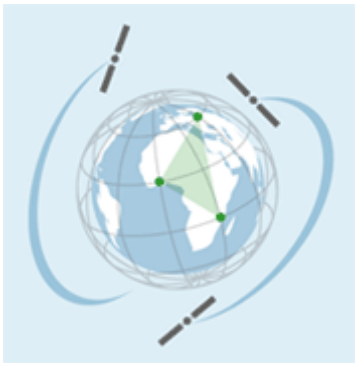

Workshop

"GRID & e-Collaboration for the Space Community"

EGEE Tutorial hands-on session

Prepared by
Sabrina Argentati, Julian Linford, Valeria Ardizzone

ESA/ESRIN
Frascati (RM), Italy

3 February 2005

CONTENTS

1 INTRODUCTION

1.1 REFERENCES

[R1] LCG-2 USER GUIDE
https://edms.cern.ch/file/454439/1/LCG-2-UserGuide.pdf

[R2] Training "Introduction to Grid Computing"
http://infnforge.cnaf.infn.it/cdsagenda/fullAgenda.php?ida=a0440

[R3] Enabling Grids for E-science in Europe
http://eu-egee.org

[R4] INFN Production Grid for Scientific Applications
http://grid-it.cnaf.infn.it/

1.2 TERMINOLOGY

API: Application Programming Interface
BDII: Berkeley Database Information Index
CATOR CERN Advanced STORage manager
CE: Computing Element
CERN: European Laboratory for Particle Physics
ClassAd: Classified advertisement
CLI: Command Line Interface
CNAF: INFN's National Center for Telematics and Informatics
DIT: Directory Information Tree
DN: Distinguished Name (LDAP's)
EDG: European DataGrid
EDT: European DataTag
EGEE: Enabling Grids for E-science in Europe
ESM: Experiment Software Manager
FNAL: Fermi National Accelerator Laboratory
GIIS: Grid Index Information Server
GLUE: Grid Laboratory for a Uniform Environment
GOC: Grid Operations Centre
GRAM: Globus Resource Allocation Manager
GRIS: Grid Resource Information Service
GSI: Grid Security Infrastructure
GUI: Graphical User Interface
GUID: Grid Unique ID
ID: Identifier
INFN: Instituto Nazionale di Fisica Nucleare
IS: Information Service
JCS: Job Control Service
JDL: Job Description Language
LB: Logging and Bookkeeping Service
LDAP: Lightweight Directory Access Protocol
LFN: Local File Name
LHC: Large Hadron Collider
LGC: LHC Computing Grid
LRC: Local Replica Catalog

LRMS: Local Resource Management System
LSF: Load Sharing Facility
MDS: Monitoring and Discovery Service
MPI: Message Passing Interface
MSS: Mass Storage System
NS: Network Server
OS: Operating System
PBS: Portable Batch System
PFN: Physical File name
PID: Process IDentifier
POOL: Pool of Persistent Objects for LHC
RAL: Rutherford Appleton Laboratory
RB: Resource Broker
RFIO: Remote File Input/Output
RLI: Replica Location Index
RLS: Replica Location Service
RM: Replica Manager
RMC: Replica Metadata Catalog
RMS: Replica Management System
ROS: Replica Optimization Service
SASL: Simple Authorization & Security Layer (LDAP)
SE: Storage Element
SMP: Symmetric Multi Processor
SRM: Storage Resource Manager
SURL: Storage URL
TURL: Transport URL
UI: User Interface
URI: Uniform Resource Identifier
URL: Universal Resource Locator
UUID: Universal Unique ID
VDT: Virtual Data Toolkit
VO: Virtual Organisation
WMS: Workload Management System
WN: Worker Node
WPn: Work Package #n

1.3 THE USER INTERFACE (UI)

The UI is the users point of access to the Grid.

It is a machine where the users have a personal account and the user's certificate is installed.

From the UI , a user can be authenticated and authorized to use the Grid resources to :

- submit a job for execution
- list all the resources suitable to execute a given job
- replicate and copy files
- cancel a job
- retrieve the output of a finished job
- show the status of a submitted job

During the tutorial we will use the Gilda User Interface.

Gilda is a real virtual laboratory for dissemination of grid computing.

2 GETTING STARTED

2.1 Log into the Gilda user Interface (UI)

Use the secure shell client to ssh to one of the two Gilda testbed UI machines :

grid-tutor.ct.infn.it

or

grid-tutor1.ct.infn.it

Username : **romexx**
Password : **GridROMxx**

Un-tar the tutorial files to your local directory

```
$ tar -xvf tutorial-esrin.tar
```

Create a new directory where your job results will be stored

```
$ mkdir JobOutput
```

Check your user credentials are stored in your **.globus** directory :

usercert.pem : your public key
userkey.pem : your private key

```
$ cd .globus
```

```
$ ls  -la
drwxr-xr-x    2 rome01   users         112 Jan 24 09:41 .
drwx------   12 rome01   users        6472 Jan 24 18:35 ..
-rw-r--r--    1 rome01   users        1127 Jan 24 09:41 usercert.pem
-r--------    1 rome01   users         963 Jan 24 09:41 userkey.pem
```

```
$ cd ..
```

Now you are logged in to the UI, but you are not yet logged in to the Grid

2.2 Log in to the Grid – create Proxy Certificate

To log in to the Grid you must create a proxy certificate. The UI will attach it to any job you submit.

The private and public keys are not sent , this proxy certificate is a temporary certificate valid for 12 hours (this is the default time if you didn't give a specific time).

Create your proxy certificate, you will be asked for a "pass phrase" : it is ROME

```
$ grid-proxy-init
```

```
Your identity: /C=IT/O=GILDA/OU=Personal
                    Certificate/L=ROME/CN=ROME01/Email=roberto.puccinelli@cedrc.cnr.it
Enter GRID pass phrase for this identity:

Creating proxy ................................ Done
Your proxy is valid until: Tue Jan 25 07:41:10 2005
```

Display information about the proxy certificate :

```
$ grid-proxy-info

subject  : /C=IT/O=GILDA/OU=Personal
                    Certificate/L=ROME/CN=ROME01/Email=roberto.puccinelli@cedrc.cnr.it/C
                    N=proxy
issuer   : /C=IT/O=GILDA/OU=Personal
                    Certificate/L=ROME/CN=ROME01/Email=roberto.puccinelli@cedrc.cnr.it
identity : /C=IT/O=GILDA/OU=Personal
                    Certificate/L=ROME/CN=ROME01/Email=roberto.puccinelli@cedrc.cnr.it
type     : full legacy globus proxy
strength : 512 bits
path     : /tmp/x509up_u1625
timeleft : 11:33:24
```

To destroy an existing proxy certificate before its expiration :

```
$ grid-proxy-destroy
```

Now check the procy info :

```
$ grid-proxy-info

ERROR: Couldn't find a valid proxy.
Use -debug for further information.
```

2.3 Register a long living proxy in the MyProxy server grid001.ct.infn.it

The proxy certificates created in the last exercise have a limited lifetime. However, if the job does not finish before the proxy expires, it will be aborted.

This is clearly a problem if, for example, the user must submit a number of jobs that take a lot of time to finish: he should create a proxy certificate with a very long lifetime, fact that would increase the security risks.

To overcome this limit, a proxy credential repository system is used, which allows the user to create and store a long-term proxy certificate on a dedicated server (Proxy Server).

To register a long living proxy in the MyProxy server **`grid001.ct.infn.it`** you use the **`myproxy-init`** command

- You will be asked for the password to access your private key certificate
- You are asked to enter a new password to be associated with your MyProxy delegation
- It must be at least 6 characters long

IT IS RECOMMENDED THAT THIS SHOULD BE DIFFERENT to that in your certificate.
IT IS ALSO RECOMMENDED THAT YOU REMEMBER IT.

```
$ myproxy-init -s grid001.ct.infn.it

Your identity: /C=IT/O=GILDA/OU=Personal
                    Certificate/L=ROME/CN=ROME01/Email=roberto.puccinelli@cedrc.cnr.it
Enter GRID pass phrase for this identity:
Creating proxy ..................................... Done
Proxy Verify OK
Your proxy is valid until: Thu Feb  3 13:11:40 2005
Enter MyProxy pass phrase:
Verifying password - Enter MyProxy pass phrase:
A proxy valid for 168 hours (7.0 days) for user rome01 now exists on grid001.ct.infn.it.
```

Now check the proxy created in the myproxy-init command

```
$ myproxy-info -s grid001.ct.infn.it

username: rome01
owner: /C=IT/O=GILDA/OU=Personal
                    Certificate/L=ROME/CN=ROME01/Email=roberto.puccinelli@cedrc.cnr.it
  timeleft: 167:21:47  (7.0 days)
```

It has a lifetime of a 1 week by default

Now, verify local proxy does not exist anymore

```
$ grid-proxy-info

>ERROR: Couldn't find a valid proxy.
Use -debug for further information.
```

2.4 Retrieve a proxy certificate from the MyProxy server

It has a default lifetime of 12 hours.THIS REQUIRES THE PASSWORD SET IN STEP 2)

```
$ myproxy-get-delegation -s grid001.ct.infn.it

Enter MyProxy pass phrase:
A proxy has been received for user rome01 in /tmp/x509up_u1625
```

Now check the proxy you just retrieved:

```
$ grid-proxy-info

 subject  : /C=IT/O=GILDA/OU=Personal
                    Certificate/L=ROME/CN=ROME01/Email=roberto.puccinelli@cedrc.cnr.it/C
                    N=proxy/CN=proxy/CN=proxy
issuer   : /C=IT/O=GILDA/OU=Personal
                    Certificate/L=ROME/CN=ROME01/Email=roberto.puccinelli@cedrc.cnr.it/C
                    N=proxy/CN=proxy
identity : /C=IT/O=GILDA/OU=Personal
                    Certificate/L=ROME/CN=ROME01/Email=roberto.puccinelli@cedrc.cnr.it
type     : full legacy globus proxy
strength : 512 bits
path     : /tmp/x509up_u1625
timeleft : 11:59:20
```

2.5 Destroy the remote proxy

```
$ myproxy-destroy -s grid001.ct.infn.it

Default MyProxy credential for user rome01 was successfully removed.
```

However, note that your local proxy will still be valid until it expires or you destory it:

```
$ grid-proxy-info

subject  : /C=IT/O=GILDA/OU=Personal
                    Certificate/L=ROME/CN=ROME01/Email=roberto.puccinelli@cedrc.cnr.it/C
                    N=proxy/CN=proxy/CN=proxy
issuer   : /C=IT/O=GILDA/OU=Personal
                    Certificate/L=ROME/CN=ROME01/Email=roberto.puccinelli@cedrc.cnr.it/C
                    N=proxy/CN=proxy
identity : /C=IT/O=GILDA/OU=Personal
                    Certificate/L=ROME/CN=ROME01/Email=roberto.puccinelli@cedrc.cnr.it
type     : full legacy globus proxy
strength : 512 bits
path     : /tmp/x509up_u1625
timeleft : 11:34:30
```

3 WORKLOAD MANAGEMENT

In the Grid, a user can submit and cancel jobs, query their status, and retrieve their output.
These tasks go under the name of ***Workload Management***.

edg-job-submit <job.jdl>	Submit a job.
edg-job-cancel <jobID>	Cancel a job.
edg-job-status <jobID>	Gives the status. The steps are: WAIT READY SCHEDULED RUNNING DONE
edg-job-get-output <jobID>	Gets your output from the Resource Broker, which holds the output sandbox (so there is no direct contact with the machine on which the job executed)
edg-job-list-match <job.jdl>	This shows the Computing Elements (CE) that are available for your job, specified by the .jdl file.

3.1 JOB DESCRIPTION LANGUAGE (JDL)

JDL files are used to describe jobs for execution on Grid. Some JDL attributes are mandatory, while other are optional. The mandatory attributes are :

The name of the executable (on the WN)

e.g. **`Executable = "/bin/hostname";`**

The name of the files where to write the standard output (on the WN)

e.g. **`StdOutput = "std.out";`**

The standard error of the job (on the WN)

e.g. **`StdError = "std.err";`**

You can optionally specify the files to be transferred between the UI and the WN before execution

e.g. **`InputSandbox = {"test.sh","std.in"};`**

and after excution

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

Now lets see a some JDL examples :

```
$ cd JobSubmission
```

The first example executes a shell command directly :

```
$ more hostname.jdl

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

While the next example sends and executes a script from your UI

```
$ more scriptls.jdl

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

Note that the script to be sent is specified using the **`InputSandbox`** attribute

```
$ more ls.sh

#!/bin/sh
/bin/ls -la
```

3.2 Submitting a simple Job

Use the **`edg-job-submit`** command to submit a job to the Grid

e.g. **`$ edg-job-submit <jdl_file>`**

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

```
$ edg-job-submit scriptls.jdl

Selected Virtual Organisation name (from UI conf file): gilda
Connecting to host grid004.ct.infn.it, port 7772
Logging to host grid004.ct.infn.it, port 9002


**********************************************************************************
                              JOB SUBMIT OUTCOME
 The job has been successfully submitted to the Network Server.
 Use edg-job-status command to check job current status. Your job identifier (edg_jobId)
                     is:

 - https://grid004.ct.infn.it:9000/eohzMeoLPQMlwqtOO4heoQ


**********************************************************************************
```

Notice that the command outputs a unique job identifier, which you use in later commands to refer to your job.

3.3 Retrieving the status of a job

Use the **edg-job-status** command to obtain the status of a submitted job :

```
edg-job-status <jobId>
```

Where **<jobId>** is the unique job id that you copy/paste from the result of the **edg-job-submit**command:

```
$ edg-job-status https://grid004.ct.infn.it:9000/eohzMeoLPQMlwqtOO4heoQ

**********************************************************
BOOKKEEPING INFORMATION:

Status info for the Job : https://grid004.ct.infn.it:9000/eohzMeoLPQMlwqtOO4heoQ
Current Status:     Done (Success)
Exit code:          0
Status Reason:      Job terminated successfully
Destination:        grid002.mporzio.astro.it:2119/jobmanager-lcgpbs-infinite
reached on:         Tue Jan 25 17:59:50 2005
**********************************************************
```

3.4 Retrieving the output of a job

When the job reaches the DONE status its output can be copied to the UI using the **edg-job-get-output** command :

```
edg-job-get-output <jobId>
```

Try it:

```
$ edg-job-get-output https://grid004.ct.infn.it:9000/eohzMeoLPQMlwqtOO4heoQ

 Retrieving files from host: grid004.ct.infn.it ( for
                     https://grid004.ct.infn.it:9000/eohzMeoLPQMlwqtOO4heoQ )

************************************************************************
                        JOB GET OUTPUT OUTCOME

 Output sandbox files for the job:
 - https://grid004.ct.infn.it:9000/eohzMeoLPQMlwqtOO4heoQ
 have been successfully retrieved and stored in the directory:
 /home/rome01/JobOutput/rome01_eohzMeoLPQMlwqtOO4heoQ

************************************************************************
```

The files specified in the **OutputSandbox** JDL attribute are transfered from the Grid to the UI machine :

```
$ cd  /home/rome01/JobOutput/rome01_eohzMeoLPQMlwqtOO4heoQ

$ ls -la

$ more std.out
```

3.5 Advanced Command Options

The `-o <file path>` option of `edg-job-submit` can be used to store the <JobId> on a file, which can later be used with `edg-job-status`

```
$ edg-job-submit  -o agenda scriptls.jdl
```

```
Connecting to host grid004.ct.infn.it, port 7772
Logging to host grid004.ct.infn.it, port 9002

****************************** edg-job-submit Success ******************************
 The job has been successfully submitted to the Network Server.
 Use edg-job-status command to check job current status. Your job identifier (edg_jobId)
                    is:

 - https://grid004.ct.infn.it:9000/lUtlS0ko2NTQETy4xmME4g

 The edg_jobId has been saved in the following file:
 /home/rome01/JobSubmission/agenda
**********************************************************************************
```

```
$ $ more agenda
###Submitted Job Ids###
https://grid004.ct.infn.it:9000/lUtlS0ko2NTQETy4xmME4g
```

Now you can submit a new job:

```
$ edg-job-submit  -o agenda hostname.jdl

Selected Virtual Organisation name (from UI conf file): gilda
Connecting to host grid004.ct.infn.it, port 7772
Logging to host grid004.ct.infn.it, port 9002

****************************** edg-job-submit Success
                    ************************************
 The job has been successfully submitted to the Network Server.
 Use edg-job-status command to check job current status. Your job identifier (edg_jobId)
                    is:

 - https://grid004.ct.infn.it:9000/qku-d0zDPVkvReFptqx7kg

 The edg_jobId has been saved in the following file:
 /home/rome01/JobSubmission/agenda
**********************************************************************************
```

```
$ more agenda
###Submitted Job Ids###
https://grid004.ct.infn.it:9000/lUtlS0ko2NTQETy4xmME4g
https://grid004.ct.infn.it:9000/qku-d0zDPVkvReFptqx7kg
```

3.6 MPI jobs

Message Passing Interface (MPI) applications are run in parallel on several processors. Jobs that must be run using MPI are specified by setting the **JobType** attribute to **MPICH** in the JDL file.

The **NodeNumber** attribute is mandatory for MPI jobs, it specifies the required number of CPUs.

The MPICH runtime environment has to be present on the CE and the number of available CPUs must be at least equal to the required number of nodes.

This can be specified in the JDL file by adding the following expression :

```
(other.GlueCEInfoTotalCPUs >= NodeNumber) &&
Member(other.GlueHostApplicationSoftwareRunTimeEnvironment,"MPICH")
```

A typical JDL needed to run a MPI job over the GRID is shown below (see mpi1 dir) :

```
Type = "Job";
JobType = "MPICH";
NodeNumber = 4;
Executable = "MPItest.sh";
Arguments = "cpi 4";
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 "NodeNumber" entry is the number of threads of MPI job; it is the second argument to let the script use it. MPItest.sh is the following script:

```
#!/bin/sh
#
# this parameter is the binary to be executed
EXE=$1
# this parameter is the number of CPU's to be reserved for parallel execution
CPU_NEEDED=$2
# prints the name of the master node
echo "Running on: $HOSTNAME"
echo "************************************"
if [ -f "$PWD/.BrokerInfo" ] ; then
TEST_LSF=`edg-brokerinfo getCE | cut -d/ -f2 | grep lsf`
else
TEST_LSF=`ps -ef | grep sbatchd | grep -v grep`
fi
if [ "x$TEST_LSF" = "x" ] ; then
# prints the name of the file containing the nodes allocated for parallel
execution
echo "PBS Nodefile: $PBS_NODEFILE"
# print the names of the nodes allocated for parallel execution
cat $PBS_NODEFILE
echo "************************************"
HOST_NODEFILE=$PBS_NODEFILE
else
# print the names of the nodes allocated for parallel execution
echo "LSF Hosts: $LSB_HOSTS"
# loops over the nodes allocated for parallel execution
```

```
HOST_NODEFILE=`pwd`/lsf_nodefile.$$
for host in ${LSB_HOSTS}
do
echo $host >> ${HOST_NODEFILE}
done
fi
echo "************************************"
# prints the working directory on the master node
echo "Current dir: $PWD"
echo "************************************"
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
echo `pwd`
ssh $i ls `pwd`
# sets the permissions of the files
ssh $i chmod 755 `pwd`/$EXE
ssh $i ls -alR `pwd`
echo "@@@@@@@@@@@@@@@@"
done
# execute the parallel job with mpirun
echo "**********************************"
echo "Executing $EXE"
chmod 755 $EXE
ls -l
mpirun -np $CPU_NEEDED -machinefile $HOST_NODEFILE `pwd`/$EXE > executable.out
echo "**********************************"
```

The MPI executable cpi is available in mpi directory

The environment variable **$HOST_NODEFILE** is the path of a file that contains the list of WNs allocated for the parallel execution. The script above works only if PBS or LSF is the local job manager. This is the **stdout** of the job after its execution (file **test.out** in the example above):

```
Running on: grid022.ct.infn.it
************************************
PBS Nodefile: /var/spool/pbs/aux/1434.grid010.ct.infn.it
grid022.ct.infn.it
grid021.ct.infn.it
grid020.ct.infn.it
grid026.ct.infn.it
************************************
Current                          dir:                          /home/gilda002/globus-
tmp.grid022.16511.0/.mpi/https_3a_2f_2fgrid004.ct.infn.it_3a9000_2f4P2yVyypejJXh
c1OuTZlOA
************************************
Mirroring via SSH to grid022.ct.infn.it
/home/gilda002/globus-
tmp.grid022.16511.0/.mpi/https_3a_2f_2fgrid004.ct.infn.it_3a9000_2f4P2yVyypejJXh
c1OuTZlOA
PI17046
cpi
new_MPItest.sh
test.err
```

```
test.out
/home/gilda002/globus-
tmp.grid022.16511.0/.mpi/https_3a_2f_2fgrid004.ct.infn.it_3a9000_2f4P2yVyypejJXh
c1OuTZlOA:
total 360
drwxr-xr-x 2 gilda002 gilda 4096 Sep 28 15:39 .
drwxr-xr-x 3 gilda002 gilda 4096 Sep 28 15:39 ..
-rw-r--r-- 1 gilda002 gilda 791 Sep 28 15:39 .BrokerInfo
-rw-r--r--       1       gilda002       gilda       0       Sep       28       15:39
.maradona.https_3a_2f_2fgrid004.ct.infn.it_3a9000_2f4P2yVyypejJXhc1OuTZlOA.outpu
t
-rw-r--r-- 1 gilda002 gilda 600 Sep 28 15:39 PI17046
-rwxr-xr-x 1 gilda002 gilda 337541 Sep 28 15:39 cpi
-rwxr-xr-x 1 gilda002 gilda 699 Sep 28 15:39 new_MPItest.sh
-rw-r--r-- 1 gilda002 gilda 0 Sep 28 15:39 test.err
-rw-r--r-- 1 gilda002 gilda 600 Sep 28 15:39 test.out
@@@@@@@@@@@@@@@
Mirroring via SSH to grid021.ct.infn.it
/home/gilda002/globus-
tmp.grid022.16511.0/.mpi/https_3a_2f_2fgrid004.ct.infn.it_3a9000_2f4P2yVyypejJXh
c1OuTZlOA
PI17046
cpi
new_MPItest.sh
test.err
test.out
/home/gilda002/globus-
tmp.grid022.16511.0/.mpi/https_3a_2f_2fgrid004.ct.infn.it_3a9000_2f4P2yVyypejJXh
c1OuTZlOA:
total 356
drwxr-xr-x 2 gilda002 gilda 4096 Sep 28 15:39 .
drwxr-xr-x 3 gilda002 gilda 4096 Sep 28 15:39 ..
-rw-r--r-- 1 gilda002 gilda 600 Sep 28 15:39 PI17046
-rwxr-xr-x 1 gilda002 gilda 337541 Sep 28 15:39 cpi
-rwxr-xr-x 1 gilda002 gilda 699 Sep 28 15:39 new_MPItest.sh
-rw-r--r-- 1 gilda002 gilda 0 Sep 28 15:39 test.err
-rw-r--r-- 1 gilda002 gilda 1427 Sep 28 15:39 test.out
@@@@@@@@@@@@@@@
Mirroring via SSH to grid020.ct.infn.it
/home/gilda002/globus-
tmp.grid022.16511.0/.mpi/https_3a_2f_2fgrid004.ct.infn.it_3a9000_2f4P2yVyypejJXh
c1OuTZlOA
PI17046
cpi
new_MPItest.sh
test.err
test.out
/home/gilda002/globus-
tmp.grid022.16511.0/.mpi/https_3a_2f_2fgrid004.ct.infn.it_3a9000_2f4P2yVyypejJXh
c1OuTZlOA:
total 356
drwxr-xr-x 2 gilda002 gilda 4096 Sep 28 15:39 .
drwxr-xr-x 3 gilda002 gilda 4096 Sep 28 15:39 ..
-rw-r--r-- 1 gilda002 gilda 600 Sep 28 15:39 PI17046
-rwxr-xr-x 1 gilda002 gilda 337541 Sep 28 15:39 cpi
-rwxr-xr-x 1 gilda002 gilda 699 Sep 28 15:39 new_MPItest.sh
-rw-r--r-- 1 gilda002 gilda 0 Sep 28 15:39 test.err
-rw-r--r-- 1 gilda002 gilda 2205 Sep 28 15:39 test.out
@@@@@@@@@@@@@@@
Mirroring via SSH to grid026.ct.infn.it
```

```
/home/gilda002/globus-
tmp.grid022.16511.0/.mpi/https_3a_2f_2fgrid004.ct.infn.it_3a9000_2f4P2yVyypejJXh
c1OuTZlOA
PI17046
cpi
new_MPItest.sh
test.err
test.out
/home/gilda002/globus-
tmp.grid022.16511.0/.mpi/https_3a_2f_2fgrid004.ct.infn.it_3a9000_2f4P2yVyypejJXh
c1OuTZlOA:
total 356
drwxr-xr-x 2 gilda002 gilda 4096 Sep 28 15:39 .
drwxr-xr-x 3 gilda002 gilda 4096 Sep 28 15:39 ..
-rw-r--r-- 1 gilda002 gilda 600 Sep 28 15:39 PI17046
-rwxr-xr-x 1 gilda002 gilda 337541 Sep 28 15:39 cpi
-rwxr-xr-x 1 gilda002 gilda 699 Sep 28 15:39 new_MPItest.sh
-rw-r--r-- 1 gilda002 gilda 0 Sep 28 15:39 test.err
-rw-r--r-- 1 gilda002 gilda 2983 Sep 28 15:39 test.out
@@@@@@@@@@@@@@@
*********************************
Executing cpi
total 348
-rwxr-xr-x 1 gilda002 gilda 337541 Sep 28 15:39 cpi
-rwxr-xr-x 1 gilda002 gilda 699 Sep 28 15:39 new_MPItest.sh
-rw-r--r-- 1 gilda002 gilda 600 Sep 28 15:39 PI17046
-rw-r--r-- 1 gilda002 gilda 0 Sep 28 15:39 test.err
-rw-r--r-- 1 gilda002 gilda 3769 Sep 28 15:39 test.out
```

This is the output of the job after its execution (file **executable.out** in the example above):

```
Process 0 of 4 on grid022.ct.infn.it
pi is approximately 3.1415926544231239, Error is 0.0000000008333307
wall clock time = 10.007429
Process 2 of 4 on grid020.ct.infn.it
Process 3 of 4 on grid026.ct.infn.it
Process 1 of 4 on grid021.ct.infn.it
```

3.7 Submit an MPI Job

Go to **mpi** directory and submit the **mpi.jdl** job

```
$ cd mpi
```

In the output directory you must have

```
$ ls -la
> total 13
drwxr-xr-x    2 rome01   users         128 Jan 27 15:36 .
drwxr-xr-x    7 rome01   users         288 Jan 27 15:36 ..
-rw-r--r--    1 rome01   users         244 Jan 27 15:36 executable.out
-rw-r--r--    1 rome01   users           0 Jan 27 15:36 test.err
-rw-r--r--    1 rome01   users        4162 Jan 27 15:36 test.out
```

```
$ more executable.out
Process 0 of 2 on grid022.ct.infn.it
pi is approximately 3.1415926544231318, Error is 0.0000000008333387
wall clock time = 10.006687
Process 1 of 2 on grid021.ct.infn.it
```

```
$ more test.out
Running on: grid022.ct.infn.it
************************************
PBS Nodefile: /var/spool/pbs/aux/5983.grid010.ct.infn.it
grid022.ct.infn.it
grid021.ct.infn.it
************************************
************************************
Current dir: /home/gilda004/globus-
                    tmp.grid022.25921.0/.mpi/https_3a_2f_2fgrid004.ct.infn.it_3a9000_2f2
                    OczOHHAPEJF4t2CFUShcA
************************************
Mirroring via SSH to grid022.ct.infn.it
/home/gilda004/globus-
                    tmp.grid022.25921.0/.mpi/https_3a_2f_2fgrid004.ct.infn.it_3a9000_2f2
                    OczOHHAPEJF4t2CFUShcA
MPItest.sh
PI26526
cpi
test.err
test.out
/home/gilda004/globus-
                    tmp.grid022.25921.0/.mpi/https_3a_2f_2fgrid004.ct.infn.it_3a9000_2f2
                    OczOHHAPEJF4t2CFUShcA:
total 360
drwxr-xr-x    2 gilda004 gilda        4096 Feb  1 19:40 .
drwxr-xr-x    3 gilda004 gilda        4096 Feb  1 19:40 ..
-rw-r--r--    1 gilda004 gilda         790 Feb  1 19:40 .BrokerInfo
-rw-r--r--    1 gilda004 gilda           0 Feb  1 19:40
                    .maradona.https_3a_2f_2fgrid004.ct.infn.it_3a9000_2f2OczOHHAPEJF4t2C
                    F
UShcA.output
-rwxr-xr-x    1 gilda004 gilda        1931 Feb  1 19:40 MPItest.sh
-rw-r--r--    1 gilda004 gilda         292 Feb  1 19:40 PI26526
-rwxr-xr-x    1 gilda004 gilda      337541 Feb  1 19:40 cpi
-rw-r--r--    1 gilda004 gilda           0 Feb  1 19:40 test.err
-rw-r--r--    1 gilda004 gilda         596 Feb  1 19:40 test.out
@@@@@@@@@@@@@@@@
Mirroring via SSH to grid021.ct.infn.it
/home/gilda004/globus-
                    tmp.grid022.25921.0/.mpi/https_3a_2f_2fgrid004.ct.infn.it_3a9000_2f2
                    OczOHHAPEJF4t2CFUShcA
MPItest.sh
PI26526
cpi
test.err
test.out
/home/gilda004/globus-
                    tmp.grid022.25921.0/.mpi/https_3a_2f_2fgrid004.ct.infn.it_3a9000_2f2
                    OczOHHAPEJF4t2CFUShcA:
total 356
drwxr-xr-x    2 gilda004 gilda        4096 Feb  1 19:40 .
drwxr-xr-x    3 gilda004 gilda        4096 Feb  1 19:40 ..
-rwxr-xr-x    1 gilda004 gilda        1931 Feb  1 19:40 MPItest.sh
-rw-r--r--    1 gilda004 gilda         292 Feb  1 19:40 PI26526
-rwxr-xr-x    1 gilda004 gilda      337541 Feb  1 19:40 cpi
-rw-r--r--    1 gilda004 gilda           0 Feb  1 19:40 test.err
-rw-r--r--    1 gilda004 gilda        1419 Feb  1 19:40 test.out
@@@@@@@@@@@@@@@@
**********************************
Executing cpi
total 348
-rwxr-xr-x    1 gilda004 gilda      337541 Feb  1 19:40 cpi
-rwxr-xr-x    1 gilda004 gilda        1931 Feb  1 19:40 MPItest.sh
-rw-r--r--    1 gilda004 gilda         292 Feb  1 19:40 PI26526
-rw-r--r--    1 gilda004 gilda           0 Feb  1 19:40 test.err
-rw-r--r--    1 gilda004 gilda        2197 Feb  1 19:40 test.out
```

```
**********
```

4 DATA MANAGEMENT

The Data Management tools allow allow users to copy files between UI, CE, WN and a SE, to register entries in the RLS and replicate files between SEs.

To perform those actions, several commandscan be used. The name and functionality overview of them is shown in the following table.

lcg-aa	Adds an alias in RMC for a given GUID
lcg-cp	Copies a Grid file to a local destination
lcg-cr	Copies a file to a SE and registers the file in the LRC
lcg-del	Deletes one file (either one replica or all replicas)
lcg-gt	Gets the TURL for a given SURL and transfer protocol
lcg-infosites	Gives information about resources on the Grid
lcg-la	Lists the aliases for a given LFN, GUID or SURL
lcg-lg	Gets the GUID for a given LFN or SURL
lcg-lr	Lists the replicas for a given LFN, GUID or SURL
lcg-ra	Removes an alias in RMC for a given GUID
lcg-rep	Copies a file from one SE to another SE and registers it in the LRC
lcg-rf	Registers in the LRC (and optionally in the RMC) a file residing on an SE
lcg-uf	Unregisters in the LRC a file residing on an SE

Each command has a different syntax (arguments and options), but the **`--vo <vo name>`** option to specify the virtual organization of the user is present in all the commands, except for **`lcg-gt`**.

4.1 Referencing files

Files stored on the Grid can be reference in a numnber of different ways:

GUID : Globally Unique ID
LFN : Logical File Name
SURL : Storage URL
TURL : Transfer URL

A **GUID** identifies a file uniquely. Each file can have only a single GUID. It takes the form:

```
guid:<40_bytes_unique_string>
```

e.g.

```
guid:38ed3f60-c402-11d7-a6b0-f53ee5a37e1d
```

An **LFN** or User Alias can be used to refer to a file in the place of the GUID. Multiple LFNs can be assigned to a GUID. The LFN has the format:

```
lfn:<anything_you_want>
```

e.g.

```
lfn:importantResults/Test1240.dat
```

A **SURL** identifies a replica in a SE, is of the form:

```
sfn://<SE_hostname><SE_Accesspoint><VO_path><filename>
```

e.g.

```
sfn://tbed0101.cern.ch/flatfiles/SE00/dteam/generated/2004-02 \
26/file3596e86f-c402-11d7-a6b0-f53ee5a37e1d
```

Finally, a **TURL** can be used to physically access a the data associated with a SURL, it has the form:

```
<protocol>://<SE_hostname><SE_Accesspoint><VO_path><filename>
```

e.g.

```
gsiftp://tbed0101.cern.ch/flatfiles/SE00/dteam/generated/2004-02- \
26/file3596e86f-c402-11d7-a6b0-f53ee5a37e1d
```

While a SURL is stored in the Replica Catalogue, the TURL is obtained either by the information provided in the Information Service (in the case of a classical SE) or by the SRM (when these are used).

4.2 Retrieving information about the Grid

To use the Replica Manager and Data management commands, we first need to know something about the available resources. We use a command like :

```
$ lcg-infosites --vo gilda all nousedspace
```

This command returns all CEs and SEs that are published in the IS for the specified VO (gilda).

For each CE, the number of CPUs and the running and queued jobs are given.

For each SE, the total and available space figures are given.

Try it:

```
$ lcg-infosites --vo gilda all nousedspace

LRC endpoint for gilda:  http://grid008.ct.infn.it:8080/gilda/edg-local-replica-
                    catalog/services/edg-local-replica-catalog
RMC endpoint for gilda:  http://grid008.ct.infn.it:8080/gilda/edg-replica-metadata-
                    catalog/services/edg-replica-metadata-catalog

**************************************************************
These are the related data for gilda: (in terms of CPUs)
**************************************************************

#CPU    Free    Total Jobs      Running Waiting ComputingElement
-------------------------------------------------------------
```

```
  24      24       0              0        0    ce.grid.unipg.it:2119/jobmanager-lcgpbs-long
  24      24       0              0        0    ce.grid.unipg.it:2119/jobmanager-lcgpbs-short
   6       6       0              0        0    grid4.na.astro.it:2119/jobmanager-lcgpbs-long
   2       1       1              0        1    cetilab.tilab.com:2119/jobmanager-lcgpbs-long
   6       6       0              0        0    grid4.na.astro.it:2119/jobmanager-lcgpbs-short
  24      24       0              0        0    ce.grid.unipg.it:2119/jobmanager-lcgpbs-infinite
   4       3       0              0        0    ced-ce0.datagrid.cnr.it:2119/jobmanager-lcgpbs-long
   2       1       1              1        0    cetilab.tilab.com:2119/jobmanager-lcgpbs-short
  16      15       0              0        0    grid010.ct.infn.it:2119/jobmanager-lcgpbs-long
   4       3       0              0        0    ced-ce0.datagrid.cnr.it:2119/jobmanager-lcgpbs-short
  16      15       0              0        0    grid010.ct.infn.it:2119/jobmanager-lcgpbs-short
   1       1       0              0        0    grid002.mporzio.astro.it:2119/jobmanager-lcgpbs-long
   6       6       0              0        0    grid4.na.astro.it:2119/jobmanager-lcgpbs-infinite
   2       1       4              0        4    cetilab.tilab.com:2119/jobmanager-lcgpbs-infinite
   4       4       0              0        0    dgt01.ui.savba.sk:2119/jobmanager-lcgpbs-long
   4       4       0              0        0    gilda-ce-01.pd.infn.it:2119/jobmanager-lcgpbs-long
   1       1       0              0        0    grid002.mporzio.astro.it:2119/jobmanager-lcgpbs-
                       short
   4       3       1              1        0    ced-ce0.datagrid.cnr.it:2119/jobmanager-lcgpbs-
                       infinite
   4       4       0              0        0    dgt01.ui.savba.sk:2119/jobmanager-lcgpbs-short
   4       4       0              0        0    gilda-ce-01.pd.infn.it:2119/jobmanager-lcgpbs-short
  16      15       1              1        0    grid010.ct.infn.it:2119/jobmanager-lcgpbs-infinite
   2       2       0              0        0    ce01vidgrid.pri.univie.ac.at:2119/jobmanager-lcgpbs-
                       long
   2       2       0              0        0    ce01vidgrid.pri.univie.ac.at:2119/jobmanager-lcgpbs-
                       short
   1       1       0              0        0    grid002.mporzio.astro.it:2119/jobmanager-lcgpbs-
                       infinite
   4       4       0              0        0    dgt01.ui.savba.sk:2119/jobmanager-lcgpbs-infinite
   4       4       0              0        0    gilda-ce-01.pd.infn.it:2119/jobmanager-lcgpbs-
                       infinite
  52      50       0              0        0    skurut1.cesnet.cz:2119/jobmanager-lcgpbs-gilda
   2       2       0              0        0    ce01vidgrid.pri.univie.ac.at:2119/jobmanager-lcgpbs-
                       infinite
   3       3       0              0        0    gridgl001.gl2006europe.com:2119/jobmanager-lcgpbs-
                       long
   3       3       0              0        0    gridgl001.gl2006europe.com:2119/jobmanager-lcgpbs-
                       short
   3       3       0              0        0    gridgl001.gl2006europe.com:2119/jobmanager-lcgpbs-
                       infinite
------------------------------------------------------------

The total values are:
------------------------------------------------------------
 218     207       8       3       5

************************************************************
These are the related data for gilda: (in terms of SE)
************************************************************

Avail Space(Kb) Used Space(Kb)          SEs
------------------------------------------------------------
224464068       3103892                 grid3.na.astro.it
33800064        2192956                 setilab.tilab.com
16082992        2010628                 ced-se0.datagrid.cnr.it
1543554640      603797932               grid009.ct.infn.it
71844040        174352                  dgt02.ui.savba.sk
522799056       752784                  gilda-se-01.pd.infn.it
```

4.3 Uploading a file from the UI to the Grid

Create a file in your home directory:

```
$ echo "Oh, yes, this is really important ..." > important-file.txt
```

Choose the ‘best’ available SE (i.e. one which is nearby your CE and has enough space available)

```
$ lcg-infosites --vo gilda se
************************************************************
```

```
These are the related data for gilda: (in terms of SE)
************************************************************

Avail Space(Kb) Used Space(Kb)          SEs
------------------------------------------------------------
224464124       3103836               grid3.na.astro.it
33800156        2192864               setilab.tilab.com
16083160        2010460               ced-se0.datagrid.cnr.it
1543555728      603796844             grid009.ct.infn.it
71844040        174352                dgt02.ui.savba.sk
522799056       752784                gilda-se-01.pd.infn.it
```

Upload the file on the grid.

Important: you should choose your own logical filename (LFN), e.g. "sabrina-gilda-important" (please try to choose a name that is unique). From now on in the text, when you see "sabrina-gilda-important" you'll substitute the unique LFN that you chose.

```
$ lcg-cr --vo gilda -d grid009.ct.infn.it -l lfn:sabrina-gilda-important
                                                  file://`pwd`/important-file.txt
```

A typical output is as follows (the GUID, of course will be unique):

```
guid:ad3e4979-a576-43f9-9531-491a7600db1c
```

Take note of the GUID (i.e. select and copy) for the next step:

4.4 Listing replicas, GUIDs and aliases

The **`lcg-lr`** command allows users to list all the replicas of a file that have been successfully registered within the Replica Location Service.

List the replicas of the registered file using the GUID:

```
$ lcg-lr --vo gilda guid:<your GUID here>
```

A typical output is as follows:

```
sfn://grid009.ct.infn.it/flatfiles/SE00/gilda/generated/2005-01-27/file02881fcc-1240- \
                                45fd-991a-fbcb2f660c4f
```

List the replicas of the registered file using the LFN (use the same LFN you chose before):

```
$ lcg-lr --vo gilda lfn:sabrina-gilda-important
```

A typical output is as follows:

```
sfn://grid009.ct.infn.it/flatfiles/SE00/gilda/generated/2005-01-27/file02881fcc-1240- \
                                45fd-991a-fbcb2f660c4f
```

Please take note of the SFN for the next execrsise.

List the GUID of the registered file using the LFN:

```
$ lcg-lg --vo gilda lfn:sabrina-gilda-important

guid:ad3e4979-a576-43f9-9531-491a7600db1c
```

List the GUID of the registered file using the SURL

```
$ lcg-lg --vo gilda sfn:<your SFN here>

guid:ad3e4979-a576-43f9-9531-491a7600db1c
```

List the aliases (LFN) of the registered files using the GUID:

```
$ lcg-la --vo gilda guid:<your GUID here>

lfn:sabrina-gilda-important
```

List the aliases (LFN) of the registered files using the SURL:

```
$ lcg-la --vo gilda sfn://grid009.ct.infn.it/flatfiles/SE00/gilda/generated/2005-01-
                     27/file02881fcc-1240-45fd-991a-fbcb2f660c4f

 lfn:sabrina-gilda-important
```

Once a file is stored on an SE and registered within the Replica Location Service, the file can be replicated using the **`lcg-rep`** command :

```
$ lcg-rep --vo gilda -d gilda-se-01.pd.infn.it lfn:sabrina-gilda-important
```

4.5 Copying files out of the Grid

The **`lcg-cp`** command can be used to copy a Grid file to a non-grid storage resource.

This is useful to have a local copy of the file. The command accepts the LFN, GUID or one SURL of the file, as it is shown in the following example:

```
$ lcg-cp --vo gilda lfn:sabrina-gilda-important file://`pwd`/new-gilda-important
```

```
$cat file://`pwd`/new-gilda-important
```

4.6 Deleting replicas

A file that is stored on a Storage Element and registered with a catalog can be deleted using the **`lcg-del`** command.

If a SURL is provided as argument, then that particular replica will be deleted. If a LFN or GUID is given instead, then the **`-s <SE>`** option must be used to indicate which one of the replicas must be erased, unless the **`-a`** option is used, in which case all replicas of the file will be deleted and unregistered (on a best-effort basis).

If the deleted replica was the last or only valid replica, the entries corresponding to its GUID are also removed from the RMC (including aliases).

The following series of commands show how to delete one particular replica of a file, and also all the available replicas (in the following example, make sure you specify your unique GUID and LFN that you obtained before) :

```
$ lcg-del --vo gilda -a guid:ad3e4979-a576-43f9-9531-491a7600db1c
```

Use the **`lcg-lr`** command to check that the replicas have been deleted:

```
$lcg-lr --vo gilda guid:ad3e4979-a576-43f9-9531-491a7600db1c

lcg_lr: No such file or directory


$lcg-lr --vo gilda lfn:sabrina-gilda-important

lcg_lr: No such file or directory
```

4.7 Job Submission with Output Data

First, lets choose a unique logical file name (LFN) : **`Rome<your_unique_ID>.out`**

e.g. **`Rome01.out`**

(from now on in the text, whenever you see **`lfn:Rome01.out`**, you'll use your chosen LFN instead).

Use the **`lcg-lr`** (locate replica) command to check that no replica already exists with your chosen name:

```
$ lcg-lr --vo gilda lfn:Rome01.out

lcg_lr: No such file or directory
```

(if a replica already exists, please choose a different name)

Now go to the JDL file directory:

```
$cd JobSubmission/Torino/Job+Data/
```

Lets examine the JDL file we will use in the next exercise :

```
$ more JobWithOutput.jdl
```

```
Type = "job";
JobType = "normal";

Executable = "scriptOutput.sh";
Arguments = "Giuseppe";

VirtualOrganisation = "gilda";
```

```
StdOutput = "sim.out";
StdError = "sim.err";

InputSandbox = {"scriptOutput.sh"};
OutputSandbox = {"sim.out","sim.err"};

OutputData = {
        [
                OutputFile = "TorinoXX.out";
                LogicalFileName = "lfn:TorinoXX.out";
                StorageElement = "grid009.ct.infn.it";
        ]
};

Requirements = (other.GlueCEInfoTotalCPUs > 4);
Rank = (other.GlueCEStateFreeCPUs);
RetryCount = 0;
```

Notice the job executes the script **scriptOutput.sh** and passes it one argument (your name - or in the example, **Giuseppe**'s name) :

```
$ more scriptOutput.sh
#!/bin/sh
/bin/echo Hello $1 and welcome to the EGEE tutorial! > TorinoXX.out
```

In the JDL file, change the **OutputFile** and **LogicalFileName** attributes, inserting your tutorial unique LFN you chose before.

e.g. change :

OutputFile = "TorinoXX.out"

to :

OutputFile = "Rome01.out"

(You may also like to substitute 'Giuseppe' with your own name)

Make the same change to the LFN in the job script:

```
$ vi scriptOutput.sh
```

```
#!/bin/sh
/bin/echo Hello $1 and welcome to the EGEE tutorial! > TorinoXX.out
```

The command **edg-job-list-match <JDL file>** allows us to check which CEs are eligible to run a job specified by a given JDL file.

Try it :

```
$ edg-job-list-match JobWithOutput.jdl
Selected Virtual Organisation name (from JDL): gilda
Connecting to host grid004.ct.infn.it, port 7772

**********************************************************************
                        COMPUTING ELEMENT IDs LIST
 The following CE(s) matching your job requirements have been found:

                     *CEId*
 skurut1.cesnet.cz:2119/jobmanager-lcgpbs-gilda
 ce.grid.unipg.it:2119/jobmanager-lcgpbs-infinite
```

```
 ce.grid.unipg.it:2119/jobmanager-lcgpbs-long
 ce.grid.unipg.it:2119/jobmanager-lcgpbs-short
 grid010.ct.infn.it:2119/jobmanager-lcgpbs-infinite
 grid010.ct.infn.it:2119/jobmanager-lcgpbs-long
 grid010.ct.infn.it:2119/jobmanager-lcgpbs-short
 grid4.na.astro.it:2119/jobmanager-lcgpbs-infinite
 grid4.na.astro.it:2119/jobmanager-lcgpbs-long
 grid4.na.astro.it:2119/jobmanager-lcgpbs-short
**************************************************************************
```

We can be reasonably confident the Resource Broker will send our job to one of the CEs listed.

Now let us submit the job :

```
$ edg-job-submit -o myjobID JobWithOutput.jdl
$ edg-job-submit -o myjobID JobWithOutput.jdl
Selected Virtual Organisation name (from JDL): gilda
Connecting to host grid004.ct.infn.it, port 7772
Logging to host grid004.ct.infn.it, port 9002

********************* edg-job-submit successful *************************
 The job has been successfully submitted to the Network Server.
 Use edg-job-status command to check job current status. Your job identifier
 (edg_jobId) is:

 - https://grid004.ct.infn.it:9000/2O0zH-cYWT3TopamPD5ccA

 The edg_jobId has been saved in the following file:
 /home/rome01/JobSubmission/Torino/Job+Data/myjobID
**************************************************************************
```

Notice we used the **`-o <jobId file>`** option. This places the unique id of our job in the specified file **`myjobID`** , so that it we can use it later on :

```
$ more myjobID
###Submitted Job Ids###
https://grid004.ct.infn.it:9000/2O0zH-cYWT3TopamPD5ccA
```

Now we can check the status of our job using the **`edg-job-status`** command. Here is where we use the **`-i <jobId file>`** option to retrieve our previous jobId :

```
$ edg-job-status -i myjobID

*************************************************************
BOOKKEEPING INFORMATION:

Status info for the Job : https://grid004.ct.infn.it:9000/2O0zH-cYWT3TopamPD5ccA
Current Status:     Scheduled
Status Reason:      Job successfully submitted to Globus
Destination:        skurut1.cesnet.cz:2119/jobmanager-lcgpbs-gilda
reached on:         Mon Jan 31 16:46:54 2005
```

Notice the job status : **`Current Status:  Scheduled`**

When our job is completed the status will change to **`Done`**

If all went well, our replica will also have been created. Lets check it:

```
$ lcg-lr --vo gilda lfn:Rome01.out

sfn://grid009.ct.infn.it/flatfiles/SE00/gilda/generated/2005-01-31/file83e381d6-73aa-
                    11d9-ad41-c4b3adf3b832
```

4.8 Job Submission with Input Data

Now we will try a similar exercise, but we will specifiy the **`InputData`** attribute for the job. This will ensure our job is sent to a CE which is located 'close' to the SE where the data is stored.

Lest examine the JDL file used in this exercise. As before, change the LFN in the JDL file and the script file to **`Rome01.out`**

```
$ more JobWithInput.jdl
Type = "job";
JobType = "Normal";

Executable = "scriptInput.sh";
Arguments = "Francesco";

VirtualOrganisation = "gilda";

StdOutput = "std.out";
StdError = "std.err";

InputSandbox = {"scriptInput.sh"};
OutputSandbox = {"std.out","std.err"};

InputData = "lfn:TorinoXX.out";
DataAccessProtocol = {"gsiftp","rfio"};

Requirements = (other.GlueCEInfoTotalCPUs > 4);
Rank = (other.GlueCEStateFreeCPUs);
RetryCount = 0;
```

Look at the job script **`scriptInput.sh`**

What does it do ?

```
$ more scriptInput.sh
#!/bin/sh
lcg-cp --vo gilda lfn:TorinoXX.out file:`pwd`/TorinoXX.out
echo "Before.."
cat TorinoXX.out
# Adding new entry on the dataset1.out file.
/bin/echo Hello $1 and welcome to the EGEE tutorial! >> TorinoXX.out
echo "After.."
cat TorinoXX.out
```

Now, check all is well using the **`edg-job-list-match`** command

```
$ edg-job-list-match JobWithInput.jdl
```

The Resource Broker should match your job to a CE which has associated with it a 'close' SE where your replica is stored.

Now submit the job:

```
$ edg-job-submit -o myjobID JobWithInput.jdl
```

Check the status of your job (note, you now have a second job ID) :

```
$ edg-job-status -i myjobID
------------------------------------------------------------------
1 : https://grid004.ct.infn.it:9000/2O0zH-cYWT3TopamPD5ccA
2 : https://grid004.ct.infn.it:9000/obVLm6YLTpLc_p61q2q8nQ
a : all
q : quit
------------------------------------------------------------------

Choose one or more edg_jobId(s) in the list - [1-2]all:2
************************************************************
BOOKKEEPING INFORMATION:

Status info for the Job : https://grid004.ct.infn.it:9000/obVLm6YLTpLc_p61q2q8nQ
Current Status:     Scheduled
Status Reason:      Job successfully submitted to Globus
Destination:        grid010.ct.infn.it:2119/jobmanager-lcgpbs-infinite
reached on:         Mon Jan 31 17:27:56 2005
$ edg-job-get-output -i myjobID
------------------------------------------------------------------
1 : https://grid004.ct.infn.it:9000/2O0zH-cYWT3TopamPD5ccA
2 : https://grid004.ct.infn.it:9000/obVLm6YLTpLc_p61q2q8nQ
a : all
q : quit
------------------------------------------------------------------

Choose one or more edg_jobId(s) in the list - [1-2]all:2
Retrieving files from host: grid004.ct.infn.it ( for
                    https://grid004.ct.infn.it:9000/obVLm6YLTpLc_p61q2q8nQ )

************************************************************************
                        JOB GET OUTPUT OUTCOME

 Output sandbox files for the job:
 - https://grid004.ct.infn.it:9000/obVLm6YLTpLc_p61q2q8nQ
 have been successfully retrieved and stored in the directory:
 /home/rome01/JobOutput/rome01_obVLm6YLTpLc_p61q2q8nQ

************************************************************************
```

When the job status reaches **`Done`**, you can check the output directory (copy/paste the Output sandbox directory from your job status result):

```
$ ls -l /home/rome01/JobOutput/rome01_obVLm6YLTpLc_p61q2q8nQ/*
```

See the result of your job:

```
$cat /home/rome01/JobOutput/rome01_obVLm6YLTpLc_p61q2q8nQ/std.out
```

END