Table of Contents

Working with the Grid in the dech VO: From VO Admission to First Grid Jobs

This short documentation describes the necessary preparations and first steps in the EGEE grid and with gLite middleware. It should not replace a detailed gLite user guide.

Conditions for Working in the Grid

Three conditions must be fulfilled before you can work with the EGEE grid.

- You must possess a valid personal certificate.
- You must be a member of a Virtual Organisation (VO).
- You must have access to a User Interface (UI).

Personal Certificate

The personal certificate is necessary for authentication and authorization in the grid. It is impossible to type-in a password repeated interactively in the widely branched grid, when a request is sent from one server to another. The certificate is also necessary to access documentation and support.

Links about how to get a certificate can be found here: DECHSupportMaterial.

Virtual Organisations (VOs)

Inside EGEE users are organized in Virtual Organisitions (VOs). VOs represent a subject, project or region inside EGEE. The EGEE VOs are listed in the Core Infrastructure Center (CIC). An application form for admission to the dech VO can be found here. A valid personal certificate is necessary to load these pages.

User Interface (UI)

User Interfaces are the entrance points to the grid. They are servers containing software for the user's grid operations, e.g. job submission. To use the grid you need to login either by typing-in user name and password or with the help of your certificate. Please ask your local site for an account on the next User Interface. If your site has no UI, you can ask your VO manager. The dech VO is managed by Kläre Cassirer.

Connecting to the User Interface (UI)

Interactive Access from a Linux/Unix Computer

For security reasons only Secure Shell (SSH) should be used.

Example: A connection to the UI of the Fraunhofer Institute for Algorithms and Scientific Computing (SCAI) can be established by the command

```
ssh lcg-ui.scai.fraunhofer.de
```

If your user names are different on both sides of the connection, declare your user name on the UI. Example:

```
ssh mustermann@lcg-ui.scai.fraunhofer.de
```

Interactive Access from a Windows Computer

Again only Secure Shell (SSH) should be used. A SSH client is not part of the Windows system software. Some SSH programs are offered for free and can be downloaded, e.g. PuTTY and Tera Term. The PuTTY executable putty.exe is able to run under Windows without installation, download is sufficient.

File Transfer between Frontend and User Interface

For security reasons only Secure Copy (SCP) should be used for file transfer. SCP can transfer files from your local machine to User Interface and in the opposite direction.

⚠ Always use the most recent version of the SSH and SCP software!

For example, if you want to copy the file `fileA` from your Linux/Unix front end to the User Interface of SCAI you may type

```
scp fileA lcg-ui.scai.fraunhofer.de:
```

Example for a transfer in the opposite direction:

```
scp lcg-ui.scai.fraunhofer.de:*.out .
```

All files with extension `.out` will be copied from your home directory on the User Interface to the current directory on your front end.

You can specify a source and/or a target directory too. Example:

```
scp lcg-ui.scai.fraunhofer.de:source/*.out  target/
```

A different user name can be attached before the host name, as mentioned above.

```
scp mustermann@lcg-ui.scai.fraunhofer.de:source/*.out  target/
```

Users with Windows front end should use SCP too. A free SCP client is WinSCP.

Installation of the Personal Certificate on the User Interface

Before new users can access the grid they must have copied their personal certificate to the User Interface. We recommend to store the certificate in the directory `.globus` and in the files

`userkey.pem`	Private key (secret key)
`usercert.pem`	Public key (the intrinsic certificate)

since then it is not necessary to declare the location of the certificate in grid commands.

For EGEE the keys must be in the PEM format. In this format private and public key are in separated files. The received certificate will usually be in the PEM or PKCS12 format (extension `.p12` or `.pfx`). In the PKCS12 format private key and public key are stored together in one file.

Certificates in the PKCS12 format can be easily converted to PEM using

```
openssl pkcs12 -clcerts -nokeys -in my_cert.p12 -out usercert.pem
openssl pkcs12 -nocerts -in my_cert.p12 -out userkey.pem
```

where `my_cert.p12` is the input PKCS12 file.

After sending the latter command an input password may be inquired, which you have set before, e.g. when you have exported the `my_cert.p12` file from your web browser.

Afterwards you will be ordered to set a pass phrase for your `userkey.pem` file. OpenSSL has encrypted your private key and you will need the pass phrase every time you want to decipher it, i.e. to use it.

⚠ Set a good and long pass phrase, since your private key enables access to a large number of grid nodes!

Blanks are allowed in a pass phrase. So you can concatenate several words to a pass phrase, which can be easily remembered. Digits and punctuation marks are allowed too. In any case a private key must be encrypted and protected by a pass phrase on a User Interface.

Additionally the permissions for the `userkey.pem` file must be set to read only, owner only.

```
chmod 400 userkey.pem
```

This prevents other users from reading your secret key. Grid commands will check this and refuse to work, if the permissions are wrong.

A PKCS12 file must be also protected (or deleted), since it contains the private key too.

⚠ Certificates are issued personally to individuals, and must *never* be shared with other users. To use the grid a user must agree to an Acceptable Use Policy, which among other things requires him to keep his private key secure.

The permissions for the public key are less critical. It can be readable for everyone although there is usually no need for it.

Verify a User Certificate

Verify the Public Key

The `usercert.pem` file can be verified by the following command.

```
openssl verify -CApath /etc/grid-security/certificates ~/.globus/usercert.pem
```

If the certificate is valid and properly signed, the output will be

```
/home/mustermann/.globus/usercert.pem: OK
```

Verify the Consistency between Private and Public Key

If for some reason the user is using a public key which does not correspond to the private key, strange errors may occur. To test, if this is the case, run the command

```
grid-proxy-init -verify
```

In case of mismatch, the following error message will occur.

```
ERROR: Couldn't verify the authenticity of the user's credential to generate a proxy from.
```

Proxy Certificates

Grid commands usually cause a whole sequence of server requests and therefore user authentications. To render possible that not only the User Interface but also other servers can authenticate itself against the next server on a user's behalf the public key alone is not sufficient. A key pair with a secret key is necessary too and must have been sent to the calling server before. For security reasons not the `userkey.pem` file itself is used for this purpose but a "proxy certificate". This proxy certificate has a short lifetime, typically 12 hours.

Create Proxy

Before you can submit jobs you have to create a proxy certificate. In the dech VO the proxy must contain an attribute that verifies your VO membership. Such a proxy can be created with the command

```
voms-proxy-init -voms dech
```

VOMS is the abbreviation for Virtual Organisation Membership Service.

⚠ The permissions of the private key `userkey.pem` must be set to read only, owner only.

The error message

```
Cannot find file or dir: /home/mustermann/.glite/vomses
```

can be ignored. With

```
Enter GRID pass phrase:
```

you are requested to type-in the pass phrase of your `userkey.pem` file.

Sometimes proxy creation is failing with the following error message.

```
Error: Could not establish authenticated connection with the server.
GSS Major Status: Unexpected Gatekeeper or Service Name
```

In this case simply try it again. The server "hangs" sometimes.

Destroy Proxy

To destroy an existing proxy certificate before its expiration, it is enough to do

```
voms-proxy-destroy
```

Output Information about Proxy

Information about a proxy certificate can be obtained with the following command.

```
voms-proxy-info
```

Job Description Language

On the grid mainly batch jobs are executed. A batch job is a non-interactive job controlled by a file (job description file) prepared by the user. The job description file contains the necessary settings to run the job. In the EGEE grid the Job Description Language (JDL) must be used. One of the easiest possible jobs written in JDL is the following.

```
Executable = "/bin/hostname";
```

Just the host name of the machine executing the job is inquired.

⚠ No blanks are allowed after the semicolon at the end of the line.

This job has a severe disadvantage. It does not produce any output, since JDL expects additional attributes where the output has to be written to. We need at least the following JDL attributes to produce output.

```
Executable = "/bin/hostname";
StdOutput = "std.out";
StdError = "std.err";
OutputSandbox = {"std.out", "std.err"};
```

The standard output will be written to `std.out` now, the standard error output to `std.err`. The `OutputSandbox` attribute must contain all files that shall be sent back.

More than one `Executable` attributes are allowed but only the last will be executed. The use of a shell script allowes us to execute more than one command.

```
Executable = "job.sh";
StdOutput = "std.out";
StdError = "std.err";
InputSandbox = {"job.sh"};
OutputSandbox = {"std.out", "std.err"};
```

The `InputSandbox` attribute names all files that need to be copied from the User Interface to the Compute Element, separated by commas. The files are expected to be in the user's home directory unless a relative path is given. If an absolute path, e.g. `/bin/hostname`, is given, the file is expected to be already on the Compute Element.

Example for a shell script with two commands:

```
#!/bin/sh
echo "Hostname:"
hostname -f   # provide full hostname
```

Constraints on the resources can be set by the `Requirements` attribute in the job description file. For example use

```
Requirements = other.GlueCEInfoTotalCPUs > 1 &&
               other.GlueCEPolicyMaxWallClockTime > 720;
```

to express the fact that the job needs at least 2 CPUs and up to 12 wall clock hours. Only one `Requirements` attribute is allowed. If you have more than one constraint like in the example, join them with `&&`, the logical AND.

To have a list of supported attributes use

```
lcg-info --list-attrs
```

Commands for Job Handling

Submit Grid Jobs

For submitting of grid jobs you need a VOMS proxy certificate which has not expired. On the User Interface the command for submitting a job is

```
glite-wms-job-submit -a job.jdl
```

where the job description file `job.jdl` is containing the necessary JDL attributes and is located in your home directory on the User Interface.

If the submission is successful a confirmation and the job ID is sent back. With the `-o` option the quite long job ID can be written to a file (in the example `id`).

```
glite-wms-job-submit -o id -a job.jdl
```

With the `-r` option the job can be sent to the given Compute Element for execution, e.g.

```
glite-wms-job-submit -a -r cedric.scai.fraunhofer.de:2119/jobmanager-lcgpbs-egdech job.jdl
```

In the case the proxy certificate is missing or expired you will get the following error message.

```
Error - Proxy File Not Found
Unable to find a valid proxy file
```

Inquire Job Status

Job status information can be obtained with

```
glite-wms-job-status <jobID>
```

or, if the job ID is stored in a file, e.g. in `id`, with

```
glite-wms-job-status -i id
```

In the gLite context a job can have the following status.

Status	Definition
SUBMITTED	The job has been submitted by the user but not yet processed by the Network Server or Workload Management Proxy
WAITING	The job has been accepted by the Network Server or Workload Management Proxy but not yet processed by the Workload Management System
READY	The job has been assigned to a Computing Element but not yet transferred to it
SCHEDULED	The job is waiting in the Computing Element's queue
RUNNING	The job is running
DONE	The job has finished
ABORTED	The job has been aborted by the Workload Management System (e.g. because it was too long, or the proxy certificated expired, etc.)
CANCELED	The job has been canceled by the user
CLEARED	The Output Sandbox has been transferred to the User Interface

The purpose of the Workload Management System (WMS) is to accept user jobs, to assign them to the most appropriate Computing Element, to record their status and to retrieve their output. The machine where the WMS services run is called "Resource Broker".

Cancel Jobs

A job can be canceled before it ends with

```
glite-wms-job-cancel <jobID>
```

or, if the job ID is stored in a file (here `id`)

```
glite-wms-job-cancel -i id
```

Retrieve Job Output

After the job has been finished successfully you may transfer the Output Sandbox to the User Interface. This can be done with the help of the command

```
glite-wms-job-output <jobID>
```

or, if the job ID is stored in a file (here `id`), with

```
glite-wms-job-output -i id
```

Output a list of Grid Resources

dech VO members have access to more than a dozen Compute Elements where jobs can be executed. A list together with information about

- the number of total CPUs
- the number of free CPUs
- the number of running and waiting jobs

can be obtained with the following command.

```
lcg-infosites --vo dech ce
```

A list together with information about memory, operating system and processor type can be got by setting an additional verbosity level.

=lcg-infosites --vo dech ce -v 2=

This topic: EGEE > DECHFirstJobs
Topic revision: r15 - 2009-08-19 - TorstenRathmann