment'].append(new_env_obj)
return job_elements

def trans_executable(jdl_dom,job_elements):
    for node in jdl_dom.childNodes:
        job_elements['executable']=string_prepare(node.data)
    return job_elements

def trans_extensions(jdl_dom,job_elements):
    for extension in jdl_dom.childNodes:
        if extension.nodeType == 1:
            new_ext_obj=copy.copy(default_extension)
            new_ext_obj['tag_name']=extension.nodeName
            new_ext_obj['value']=string.strip(extension.firstChild.data)
            job_elements['extension_list'].append(new_ext_obj)
    return job_elements

#Needs some extra work to figure out the exact syntax of the cleanup.
# Currently we assume that it contains 'delete' elements with one
# 'file' element each which should be deleted.
def trans_file_clean_up(jdl_dom,job_elements):
    filenames=jdl_dom.getElementsByTagName('file')
    for node in filenames:

job_elements['file_clean_up'].append(string_prepare(node.firstChild.data))
    return job_elements

#def trans_file_stage_in(jdl_dom,job_elements):
#    file_stage_in=jdl_dom.getElementsByTagName('fileStageIn')
#    for arg in file_stage_in:
#        filenames=arg.getElementsByTagName('file')
#        for node in filenames:
#
job_elements['file_stage_in'].append(string_prepare(node.firstChild.data))
#    return job_elements

def trans_file_stage_in(jdl_dom,job_elements):
    transfers=jdl_dom.getElementsByTagName('transfer')
    for transfer in transfers:
        new_fsi_obj=copy.copy(default_file_transfer)
        fsi_source=transfer.getElementsByTagName('sourceUrl')
        new_fsi_obj['source']=string_prepare(fsi_source[0].firstChild.data)

        fsi_source=transfer.getElementsByTagName('destinationUrl')

new_fsi_obj['destination']=string_prepare(fsi_source[0].firstChild.data)

        job_elements['file_stage_in'].append(new_fsi_obj)
    return job_elements

def trans_host_count(jdl_dom,job_elements):
    for node in jdl_dom.childNodes:
        job_elements['host_count']=string.strip(node.data)
    return job_elements

# Currently the job_type class is not used. Maybe it's also pointless :)
def trans_job_type(jdl_dom,job_elements):
    for node in jdl_dom.childNodes:
        job_elements['job_type']=string.strip(node.data)
    return job_elements

def trans_library_path(jdl_dom,job_elements):
    for node in jdl_dom.childNodes:
        job_elements['library_path']=string_prepare(node.data)
```

```

    return job_elements

def trans_max_cpu_time(jdl_dom, job_elements):
    for node in jdl_dom.childNodes:
        job_elements['max_cpu_time']=string.strip(node.data)
    return job_elements

def trans_max_memory(jdl_dom, job_elements):
    for node in jdl_dom.childNodes:
        job_elements['max_memory']=string.strip(node.data)
    return job_elements

def trans_max_time(jdl_dom, job_elements):
    for node in jdl_dom.childNodes:
        job_elements['max_time']=string.strip(node.data)
    return job_elements

def trans_max_wall_time(jdl_dom, job_elements):
    for node in jdl_dom.childNodes:
        job_elements['max_wall_time']=string.strip(node.data)
    return job_elements

def trans_min_memory(jdl_dom, job_elements):
    for node in jdl_dom.childNodes:
        job_elements['min_memory']=string.strip(node.data)
    return job_elements

def trans_project(jdl_dom, job_elements):
    for node in jdl_dom.childNodes:
        job_elements['project']=string.strip(node.data)
    return job_elements

def trans_queue(jdl_dom, job_elements):
    for node in jdl_dom.childNodes:
        job_elements['queue']=string.strip(node.data)
    return job_elements

def trans_stderr(jdl_dom, job_elements):
    for node in jdl_dom.childNodes:
        job_elements['stderr']=string_prepare(node.data)
    return job_elements

def trans_stdin(jdl_dom, job_elements):
    for node in jdl_dom.childNodes:
        job_elements['stdin']=string_prepare(node.data)
    return job_elements

def trans_stdout(jdl_dom, job_elements):
    for node in jdl_dom.childNodes:
        job_elements['stdout']=string_prepare(node.data)
    return job_elements

def trans_file_stage_out(jdl_dom, job_elements):
    transfers=jdl_dom.getElementsByTagName('transfer')
    for transfer in transfers:
        new_fsi_obj=copy.copy(default_file_transfer)
        fsi_source=transfer.getElementsByTagName('sourceUrl')
        new_fsi_obj['source']=string_prepare(fsi_source[0].firstChild.data)

        fsi_source=transfer.getElementsByTagName('destinationUrl')

    new_fsi_obj['destination']=string_prepare(fsi_source[0].firstChild.data)

    job_elements['file_stage_out'].append(new_fsi_obj)
    return job_elements

def processNodeChildren(job, new_job_elements):
    global UNSUPPORTED_ELEMENTS_FOUND

```

```
for node in job.childNodes:
    if node.nodeType==1:
        if string.lower(node.nodeName) == "executable":
            # extract "executable" elements from jdl
            new_job_elements=trans_executable(node,new_job_elements);
        elif string.lower(node.nodeName) == "argument":
            # extract "argument" elements from jdl
            new_job_elements=trans_arguments(node,new_job_elements);
        elif string.lower(node.nodeName) == "count":
            # extract "count" elements from jdl
            new_job_elements=trans_count(node,new_job_elements);
        elif string.lower(node.nodeName) == "directory":
            # extract "directory" elements from jdl
            new_job_elements=trans_directory(node,new_job_elements);
        elif string.lower(node.nodeName) == "environment":
            # extract "environment" elements from jdl
            new_job_elements=trans_environment(node,new_job_elements);
        elif string.lower(node.nodeName) == "extensions":
            # extract "extension" elements from jdl
            new_job_elements=trans_extensions(node,new_job_elements);
        elif string.lower(node.nodeName) == "filecleanup":
            # extract file_cleanup info
            new_job_elements=trans_file_clean_up(node,new_job_elements);
        elif string.lower(node.nodeName) == "filestagein":
            # extract stage in info
            new_job_elements=trans_file_stage_in(node,new_job_elements);
        elif string.lower(node.nodeName) == "filestageout":
            # extract stage out info

new_job_elements=trans_file_stage_out(node,new_job_elements);
    elif string.lower(node.nodeName) == "host_count":
        # extract host_count info

new_job_elements=trans_file_stage_out(node,new_job_elements);
    elif string.lower(node.nodeName) == "job_type":
        # extract job_type info
        new_job_elements=trans_job_type(node,new_job_elements);
    elif string.lower(node.nodeName) == "library_path":
        # extract library_path info
        new_job_elements=trans_library_path(node,new_job_elements);
    elif string.lower(node.nodeName) == "max_cpu_time":
        # extract max_cpu_time info
        new_job_elements=trans_max_cpu_time(node,new_job_elements);
    elif string.lower(node.nodeName) == "max_memory":
        # extract max_memory info
        new_job_elements=trans_max_memory(node,new_job_elements);
    elif string.lower(node.nodeName) == "max_time":
        # extract max_time info
        new_job_elements=trans_max_time(node,new_job_elements);
    elif string.lower(node.nodeName) == "max_wall_time":
        # extract max_wall_time info
        new_job_elements=trans_max_wall_time(node,new_job_elements);
    elif string.lower(node.nodeName) == "min_memory":
        # extract min_memory info
        new_job_elements=trans_min_memory(node,new_job_elements);
    elif string.lower(node.nodeName) == "project":
        # extract project info
        new_job_elements=trans_project(node,new_job_elements);
    elif string.lower(node.nodeName) == "queue":
        # extract queue info
        new_job_elements=trans_queue(node,new_job_elements);
    elif string.lower(node.nodeName) == "stderr":
        # extract stderr info
        new_job_elements=trans_stderr(node,new_job_elements);
    elif string.lower(node.nodeName) == "stdin":
        # extract stdin info
        new_job_elements=trans_stdin(node,new_job_elements);
    elif string.lower(node.nodeName) == "stdout":
```

```

        # extract stdout info
        new_job_elements=trans_stdout(node,new_job_elements);

        #UNSUPPORTED ELEMENTS
        else:
            # extract stdout info
            if string.lower(node.nodeName) != 'job':
                UNSUPPORTED_ELEMENTS_FOUND.append(node.nodeName)
    return new_job_elements

def parseJDL(fileIO=None,stringIO=None):
    try:
        if fileIO==None and stringIO!=None:
            dom=xml.dom.minidom.parseString(stringIO)
        elif fileIO!=None and stringIO==None:
            dom = xml.dom.minidom.parse(fileIO)
        else:
            sys.stderr.write('XML source problem, exiting...\n')
            sys.exit(1)
    except:
        sys.stderr.write('Exception trying to parse JDL input\n')
        sys.exit(1)

    job_elements_list=[]
    for node in dom.childNodes:
        if node.nodeType==1 and string.lower(node.nodeName) == "multijob":
            processNodeChildren(node,default_job_elements)
            dom=node
    job_list=dom.getElementsByTagName('job');
    for job in job_list:
        new_job_elements=copy.deepcopy(default_job_elements)
        new_job_elements=processNodeChildren(job,new_job_elements)
        job_elements_list.append(new_job_elements)
    #multijob=dom.getElementsByTagName('multiJob')
    #if multijob!=[]:
    #    # update the multijob info for all jobs without overwriting the
    #    # settings already extracted
    #    for node in multijob:
    #        for element in node.childNodes:
    #            if element.nodeType==1:
    #                #
    return job_elements_list

def create_rsl(job_elements_list, rsl_file):
    global TRANSLATE_EXTENSIONS
    global MULTIJOB
    rsl_out=sys.stdout
    if rsl_file != None:
        try:
            rsl_out=open(rsl_file,"w")
        except sys.IOError:
            sys.stderr.write("I/O error" + sys.exc_info()[0] )
            sys.exit(1)
        except:
            sys.stderr.wrote("Unexpected error:", sys.exc_info()[0])
            sys.exit(1)

    rsl_out.write('(* automatically generated from JDL file *)\n')
    indent=''
    if len(job_elements_list) > 1:
        MULTIJOB=1
        rsl_out.write('+')
        indent=' '
    for job in job_elements_list:
        if len(job_elements_list) > 1:
            rsl_out.write('\n')
        rsl_out.write(indent + '&\n')
        for element in job:

```

```
if element=='arguments' and job[element] != []:
    rsl_out.write(indent + ' ( arguments = ')
    for arg in job[element]:
        if (arg == job[element][0]):
            rsl_out.write('' + arg + ''\n')
        else:
            rsl_out.write('
            ')
            rsl_out.write('' + arg + ''\n')
    rsl_out.write(indent + ' )\n')

elif element=='file_stage_in' and job[element] != []:
    rsl_out.write(indent + ' ( file_stage_in = ')
    for transfer in job[element]:
        if (transfer == job[element][0]):
            rsl_out.write('(' + transfer['source'] + '\n')
            rsl_out.write('
            ')
            rsl_out.write(transfer['destination'] + ' )\n')
        else:
            rsl_out.write('
            ')
            rsl_out.write('(' + transfer['source']
            ')
            ')
            rsl_out.write(transfer['destination'] + ' )\n')
    rsl_out.write(' )\n')

elif element=='environment' and job[element] != []:
    rsl_out.write(indent + ' ( environment = ')
    for environment in job[element]:
        if (environment == job[element][0]):
            rsl_out.write('(' + environment['name'] + ' '
            ')
            ')
            rsl_out.write('' + environment['value'] + ' '
            ')
            ')
        else:
            rsl_out.write('
            ')
            rsl_out.write('(' + environment['name'] + ' '
            ')
            ')
            rsl_out.write('' + environment['value'] + ' '
            ')
            ')
    rsl_out.write(indent + ' )\n')

elif element=='extension_list' and job[element] != []:
    if TRANSLATE_EXTENSIONS==1:
        for extension in job[element]:
            if (extension == job[element][0]):
                rsl_out.write(indent + ' ( ' +
                extension['tag_name'] + ' = ')
                rsl_out.write(extension['value'] + ' )\n')
            else:
                rsl_out.write(indent + ' ( ' +
                extension['tag_name'] + ' = ')
                rsl_out.write(extension['value'] + ' )\n')

elif element=='file_stage_out' and job[element] != []:
    rsl_out.write(indent + ' ( file_stage_out = ')
    for transfer in job[element]:
        if (transfer == job[element][0]):
            rsl_out.write('(' + transfer['source'] + '\n')
            rsl_out.write('
            ')
            rsl_out.write(transfer['destination'] + ' )\n')
        else:
            rsl_out.write('
            ')
            rsl_out.write('(' + transfer['source']+ '\n')
            ')
            ')
            rsl_out.write(transfer['destination'] + ' )\n')
    rsl_out.write(' )\n')

elif element=='file_clean_up' and job[element] != []:
    rsl_out.write(indent + ' ( file_clean_up = ')
    for file in job[element]:
        if (file == job[element][0]):
            rsl_out.write(' ' + file + '\n')
        else:
```

```

        rsl_out.write(' ')
        rsl_out.write(' ' + file + '\n')
        rsl_out.write(' )\n')

    elif job[element] != '' and job[element] != []:
        rsl_out.write(indent + ' ( ')
        rsl_out.write(element)
        rsl_out.write(' = ')
        rsl_out.write(str(job[element]))
        rsl_out.write(' )\n')
    #rsl_out.write(indent + ' )\n')
    if len(job_elements_list) > 1:
        rsl_out.write('\n')

def parseEnvString(env_string):
    global RSL_VARIABLE_MAPPINGS
    try:
        env_list=env_string.split(',')
        for env in env_list:
            env_map=env.split(':')
            RSL_VARIABLE_MAPPINGS['${'+env_map[0]+'}']='${'+env_map[1]+'}'
    except:
        sys.stderr.write("Error parsing environment string\n")
        sys.exit(1)

# Checking commandline arguments
def optionSet():
    global TRANSLATE_EXTENSIONS
    global RSL_VARIABLE_MAPPINGS
    global READ_FROM_STDIN
    pts=0
    jdl_file=''
    parser = OptionParser()
    parser.add_option("-j", "--jdl-file",
        dest="jdl_file",
        help="File containing the GT4 job description. Use '-' for stdin",
        metavar="JDL_FILE")
    parser.add_option("-n", "--no-extensions",
        action="store_true",dest="no_extensions",
        help="Do not translate extensions.")
    parser.add_option("-f", "--rsl-file",
        dest="rsl_file",
        help="File to output the rsl job description",
        metavar="RSL_FILE")
    #parser.add_option("-e", "--environment-mappings",
    #    dest="env_mapping_string",
    #    help="String containing environment variable mappings. The string \
#consists of comma-separated pairs of names as follows: \
#RSL4_ENV_NAME:RSL5_ENV_NAME,... \nBy default, if a variable '${VAR_NAME}' \
is \
#found in the RSL4 file it will be translated as $(VAR_NAME). The following \
\
#mappings are already defined: \
GLOBUS_USER_HOME:HOME,GLOBUS_USER_NAME:LOGNAME",
    #    metavar="ENV_MAP_STRING")
    (options, args) = parser.parse_args()
    if options.jdl_file == None:
        print "Need a jdl file as input."
        sys.exit(-1)
    else:
        jdl_file=options.jdl_file
        if jdl_file == '-':
            READ_FROM_STDIN=1

    if options.no_extensions:
        TRANSLATE_EXTENSIONS=0
    #if options.env_mapping_string!=None:
    #    parseEnvString(options.env_mapping_string)

```

```
    return [jdl_file,options.rsl_file]

if __name__ == '__main__':
    optlist=optionSet()
    if READ_FROM_STDIN==1:
        try:
            job_str=sys.stdin.read()
            job_elements_list=parseJDL(fileIO=None,stringIO=job_str);
        except:
            sys.stderr.write('Cannot read input from stdin. Exiting\n')
            sys.exit(1)
    else:
        job_elements_list=parseJDL(optlist[0]);
    if(optlist[1]==None):
        create_rsl(job_elements_list,None)
    else:
        create_rsl(job_elements_list,optlist[1])
    if UNSUPPORTED_ELEMENTS_FOUND!=[]:
        if len(set(UNSUPPORTED_ELEMENTS_FOUND))>1:
            sys.stderr.write('Warning: Elements ' + ''.join(['%s' " % k for
k in \
                set(UNSUPPORTED_ELEMENTS_FOUND)]) + \
                'cannot be readily translated to GRAM5 RSL. If you need it '+\
                'please contact support@ige-project.eu\n')
        else:
            sys.stderr.write('Warning: Element ' + ''.join(['%s' " % k for
k in \
                set(UNSUPPORTED_ELEMENTS_FOUND)]) + \
                'cannot be readily translated to GRAM5 RSL. If you need it '+\
                'please contact support@ige-project.eu\n')
    if MULTIJOB==1:
        sys.stderr.write('Warning: Multijob requests are not currently ' +\
            'supported in GRAM5\n')
```