GETTING STARTED WITH ARAGRID

A BRIEF USER GUIDE FOR ARAGRID MEMBERS

Document Log

Issue	Date	Comment	Author/Partner
0.0	03-09-2009	EGEE document	Guillermo Losilla Jaime Ibar (BIFI)
0.1	14-03-2010	First PireGrid version used for the training session at BIFI	Patricia Santos (BIFI)
0.2	17-05-2011	First AraGrid version	Patricia Santos (BIFI)
0.3	24-10-2012	General update	Jaime Ibar (BIFI)
0.4	03-12-2013	General update	Jaime Ibar (BIFI)

This guide has been written by the grid management staff at the Biocomputing and Physics of Complex Systems Research Institute (“BIFI”, Zaragoza, Spain).

e-mail: aragrid-admin[AT_NO_SPAM]bifi.es
http://www.aragrid.es

TABLE OF CONTENTS

0. ABOUT THIS DOCUMENT

This document is not intended to be a generic Glite user guide. It should be considered just a brief summary of the main concepts and commands an AraGrid end user should know. It has been elaborated by the grid management staff at the Biocomputing and Physics of Complex Systems Research Institute ("BIFI", Zaragoza, Spain).

The contents of this document are not guaranteed to be "up-to-date" and people looking for more in depth and updated information about using the gLite grid middleware, should refer to the official "Glite user manual" [R1]

1. INTRODUCTION TO ARAGRID

Grid technology allows to share computing power and storage capacity transparently across the Internet; it somehow tries to "virtualize" the computing resources.

Each computer hosted in each site, is dedicated to perform a specific task and so they are classified as different "node types": "User Interface", "Computing Element", "Worker Node", "Storage Element", "Resource Broker/Workload Management System", "Proxy Server"...

The special software which makes possible to integrate all these disparate kinds of computers is called "grid middleware". The current grid middleware version running in AraGrid is "UMD2".

Grid middleware is formed by three main components:

- **Workload Management System:** provides the mechanisms to submit jobs to the Grid
- **Data Management System:** allows the management of data across the Grid
- **Information System:** provides information about grid resources

Each of them is in turn composed of different services which are distributed across the different "node types" we explained before.

2. INITIAL SETUP

2.1 Obtain a digital certificate

The first step an AraGrid user must complete is to obtain a digital certificate issued by one of the Certification Authorities supported by the project. It is necessary because most grid services require to own a digital certificate to be able to interact with them (SSL based communication). This way users access the Grid in a secure manner.

2.2 Join a Virtual Organization

As a second step, the user must join (at least) one of the Virtual Organizations the AraGrid users are grouped in. Sites give access to their resources to the VOs they choose, so it is not possible for a user to exploit the Grid, unless gaining membership in (at least) one Virtual Organization.

The procedure to obtain a digital certificate and join to the AraGrid VOs is documented at **[R2].**

2.3 Setting up the user account

Finally, every AraGrid user needs an account on a “User Interface” box. It is a usual Unix/Linux account where the user should copy his digital certificate in PEM format under the directory $HOME/.globus:

```
user@UserInterface$ ll /home/user/.globus
                -r--r--r--   1 user  user      3807 Jan 12 11:33 usercert.pem
                -r--------   1 user  user       963 Jan 12 11:39 userkey.pem
```

(NOTE: due to security reasons, file permissions are very important!)

3. PROXY CERTIFICATE MANAGEMENT

Once logged in his/her User Interface's account, the first thing a user must do before interacting with the Grid, is to generate a proxy certificate, which is somehow a “temporary certificate” used for the actual authentication with grid resources. The proxy is derived from the user's personal certificate and it has a limited duration (12 hours by default) since it is not protected by a password.

VOMS allows having one certificate registered with several VOs and the user only needs to specify the VO he wants to use at the moment of creating the proxy certificate. Therefore the user will not have to specify it using the *--vo* option in further commands.

3.1 Create a proxy certificate

Requires the user to supply the pass phrase which protects the user's certificate

```
[user@ui-ara$ ] > voms-proxy-init --voms  vo.unizar.aragrid.es
Enter GRID pass phrase:
Your identity: /DC=es/DC=irisgrid/O=bifi-unizar/CN=jaime-ibar
Creating temporary proxy ..................................... Done
Contacting  voms-prg.bifi.unizar.es:15005
[/DC=es/DC=irisgrid/O=bifi-unizar/CN=voms-prg.bifi.unizar.es] "vo.unizar.aragrid.es" Done
Creating proxy ................................................. Done
Your proxy is valid until Tue Dec  3 22:57:19 2013
```

3.2 Print info about current proxy

This command shows grid proxy information and voms extensions.

```
[user@ui-ara$] > voms-proxy-info –all
subject   : /DC=es/DC=irisgrid/O=bifi-unizar/CN=jaime-ibar/CN=proxy
issuer    : /DC=es/DC=irisgrid/O=bifi-unizar/CN=jaime-ibar
identity  : /DC=es/DC=irisgrid/O=bifi-unizar/CN=jaime-ibar
type      : proxy
strength  : 1024 bits
path      : /tmp/x509up_u1000
timeleft  : 11:58:42
=== VO vo.unizar.aragrid.es extension information ===
VO        : vo.unizar.aragrid.es
subject   : /DC=es/DC=irisgrid/O=bifi-unizar/CN=jaime-ibar
issuer    : /DC=es/DC=irisgrid/O=bifi-unizar/CN=voms-prg.bifi.unizar.es
attribute : /vo.unizar.aragrid.es/Role=NULL/Capability=NULL
timeleft  : 11:58:42
uri       : voms-prg.bifi.unizar.es:15005
```

3.3 Destroy a proxy certificate

This command destroy the proxy

[user@ui-ara]$ ***voms-proxy-destroy***

3.4 Change the passphrase which protects the user's certificate

[user@ui-ara]$ ***grid-change-pass-phrase***

4. JOB SUBMISSION

The computing resources in AraGrid are abstracted by a special node type called “Computing Element” (CE), behind of which normally exists a cluster running any of the most common batch schedulers (PBS, SGE, Condor, LSF...). This section describes the whole process of submitting a simple job in the Grid using the command line interface present in every User Interface.

Imagine we have a very simple program called “add” which reads 2 integers from standard input and stores the addition in the file “result.dat”. We use the following script (“test.sh”) to invoke our program:

[user@ui-ara]$ ***cat test.sh***

```
#!/bin/bash
echo "Starting program..."
./add 3 12
echo "Execution finished!"
```

4.1 Create the proxy certificate

As it was described in section 3.1, you have to create a proxy certificate every time you initiate a session in the Grid.

4.2 Write a JDL file

JDL is the special language used to describe a job before its submission to the Grid. A JDL file is composed by several statements (definition of attributes) ended by a semicolon. Some attributes are mandatory: *Executable* (specifies the name of the executable), *StdOutput* (tells the name of the file where the info printed by the job in the standard output is saved) and *StdError* (idem for the standard error). Some others are optional but very common like *InputSandbox* (files to be transferred

to the Grid before job execution) and *OutputSandbox* (files to be transferred from the Grid after job execution).

There are other interesting attributes like *StdInput* (sets the standard input), *Arguments* (arguments to be supplied to the executable) and *Environment* (allows setting environment variables for the job). The whole range of attributes allowed in a JDL file can be found in **[R3]**.

The JDL file for our simple program (sample.jdl) could be:

```
[user@ui-ara]$ cat sample.jdl
        Executable = "test.sh";
        StdOutput = "test.out";
        StdError = "test.err";
        InputSandbox = {"test.sh","add"};
        OutputSandbox = {"result.dat","test.out","test.err"};
```

Since the executable flag is not preserved for the files included in the InputSandbox (except for the file specified as Executable), we have to make a slight modification in our invoking script (test.sh):

```
[user@ui-ara]$ cat test.sh
        #!/bin/bash
        echo "Starting program..."
        chmod u+x add
        ./add 3 12
        echo "Execution finished!"
```

4.3 Listing computing resources available for a VO

With the *lcg-infosites* command, we can discover the AraGrid sites which accept jobs for a given VO. For instance, following is the order to list sites accepting jobs from your VO.

```
[user@ui-ara]$ lcg-infosites --vo vo.unizar.aragrid.es ce
#CPU    Free    Total Jobs      Running Waiting ComputingElement
----------------------------------------------------------
 432    408     0               0       0       cream-eupt.unizar.es:8443/cream-pbs-unizara
 432    216     0               0       0       cream-ciencias.bifi.unizar.es:8443/cream-pbs-unizara
 432    216     0               0       0       cream-ara.bifi.unizar.es:8443/cream-pbs-unizara
```

4.3 Delegating our credentials to the WMS

Once created our voms proxy, we have to delegate our credentials to the WMS.

4.3.1 – Automatic delegation

```
[user@ui-ara]$ glite-wms-job-delegate-proxy  -a
Connecting to the service https://wms-ara.bifi.unizar.es:7443/glite_wms_wmproxy_server
================== glite-wms-job-delegate-proxy Success ==================
    Your proxy has been successfully delegated to the WMProxy:
    https://wms-ara.bifi.unizar.es:7443/glite_wms_wmproxy_server
    with the delegation identifier: Ix48jyUT7gVOoAuAokhmOA
==========================================================================
```

4.4 Listing sites which can execute a job

We can use the *glite-wms-job-list-match* command to find out which sites (among the discovered in the previous section) can execute our job, according to the requirements we specified in the JDL file:

```
[user@ui-ara]$ glite-wms-job-list-match - a sample.jdl

Connecting to the service https://wms-ara.bifi.unizar.es:7443/glite_wms_wmproxy_server

==========================================================================

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

        *CEId*                                  *Rank*

- cream-ara.bifi.unizar.es:8443/cream-pbs-unizara               0
- cream-ciencias.bifi.unizar.es:8443/cream-pbs-unizara          0
- cream-eupt.unizar.es:8443/cream-pbs-unizara                   0

==========================================================================
```

4.5 Submit a job

Once the JDL file is written, the proxy certificate generated and delegated our credentials, we can issue the *glite-wms-job-submit* command to submit a job to the Grid:

```
[user@ui-ara]$ glite-wms-job-submit  -a  -o jobid sample.jdl
Connecting to the service https://wms-ara.bifi.unizar.es:7443/glite_wms_wmproxy_server
================ glite-wms-job-submit Success ====================
        The job has been successfully submitted to the WMProxy
Your job identifier is:
        https://wms-ara.bifi.unizar.es:9000/fzUMbyPypwFGERms-EhJfw
The job identifier has been saved in the following file:
        /home/user/pruebas/jobid
==========================================================================
```

The glite-wms-job-submit command returns a jobID (“job identifier”) which defines uniquely the job and can be used to perform further operations on the job as we will see in the following subsections. In this example, we have passed -o modifier to the command, this way, the job identifier will be stored in a file called jobid.

4.6 Query the status of a job

Until a job is executed somewhere in the Grid, it crosses different states: SUBMMITTED, WAITING, READY, SCHEDULED, RUNNING, DONE...

With the glite-wms-job-status command a user can query the current status of a job:

```
[user@ui-ara]$ glite-wms-job-status  -i  jobid

**************************************************************
BOOKKEEPING INFORMATION:
Status info for the Job : https://wms-ara.bifi.unizar.es:9000/fzUMbyPypwFGERms-EhJfw
                        Current Status:     Done (Success)
                        Logged Reason(s):
                         -
                         - Job terminated successfully
                        Exit code:          0
                        Status Reason:      Job terminated successfully
                        Destination:        cream-ara.bifi.unizar.es:2119/jobmanager-lcgpbs-yourVO
                        Submitted:          Wed Mar 17 16:33:45 2010 CET
**************************************************************
```

4.7 Cancel a job

If for some reason a user needs to cancel a job, the glite-wms-job-cancel command should be used:

```
[user@ui-ara]$ glite-wms-job-cancel  -i  jobid
Connecting to the service https://wms-ara.bifi.unizar.es:7443/glite_wms_wmproxy_server

================ glite-wms-job-cancel Success================

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

                        - https://wms-ara.bifi.unizar.es:9000/JIvAn6RQBOHVDEpEjjlNCg
============================================================
```

4.8 Retrieve the logging info about a submitted job

Using glite-wms-job-logging-info can be used to get a complete trace log of the life-cycle of the job (very useful for debugging!):

[user@ui-ara]$ ***glite-wms-job-logging-info -v 2 -i jobid***
****
LOGGING INFORMATION:
Printing info for the Job : https://wms-ara.bifi.unizar.es:9000/fzUMbyPypwFGERms-EhJfw
---
...<large output>...

4.9 Retrieve the output of a job

Once a job has reached the “Done” status, glite-wms-job-output can be used to retrieve the output of the job (the files which were specified in the OutputSandbox attribute):

[user@ui-ara]$ **glite-wms-job-output -i jobid**

We can find the output of the job in the directory reported by the previous command:

[user@ui-ara]$ ***ls -l /tmp/jobOutput/user_lO0-xMwkdxEKYnY9Z2sdBw***

5. DATA HANDLING

One of the most attractive advantages that the Grid offers the end user is “almost” unlimited storage capacity. The storing resources in AraGrid are abstracted by Storage Elements (SE) which can be of different types, for example “Disk Pool Manager” (DPM) . Behind these nodes may exist a simple disk, a tape server, a pool of disks...

In this section we will briefly analyze the most commonly used tools provided by the AraGrid middleware for data management.

5.1 File naming in AraGrid

There are 4 different types of names that can be used to refer to a file that has been stored in the Grid:

- **Grid Unique Identifier (GUID):** which identifies a file uniquely, example:
  guid:0efb1de3-3522-4917-9d21-b2570acd9ee7
- **Logical File Name (LFN):** which is an alias (human-readable) of the GUID of a file:
  lfn:/grid/your_VO/user/library.lib
- **Storage URL (SURL):** which identifies a replica of a file stored in a Storage Element:
  sfn://dpm-ara.bifi.unizar.es/storage/your_VO/generated/2006-02-21/filee069e779-486e-4546-a1d
- **Transport URL (TURL):** which is a valid URI with the necessary information to access a file in a SE:
  gsiftp://dpm-ara.bifi.unizar.es/storage/your_VO/generated/2006-02-21/filee069e779-486e-4546-a1d

5.2 File catalogs

The mapping between the logical file names (LFNs) , the corresponding GUIDs and the physical location of the replicas (SURLs), is stored in a special node type called “LFC catalog”. The LFNs stored in a LFC catalog follows a hierarchical namespace (Unix-like): /grid/vo.unizar.aragrid.es/<dir>/... For example a valid LFN of a file could be */grid/vo.unizar.aragrid.es/user/my_file.*

There must be one catalog per Virtual Organization. For instance, the LFC catalog for AraGrid VOs is hosted at *lfc-ara.bifi.unizar.es.*

5.3 Proxy certificate creation and environment settings

Imagine we have a big file (library.lib) which contains a library we need to use in some jobs we want to execute in the Grid. In the following subsections we will see how to store, retrieve and clean several replicas of that file across the Grid using the Command Line Interface.

As it happened with job submission, remember you have to create a proxy certificate every time you initiate a session in the Grid. It was described in section 3.1

Also note that some of the commands (lfc-*) we will use, need the LFC_HOST variable to be set with the URL of the VO's LFC catalog we belong to. For instance, in the case of AraGrid we should issue:

[user@UserInterface]$ ***export LFC_HOST=lfc-ara.bifi.unizar.es***

5.4 List the contents of a catalog

We can easily browse the contents of the catalog using the command *lfc-ls*:

[user@ui-ara]$ ***lfc-ls /***
grid
[user@ui-ara]$ ***lfc-ls /grid***
...
vo.unizar.aragrid.es
...

5.5 Create a directory in a catalog

With *lfc-mkdir* we create dirs in the catalog:

[user@ui-ara]$ ***lfc-mkdir /grid/vo.unizar.aragrid.es/user***

5.6 Listing storage resources available for a VO

With the *lcg-infosites* command, we can discover the AraGrid sites which offers storage capacity (Storage Elements) for a given VO. For instance, following is the order to list sites accepting files from AraGrid VOs:

[user@ui-ara]$ ***lcg-infosites --vo vo.unizar.aragrid.es se***

```
Avail Space(Kb) Used Space(Kb)  Type    SEs
----------------------------------------------------------
1866848145      101762084       n.a     dpm-eupt.unizar.es
1867090255      101519974       n.a     dpm-ciencias.bifi.unizar.es
1834015608      134594621       n.a     dpm-ara.bifi.unizar.es
1868405579      100204650       n.a     dpm-epsh.unizar.es
```

5.7 Copy and register a file in the Grid

Using the command *lcg-cr* we can store a replica of a file in the Grid and register it in a catalog. For instance, in the case of our library file it would be (note we don't specify any Storage Element):

*[user@ui-ara]$* ***lcg-cr --vo vo.unizar.aragrid.es -l /grid/vo.unizar.aragrid.es/user/library.txt file:$(pwd)/library.txt***

guid:0f88793a-38f8-4390-8fcb-0ff017dd9f32

5.8 List the replicas of a file

We can query the replicas of a file stored in the Grid using *lcg-lr*:

[user@ui-ara]$ ***lcg-lr --vo vo.unizar.aragrid.es lfn:/grid/vo.unizar.aragrid.es/user/library.txt***

srm://dpm-ara.bifi.unizar.es/dpm/bifi.unizar.es/home/vo.unizar.aragrid.es/generated/2010-01- 20/file2 a62e0a1-fc35-41af-8aef-9f5903d1227e

5.9 Replicate a file

With *lcg-rep* we can create new replicas of a file stored in the Grid (note we do specify the destination Storage Element):

[*user@ui-ara]$* ***lcg-rep --vo vo.unizar.aragrid.es -d dpm-ara.bifi.unizar.es lfn:/grid/vo.unizar.aragrid.es/user/library.txt***

5.10 Retrieve a copy of a file stored in the Grid

Using lcg-cp we can get a copy of a file stored in the Grid. This should be done from our grid jobs in order to get a copy library.txt from the closest SE:

*[user@ui-ara]$* ***lcg-cp --vo vo.unizar.aragrid.es lfn:/grid/vo.unizar.aragrid.es/user/library.txt file:$(pwd)/my_library.lib***

5.11 File removal

In order to remove all the replicas of a Grid file we should use *lcg-del*:

[user@ui-ara]$ ***lcg-del -a --vo vo.unizar.aragrid.es lfn:/grid/vo.unizar.aragrid.es/user/library.txt***

5.12 Remove a directory from a catalog

Once we have removed the entries of a directory we should remove it using *lfc-rm*:

[user@ui-ara]$ ***lfc-rm -r /grid/vo.unizar.aragrid.es/user/***

6. USER SUPPORT

For a more user support, please visit AraGrid wiki: http://wiki.aragrid.es

If you have any question, you can mail to aragrid-admin[AT_NO_SPAM]bifi.es

7. ADVANCED GRID MANAGEMENT

7.1 Sending jobs to a particular CE

This can be done using the *-r <ceID>* argument of *glite-wms-job-submit*, where <ceID> is the CE string as returned by *glite-wms-job-list-match* (subsection 4.4) or *lcg-infosites* (subsection 4.3). For instance:

[user@ui-ara]$ ***glite-wms-job-submit -a -o jobid -r cream-ara.bifi.unizar.es:cream-ara.bifi.unizar.es:8443/cream-pbs-unizara sample.jdl***

Other posibility is to specify the desired CE in the JDL file:

```
[user@ui-ara]$ cat sample.jdl
        Executable = "test.sh";
        StdOutput = "test.out";
        StdError = "test.err";
        InputSandbox = {"test.sh","a.out"};
        OutputSandbox = {"result.dat","test.out","test.err"};
        Requirements = other.GlueCEUniqueID ==
        "cream-ara.bifi.unizar.es:8443/cream-pbs-unizara";
```

7.2 Long job submission

The proxy certificates created as described in section 3 have an inconvenient: if the job does not finish before the proxy expires, it is 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.

A first solution is to create a proxy certificate with a very long lifetime using the *-valid* option of *voms-proxy-init*. For instance, to generate a proxy certificate valid for 2 days, a user could issue:

```
[user@ui-ara]$ voms-proxy-init --voms vo.unizar.aragrid.es -valid 48:0
```

However this practice increases the security risks since the proxy certificate travels with the job across the Grid. 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 special node type (Proxy Server). The WMS will then be able to use this long-term proxy to periodically renew the proxy for a submitted job before it expires and until the job ends (or the long-term proxy expires).

Following is an example of creation and use of a long term proxy, before submitting a very long AraGrid job. It uses the AraGrid Proxy Server running at *proxy-ara.bifi.unizar.es:*

```
#Create and store a long-term proxy in the Proxy Server
[user@ui-ara]$ myproxy-init -s proxy-ara.bifi.unizar.es -d -n
#Get information about the stored long-term proxy
[user@ui-ara]$ myproxy-info -s proxy-ara.bifi.unizar.es -d
#Add the following attribute to the JDL file of the long job
MyProxyServer = "proxy-ara.bifi.unizar.es";
#Once the job has finished, the user can remove manually the long-term proxy from the Proxy Server
[user@ui-ara]$ myproxy-destroy -s proxy-ara.bifi.unizar.es -d
```

7.3 Dealing with software requirements

It is very common that jobs submitted to the Grid require not only the executable files containing the application, but big files as libraries, input data... This can be a drawback since those files should “travel” across the Grid every time the job is submitted, increasing the latency needed to complete the overall execution of the job due to the time taken to transfer the data. Indeed, resource brokers have restrictions about the total size of the files specified by the user in the input sandbox.

To overcome this limitations a single user could first store the data in a Storage Element and then modify his/her jobs to first download from that SE the needed files in the Worker Node where the job has arrived and then access them as usual.

Virtual organizations use a more advanced strategy which consists in installing the common software needed for the experiments in the sites supporting the VO. This is accomplished by members of the VO with a special role: experiment software managers. Once a software manager has installed a specific version of a experiment software in a site, he labels the site with a “tag” which is automatically published through its information system. An example of tag could be “VO-*your_VO*-example_libraries-v12r1”.

A user can query the sites publishing a specific tag using the command *lcg-info*:

[user@ui-ara]$ ***lcg-info --vo vo.unizar.aragrid.es --list-ce --query 'Tag=VO-your_VO-example_libraries-v12r1'***

Moreover, a user can restrict in the JDL the list of matching CEs for a job, to those which have some especific version of VO software installed:

Requirements = Member("VO-your_VO-example_libraries-v12r1",other.GlueHostApplicationSoftwareRunTimeEnvironment);

7.4 Managing several jobs simultaneously

It is very common for grid users to submit several jobs simultaneously. This can be handled from the shell using the suitable arguments of the command line tools.

Following is an example of simultaneous submission and management of three jobs:

 # Submit 3 jobs saving the jobIDs in a file

```
[user@ui-ara]$ glite-wms-job-submit -o job.list test.jdl
[user@ui-ara]$ glite-wms-job-submit -o job.list test.jdl
[user@ui-ara]$ glite-wms-job-submit -o job.list test.jdl

# Query the status of the 3 jobs
[user@ui-ara]$ glite-wms-job-status -i job.list

# Retrieve the output of the 3 jobs at a time
[user@ui-ara]$ glite-wms-job-output -i job.list
```

Even more practical for the end user, can be the Job Management Framework (**[R4]**); a bunch of scripts which easily automatizes the management of several jobs simultaneously.

7.5 Priorizing the list of matching CEs

The choice of the CE where to execute the job, among all the ones satisfying the requirements, is based on the “rank” of the CE, which is by default the number of free CPUs. To change the rank criteria, a user can use the Rank attribute in the JDL file.

For instance the following jdl sample uses as rank the number of free CPUs in the CE if there are not waiting jobs, otherwise it uses the negative number of waiting jobs:

```
[user@ui-ara]$ cat sample.jdl
        Executable = "test.sh";
        StdOutput = "test.out";
        StdError = "test.err";
        InputSandbox = {"test.sh","a.out"};
        OutputSandbox = {"result.dat","test.out","test.err"};
        Rank = ( other.GlueCEStateWaitingJobs == 0 ? other.GlueCEStateFreeCPUs : -
        other.GlueCEStateWaitingJobs);
```

7.6 AraGrid APIs & GUIs

Remember the intention of this guide was to serve as a starting point for end AraGrid users, showing the basic commands to work in the Grid.

For those looking for an API (Application Program Interface) to interact directly with the Grid from an application, they should look the “LCG User Developer Guide” or the specific documentation of the component of the middleware he/she wants to interact with.

7.7 LINKS & REFERENCES

[R1] Glite user manual: https://edms.cern.ch/file/722398//gLite-3-UserGuide.html
[R2] http://www.aragrid.es/wiki/index.php/T%C3%A9cnica#Soporte_a_usuarios
8.
[R3] http://server11.infn.it/workload-grid/docs/DataGrid-01-TEN-0142-0_2.pdf
[R4] http://goc.grid.sinica.edu.tw/gocwiki/Job_Management_Framework