



Enabling Grids for E-scienceE

# GILDA Practicals

**Giuseppe La Rocca**  
**INFN – Catania**

**BIOINFOGRID Initial training course**  
**BARI, 08-10 March 2006**

[www.eu-egee.org](http://www.eu-egee.org)



- In the glite middleware a user can submit and cancel jobs, query their status, and retrieve their output. These tasks go under the name of ***Workload Management***.
- There are two different User Interfaces to accomplish these tasks. One is the Command Line Interface and the other is the Graphical User Interface.

- **Job Submission**

- Perform the job submission to the Grid.

```
$ glite-job-submit [options] <jdl_file>
```

- where <jdl file> is a file containing the job description, usually with extension .jdl.

**--vo** <vo name> : perform submission with a different VO than the UI default one.

**--output, -o** <output file> save jobId on a file.

**--resource, -r** <resource value> specify the resource for execution.

**--nomsgi** neither message nor errors on the stdout will be displayed.

If the request has been correctly submitted this is the typical output that you can get:

**glite-job-submit test.jdl**

```
=====glite-job-submit Success =====  
The job has been successfully submitted to the Network Server.  
Use glite-job-status command to check job current status.  
Your job identifier (edg_jobId) is:  
- https://lxshare0234.cern.ch:9000/rIBubkFFKhnsQ6CjiLUY8Q  
=====
```

In case of failure, an error message will be displayed instead, and an exit status different from zero will be returned.

If the command returns the following error message:

```
**** Error: API_NATIVE_ERROR ****
```

```
Error while calling the "NSClient::multi" native api
```

```
AuthenticationException: Failed to establish security context...
```

```
**** Error: UI_NO_NS_CONTACT ****
```

```
Unable to contact any Network Server
```

it means that there are authentication problems between the UI and the *Network Server* (check your proxy or contact the site administrator).

It is possible to see which CEs are eligible to run a job specified by a given JDL file using the command

**glite-job-list-match test.jdl**

**Connecting to host lxshare0380.cern.ch, port 7772**

**Selected Virtual Organisation name (from UI conf file): dteam**

\*\*\*\*\*

### **COMPUTING ELEMENT IDs LIST**

**The following CE(s) matching your job requirements have been found:**

**adc0015.cern.ch:2119/jobmanager-lcgpbs-infinite**

**adc0015.cern.ch:2119/jobmanager-lcgpbs-long**

**adc0015.cern.ch:2119/jobmanager-lcgpbs-short**

\*\*\*\*\*

After a job is submitted, it is possible to see its status using the `glite-job-status` command.

`glite-job-status` <https://lxshare0234.cern.ch:9000/X-ehTxfdlXxSoIdVLS0L0w>

\*\*\*\*\*

**BOOKKEEPING INFORMATION:**

**Printing status info for the Job:**

**`https://lxshare0234.cern.ch:9000/X-ehTxfdlXxSoIdVLS0L0w`**

**Current Status: Scheduled**

**Status Reason: unavailable**

**Destination: `lxshare0277.cern.ch:2119/jobmanager-pbs-infinite`**

**reached on: Fri Aug 1 12:21:35 2003**

\*\*\*\*\*

The option **-i <file path>** can be used to specify a file with a list of job identifiers (saved previously with the **-o** option of `glite-job-submit`).

**glite-job-status -i jobs.list**

```
-----
1 : https://lxshare0234.cern.ch:9000/UPBqN2s2ycxt1TnuU3kzEw
2 : https://lxshare0234.cern.ch:9000/8S6IwPW33AhyxhkSv8Nt9A
3 : https://lxshare0234.cern.ch:9000/E9R0Yl4J7qgsq7FYTnhmsA
4 : https://lxshare0234.cern.ch:9000/Tt80pBn17AFPJyUSN9Qb7Q
a : all
q : quit
-----
```

Choose one or more `edg_jobId(s)` in the list - [1-4]all:

If the **--all** option is used instead, the status of all the jobs owned by the user submitting the command is retrieved.

The **--status <state>** (-s) option makes the command retrieve only the jobs that are in the specified state, and the **--exclude**

**<state>** (-e) option makes it retrieve jobs that are not in the specified state.

This two lasts options are mutually exclusive, although they can be used with **--from** and **--to**.

Example: All jobs of the user that are in the state **DONE** or **RUNNING** are retrieved.

```
glite-job-status --all -s Done -s Running
```

A job can be canceled before it ends using the command `glite-job-cancel`.

`glite-job-cancel` <https://lxshare0234.cern.ch:9000/dAE162is6EStca0VqhVkog>

Are you sure you want to remove specified job(s)? [y/n]n :y

===== `glite-job-cancel` Success=====

The cancellation request has been successfully submitted for the following job(s)

- <https://lxshare0234.cern.ch:9000/dAE162is6EStca0VqhVkog>

=====

After the job has finished (it reaches the DONE status), its output can be copied to the UI

**glite-job-output** <https://lxshare0234.cern.ch:9000/snPegp1YMJcnS22yF5pFlg>

**Retrieving files from host lxshare0234.cern.ch**

\*\*\*\*\*

**JOB GET OUTPUT OUTCOME**

**Output sandbox files for the job:**

- <https://lxshare0234.cern.ch:9000/snPegp1YMJcnS22yF5pFlg>

**have been successfully retrieved and stored in the directory:**

**/tmp/jobOutput/snPegp1YMJcnS22yF5pFlg**

\*\*\*\*\*

By default, the output is stored under /tmp, but it is possible to specify in which directory to save the output using the **- -dir <path name>** option.

# Exercise 1



Create or modify `ls.jdl` and `ls.sh` as follow:

```
[
  Executable = "ls.sh";
  Arguments = "-al";
  StdError = "stderr.log";
  StdOutput = "stdout.log";
  InputSandbox = "ls.sh";
  OutputSandbox = {"stderr.log", "stdout.log"};
]
```

**ls.sh**

```
#!/bin/sh
```

```
/bin/ls $1
```

Make `ls.sh` script executable with `chmod +x ls.sh`

## Exercise 2



[

**JobType = "MPICH";**

**Executable = "MPItest.sh";**

**NodeNumber = 2;**

**Arguments = "cpi 2";**

**StdOutput = "test.out"; StdError = "test.err";**

**InputSandbox = {"MPItest.sh", "cpi"};**

**OutputSandbox =**

**{"test.err", "test.out", "executable.out"};**

**Requirements = other.GlueCEInfoLRMSType == "PBS" ||**

**other.GlueCEInfoLRMSType == "LSF";**

]

The number of threads specified with NodeNumber attribute agrees with the second Argument. It will be used during the invoking of mpirun command.

```

for i in `cat $HOST_NODEFILE` ; do
  echo "Mirroring via SSH to $i"
  # creates the working directories on all the nodes allocated for parallel
  # execution.
  ssh $i mkdir -p `pwd`
  # copies the needed files on all the nodes allocated for parallel
  # execution.
  /usr/bin/scp -rp ./* $i:`pwd`
  # checks that all files are present on all the nodes allocated for parallel
  # execution.
  ssh $i ls `pwd`
done

# execute the parallel job with mpirun.
echo "Executing $EXE"
chmod 755 $EXE
mpirun -np $CPU_NEEDED -machinefile $HOST_NODEFILE
  `pwd`/$EXE > executable.out
  
```

The Environment variable **\$HOST\_NODEFILE** contains the list of WNs allocated for the parallel execution.

# Exercise 3



This exercise allows user to submit a C program.

Modify **c\_sample.c** file as follow:

```
#include <stdio.h>  
int main(int argc, char **argv)  
{  
    printf("\n\n\n");  
    printf("Hello !\n");  
    printf("Welcome to the BIOINFOGRID Initial training  
course – Bari 08-10 March - 2006 \n\n\n");  
    exit(0);  
}
```

**Compile your script with:** `gcc -o c_sample c_sample.c`

**Submit the `c_sample.jdl` job to the grid**

[

**Executable = "/bin/sh";**

**Arguments = "start\_c\_sample.sh";**

**StdOutput = "std.out";**

**StdError = "std.err";**

**InputSandbox =**

**{"c\_sample", "start\_c\_sample.sh"};**

**OutputSandbox = {"std.err", "std.out"};**

]

Create the **start\_c\_sample.sh** script as follow:

```
#!/bin/sh
```

```
chmod 777 c_sample
```

```
./c_sample
```

Inspect the status and retrieve its output when the job is finished.

# Exercise 4



Modify **c\_sample.c** file as follow:

```
#include <stdio.h>  
int main(int argc, char **argv)  
{  
    char *name = argv[1];  
    printf("\n\n\n");  
    printf("Hello %s!\n",name);  
    printf("Welcome to ICPT/INFM Tutorial, Trieste 07th-  
        17th Feb. - 2006 \n\n\n");  
    exit(0);  
}
```

Compile your script with: `gcc -o c_sample c_sample.c`

Modify the `start_c_sample.sh` script as follow:

```
#!/bin/sh  
chmod 777 c_sample  
./c_sample $1
```

Modify `c_sample.jdl`'s Arguments as follow:

```
Arguments = "start_c_sample.sh <Your Name>";
```

Submit, inspect the status and retrieve its output when the job is finished.

# Exercise 5

**View user Credits**



## **\$ dgas-check-balance**

**User:** Giuseppe La Rocca

**E-mail:** giuseppe.larocca@ct.infn.it

**Subject:** /C=IT/O=GILDA/OU=Personal Certificate/L=INFN  
Catania/CN=Giuseppe La  
Rocca/Email=giuseppe.larocca@ct.infn.it

**Assigned credits (0=infinite): 0**

**Booked credits: 0**

**Used credits: 451**

**Used wall clock time (sec): 1187**

**Used CPU time (sec): 264**

**Accounted jobs: 22**

# Exercise 6

**View CE Price**



**Usage: dgas-check-ce-price <CE name>:2119/jobmanager-lcgpbs-  
<queue>**

**Example: dgas-check-ce-price  
grid010.ct.infn.it:2119/jobmanager-lcgpbs-short**

**Price Authority queried at: Thu Oct 20 18:43:39 CEST 2005**

**Computing Element: grid010.ct.infn.it:2119/jobmanager-lcgpbs-  
short**

**Price (credits for 100 CPU secs): 170**

