

EGRID Portal Manual

Table of Contents

- [1. Introduction](#)
- [2. Certificates](#)
 - ◆ [2.1 Proxies and MyProxy server concepts](#)
 - ◆ [2.2 MyProxyManager tool and the Proxy Retrieval Page](#)
- [3. Data management](#)
- [4. Job submission](#)
 - ◆ [4.1 Job submission in gLite grids](#)
 - ◆ [4.2 Job submission through the portal](#)
 - ◇ [4.2.1 Personal portal area](#)
 - ◇ [4.2.2 Creating a workflow](#)
 - ◇ [4.2.3 Launching a workflow](#)
 - ◇ [4.2.4 If the programme is already installed in the grid](#)
 - ◆ [4.3 gLite grids and the portal workflow](#)
 - ◇ [4.3.1 Example of a single job](#)
 - ◇ [4.3.2 Previous example rewritten as a Workflow of 5 jobs](#)
 - ◇ [4.3.3 Previous example rewritten as a Parametric job](#)
 - ◇ [4.3.4 Previous example rewritten as an MPI job](#)
- [5. Information system](#)

1. Introduction

The EGRID Portal allows grid access from any computer connected to the Internet with the help of a Java enabled browser. Mozilla and Firefox have been fully tested while for Internet Explorer tests are still being completed. Some pages of the portal use Applets and Java Web Start: please check that your browser supports both of them and that you have Java 5 (1.5) installed. For some common problems and resolutions, please see Appendix X.

Access to grid storage and to job submission require the portal to hold proxy certificates of your original grid credentials. Before being able to fully use the portal, it is therefore essential that there be at least one of your proxy certificates present. In order to do so, generally you'll first have to create a proxy certificate; you'll then have to upload it into a MyProxy server, which is a grid service purpose-made to safely and temporarily store proxy certificates; finally, you'll have to feed the proxy just created and safely stored, to the portal itself. To carry out all of these steps, the portal provides the Certificates tab: there you will find all the tools you need; these tools are the topic of the next chapter.

2. Certificates

The EGRID Portal works with proxy certificates which you create from your original personal certificate. Like your original personal certificate, a proxy too has an associated expiry time called Proxy Lifetime which you define when you create it. It is typically in the order of hours or days, and in any case it cannot exceed your personal certificate's expiry time.

The portal however cannot get a proxy directly from you: instead it can retrieve your proxy from a MyProxy server where it was previously uploaded. Still, the portal includes the MyProxyManager tool which allows you to create a proxy from your original certificate, and to upload it directly to a MyProxy server of your choice, all in one step. You then have only got to tell the portal to retrieve the proxy.

If you are not familiar with the concepts involved in MyProxy servers, the following paragraphs will provide some extra information. Otherwise you may skip to section 2.2 which describes the full set of steps involved in typical scenarios of the EGRID Portal: the use of the MyProxyManager tool for proxy generation/uploading, and the Proxy Retrieval Page for fetching previously stored proxies.

2.1 Proxies and MyProxy server concepts

The use of proxies instead of original certificates is a safety measure: if they get stolen while working with grid services, because of their shorter lifetime then the damage is automatically limited. The mechanism that allows proxies to represent you for all practical purposes, is called certificates chain.

A MyProxy server is identified by a hostname such as myproxy.cnaf.infn.it, and by a port number such as 7512. The server is also guaranteed to be authentic by a Certification Authority CA, which publishes a special file: the public certificate of the CA. You need all three data when selecting a MyProxy server to store your proxy.

Each proxy kept in a MyProxy server is identified by a Login and a Password, both of which you choose at the time of uploading the proxy: in this way only who knows that login and that password can retrieve your proxy. There is also a third optional identification parameter for the proxy: the Credential name; it is needed for special functionality (Proxy Renewal) currently not supported by the portal, and therefore it will not be explained further.

Two lifetimes must also be specified when uploading the proxy: the Credential Lifetime and the Proxy Lifetime. The Credential Lifetime tells how long you want the MyProxy server to hold your proxy; when time is up your proxy will be automatically deleted from the server. The Proxy Lifetime tells how long you want a retrieved proxy to be valid for Grid usage; upon expiry Grid services will not accept that retrieved proxy as valid, so a new one will have to be retrieved for continuing to use the Grid. An example on the meaning of the two lifetimes follows.

Suppose that for about a week you will be working with the Grid from several computers: your personal laptop, the office desktop, and a PC in some other institute. You know that you will repeatedly be running jobs that will only last about eight hours, during which time some data management will also take place. You should ask the MyProxy server for a Credential Lifetime of a week (168 hours), and for a Proxy Lifetime of 8 hours. This means that during that week you will be able to retrieve the proxy for as many times as you want, but each time its validity for Grid usage will be restricted to only eight hours from the time of retrieval.

To create a proxy from your original certificate, you must remember the Private Key Password (also known as Passphrase) you chose when you requested your original certificate to your Registration Authority (or to your Grid system administrator). At that time you should have been given the certificate file (usually usercert.pem) and the private key file (usually userkey.pem), both of which you should have at hand.

2.2 MyProxyManager tool and the Proxy Retrieval Page

MyProxyManager is a general purpose tool used to work with any MyProxy server. As such it provides many options for allowing full interaction with any chosen MyProxy server. In the special case of EGRID, some of those options must be set to the values that will be provided later on. A step-by-step explanation of the procedures involved in creating and uploading a proxy to a MyProxy server, as well as subsequent retrieving of the proxy for portal work, follows.

Step 1: Click on the Certificates tab: the Certificate Manager pane appears as shown in figure 3.1. Any proxy previously fed to the portal will be listed there.

Figure 2.1 – Certificate Manager pane

Step 2: To generate a proxy and upload it to a MyProxy server, click the Upload button.

The Upload button will launch a Java Web Start application: you may get a security warning as shown figure 2.2. One warning will be about code signed by Massimo Sponza, and another one will be about code signed by the "Legion of the Bouncy Castle": accept both of them. After a few seconds the MyProxymanager Tool dialog box will appear as shown figure 2.3. you may get a security warning as shown figure 2.2. One warning will be about code signed by Massimo Sponza, and another one will be about code signed by the "Legion of the Bouncy Castle": accept both of them. After a few seconds the MyProxymanager Tool dialog box will appear as shown figure 2.3.

Note: if your browser is not configured to run it, it will ask you to select an application for opening the MyProxymanagerTool.jnlp file. Please refer to Appendix X for further information.

Note: You can verify the digitally signed files by clicking on the Details button.

The MyProxyManager tool has fields for all the information described in the previous section; following is a concise list of those fields together with the values for EGRID:

Figure 2.2 – Starting signed applications

Figure 2.3 – MyProxymanager Tool dialog box

- Upload: upload user credentials to the MyProxy server;
- Delete: remove user credentials from the MyProxy server;
- Proxy Info: provides information about an already uploaded proxy;
- Hostname: the host name of the MyProxy server. For EGRID: myproxy.cnaf.infn.it;
- Port: the port name that is used to communicate with the MyProxy server. For EGRID: 7512;
- CA Certificate Path: the path to the CA certificate that certified the MyProxy server being used. It can be edited directly, or you can choose a file by clicking on the Browse button. Ask your grid administrator how to obtain this file. For EGRID: you can get it from <https://security.fi.infn.it/CA/mgt/getCA.php>.
- Login: specifies the MyProxy account under which the credentials should be store;
- Password: the password that the portal will use to download the proxy certificate from the MyProxy server. This password should have 6 or more characters;
- Credential Name: the credential names. This field is inactive by default: it can be enabled from the Options menu. For EGRID: leave it blank;
- Credential Lifetime: the time to keep the user proxy in the MyProxy server. This is given in hours and by default it is set to 7 days;
- Proxy Lifetime: the life time of a downloaded credential. This is the maximum lifetime of the proxy credential used by the portal;
- PrivateKey Password: the password to access your encrypted private key;
- Certificate File: the path to the user's usercert.pem file. It can be edited directly, or you can choose a file by clicking on the Browse button;
- Private Key File: the path to the user's userkey.pem file. It can be edited directly, or you can choose a file by clicking on the Browse button;

Step 3: Double check your inputs and click the Upload Proxy button.

If it is successful a message box similar to that shown figure 2.4 will appear. Now your proxy certificate is in

the MyProxy Server.

Uploaded proxy information (figure 2.5) can be found by clicking on the Proxy Info button: for exaple you can see your proxy will remain valid for the amount of time you specified in the Credential Lifetime field.

If the uploaded proxy is no longer needed, you can remove it by clicking on the Delete MyProxy button.

Figure 2.5 – Retrieved proxy information

Step 4: For running job submissions and for managing files in grid storage, the proxy certificate must be retrived from the MyProxy Server and fed to the portal. To do this, close the MyProxyManager Tool application and go back to the Certificate Manager pane: click on the Download button.

Step 5: The Download from MyProxy server page appears as shown in figure 2.6.

Figure 2.6 – Downloading proxy certificates

–hostname: the host name of the MyProxy server

–port: please type in 7512

–login: specifies the MyProxy account under which the credentials were stored

–password: the password to access the MyProxy account

–description: can be left empty

–lifetime: for your convenience, it allows you to specify a Proxy Lifetime that is shorter than that specified during upload to the MyProxy server.

Click on Download to retrieve the proxy and feed it to the portal.

Step 6 : Upon successfully retrieving the credentials the page in figure 2.7 is dispalyed:

Figure 2.7 – Assigning a proxy certificate to a Grid

You can now choose in which grid your proxy will be used: select it from the dropdown list and press the OK button. You may select it at later time and also assign it to different grids: just press Cancel to continue.

Note: For EGRID, LCG_2_BROKER suffix must be present in the name.

3. Data management

The data management tab allows to copy local files into grid storage, and viceversa. Moreover it is possible to move files between grid Storage Elements (SE)

Note: currently the infrastructure on top of which the portal's data management is based, is still experimental. In particular you cannot move or copy files between different SEs as long as they have the same path.

Figure 3.1 – Data Management Page

As shown in figure 3.1 there are two navigational panes. For uploading and downloading files into and out of grid storage, only one pane is used. For transfers between different SEs both panes must be used.

To use a navigational pane you must first select Portal resource and then press any of the Go buttons as shown in figure 3.2;

Figure 3.2 – First select resource

it is now possible to browse grid files in three different ways:

- first by selecting the elfi directory from the selection list and then pressing the corresponding Go button. By repeatedly selecting a displayed directory and then pressing the corresponding Go button you can reach any desired location; see figure 3.3.
- second, you can use the drop-down list to go straight to any directory in the current path, after pressing the corresponding Go button
- third, you can type in the path and press the corresponding Go button; note that the path must start with /elfi;

Figure 3.3

Figure 3.4

The content of the elfi directory is structured as shown in figure 3.4:

- the Logical Filenames Catalog (LFC): it shows all files in grid storage regardless of the SEs that physically host them
- direct access to the SEs: they show only the files that physically reside in them.

Note: you are encouraged to manage grid files exclusively through the LFC directory: the system will automatically choose proper SEs. If you are then curious to know in which SE the files ended up, you can always list the SEs. For more detailed information on the grid file model employed, please refer to <http://www.egrid.it/sw/elfi>

For EGRID, the path to the user's home directory is /elfi/LFC/grid/egrid/utenti/username, where with username we refer to the id specified during registration to the VO (figure 3.5). Notice that for job submission, explained in chapter five, you must omit the /elfi/LFC prefix when specifying the grid files used in the job. This is due to the fact that the grid file management module in the portal, is slightly different from that used in job submission.

Figure 3.5

In a directory in which the user has read and write permissions it is possible to upload or download a file. It must be clear you that such up/downloading is taking place in the Grid, and not in the machine hosting the portal. Also keep in mind that both operations can only work on a file at a time, and cannot operate on directories.

Uploading a file

The upload button invokes the page shown in figure 3.6. By pressing the **Browse..** button is possible, through a graphical interface, to select a local file to be uploaded. Press upload to transfer the file into Grid storage.

Figure 3.6

Downloadig a file

The download button invokes the page shown in figure 3.7. To carry out the download it is necessary to first select a file. The file to be downloaded is presented as a link in a page. Clicking on the link will save the file in your local computer.

Figure 3.7 – Follow the link to downlad the file

If the file got opened in a browser window, you should instead right click on the link and choose Save Target Link As... as shown in figure 3.8.

fig. 3.8 – Click with the right mouse button to force the downloading

Setting ACL to a file

Comming soon

4. Job submission

The *Workflow Tab* is used to submit jobs to the grid; in fact the portal's view of job submission differs from that of the pure *gLite middleware* upon which it is based. For the portal all jobs get executed within the context of *workflows* that induce an ordering in execution according to the input and output data required; if a job needs an input file that will be produced by another job, then it will wait for that job to finish first. Compare it to *gLite*, which until very recently had a simple view of discrete and non-related jobs, and so no concept of workflow.

The portal sees workflows as graphs, where the *nodes* represent single jobs, special *node-attributes* represent the produced or required files, and where *arcs* connecting different jobs represent the flow of those files and hence an implicit job execution ordering. Therefore in case of only *one* job submission without any dependency on data produced by other jobs, the workflow is reduced to a graph with a single node. Moreover, the portal restricts the possible workflows to those that are acyclic, that is those that do not loop back.

The portal provides the tools to graphically manipulate workflows: to define them and to run them. Since the portal is designed to work with different middleware and not only *gLite*, the graphical formalism adopted to represent workflows is independent of the actual grid. Yet this generality limits what can be done through the portal on *gLite* grids, by not leveraging the intended use and organisation of *gLite* based grids, of which *EGRID* is a VO. These limits will be highlighted together with some strategies to still exact the most out of *gLite* grids through the portal.

The next section will describe the portal tools applied to the development of a simple workflow consisting of only one job. Section 4.1 introduces the concepts involved in job submission in a *gLite* grid; if you already know about Job submission and just wish to know how to do it through the portal, please skip it to section 4.2; finally section 4.3 serves a double purpose by first describing the limits of the portal workflow regarding *gLite* production grids, and by then also giving some worked examples on how to make the most out of it.

4.1 Job submission in gLite grids

In this short section the concepts regarding job submission in *gLite* grids will be presented. This is not a full description of the middleware, but only of those parts that are used by the portal. Further restrictions also apply because the current installation of the portal is specifically geared towards the *EGRID* production

facility: some of the middleware options that are also available in the portal are not used, and therefore they will not be explained. This is due to the fact that EGRID uses StoRM secure SEs, ELFI and special clients, which make certain portal/gLite options in Job submission unnecessary. For a complete explanation of Job Submission, please refer to the official EGRID general manual.

For gLite a *Job Submission* implies some precise actions. An executable file must be transferred to a WN where it will be run. The transferred programme may have command line parameters, which must be specified when executed. Any required data file must also get transferred to the WN, and similarly any produced output file must also be transferred back from the WN, unless it is a temporary file for which there is no user interest. gLite grids are not guaranteed to be uniform in terms of both hardware and software, so not all WNs are the same: jobs may need to specify strict WN constraints as for example Xeon class processors or the availability of a precise version of Java. The set of these characteristics constitutes a description of the job that you want to run in the grid; prior to launching the job, the description must be filled in a text file, using a precise formalism known as *Job Description Language*; the text file so written is called a *JDL file*.

The JDL file is then given to a special client software, installed in the user machine, that contacts specific grid services. The end result is that the executable programme together with the input files end up in WNs that satisfy the constraints specified. The status of the job can be monitored by the user, and any output can be conveniently retrieved upon successful completion. The transfer of the executable, of the input files and of the output files around the grid, occurs through a mechanism called *Job Sandbox*. This mechanism has some limits on the amount of data that can be transferred, and is usually advised only for jobs that require and produce little data in the order of Megabytes.

To overcome the limits of the Job Sandbox, a different strategy is advised in gLite grids. The executable programme is usually already installed in the grid, where it gets identified by a specific name; generally, the user asks the grid administrator to install the software, as well as for the grid name that was assigned to it; software installed in the grid is referred to as *Experiment Software*. Input files are separately uploaded by the user in grid Storage Elements; likewise for output files, which get transferred directly from the WNs to StorageElements by the user. The user then writes a purpose made script in a language that is natively supported by the WNs: it's this script that is submitted as a job! The trick is that the script contains the commands that will transfer files from the SEs, execute the programme, and send back files to the SEs. Therefore the Sandbox is used only to transfer the script: the JDL will contain only the name of the script, and possibly a constraint to choose only WNs with the right Experiment Software.

4.2 Job submission through the portal

All job submissions occur through Workflows that users define through the graphical tools supplied by the portal; the workflows are then saved in in the user's private portal area for which the portal supplies the necessary management tools; finally, the workflows are executed again with the tools supplied by the portal. The following sections describe each of these operations.

4.2.1 Personal portal area

Upon clicking on the *Workflow tab* fig. 1 is displayed. It presents a list of *recently* run and defined workflows: pay attention to the amount of space occupied by each, and to the total space in your *personal portal area*. All of your workflows are saved within the portal, so each user has a quota that limits the amount of used space. That space is also used to save results and job input data, if the respective files were specified to be managed through the sand box.

From this screen you can run a saved workflow by clicking on the *submit* button; view the results of a workflow by clicking on the *Details* button; edit a workflow through the *Attach* button; and Delete a workflow either from the list of most recently run/edited ones, or permanently from your personal portal area, by clicking on the *Delete* button. If you want to create a new workflow, click on the *Workflow editor* button.

Before describing each of these functionalities in detail, notice the *Storage* and the *Upload* links. If you click on the *Storage* link, fig. 2 is displayed. It presents you with the content of your special portal area: this list, therefore, is not limited to the most recently run and edited workflows. From this screen you can delete the files managed through the sandbox, when clicking on the *Set Init* button, thereby freeing room in your personal area but leaving the workflow otherwise intact. You can also download a *zip* file with the exact structure of files and directories used for workflow submission: this may be useful during workflow debugging sessions because there are log files which are not shown through the portal, and because in this way you can also share your workflows with other users who can then upload them. In fact, by clicking on the *Upload* link you are presented with fig. 3, where you can choose to upload into your personal portal area a *zip* file found either in your local machine, or in a pre-configured FTP repository. For EGRID, there is no FTP repository configured.

Notice that the *zip* file will be saved with a name ending with the extension *.tgz*, which is misleading because it is not the common Unix format *tar* followed by *gzip*: it is just a classic *zip* file.

4.2.2 Creating a workflow

To create a new workflow you press the *Workflow editor* button, which starts a Java WebStart application and presents you with the screens shown in fig. 4 and fig. 5: you should accept and trust the application. Let's build a workflow that represents a single job submission: press the *Node* button and double click on the graphical element that appears, so fig. 6 gets displayed.

You can specify a name for the graphical element in the *Name* field, whether the programme to be run is MPI or not with the toggle button *Job Type* (and in that case you must specify the number of processors you want, in the *Process Number* field), the name of the programme (*Job executable*) that must be uploaded from your local machine, and any command line parameters that your programme requires (*Attributes*). For EGRID you must then choose *egrid_LCG2_BROKER* in the drop down list *Grid*, and leave the *Resource* drop down list as is. Do **not** choose a grid without the *_LCG2_BROKER* suffix, or else your job won't be submitted correctly to EGRID. Pressing *OK* saves your job definition and lets you return to the workflow graphical pane.

The next step is to define the files that will be needed by your programme and that therefore need to be transferred to/from the WN executing it. For this purpose the editor defines *Port* graphical elements, which get applied to the currently selected node by pressing the respective button as shown in Fig 8. By double clicking on the *Port* graphical element the screen in Fig. 9 is shown.

The *Port name* field allows you to define a name for the graphical element. The *Type* toggle button allows you to specify the file as one that will be used by your programme and that therefore needs to be transferred **to** the WN (*In* option), or as a file that will be created by your programme and that needs to be transferred **back** from the WN (*Out* option). The *File type* toggle button let's you specify if the file transfer will be done through the **SandBox** (*Local* option), or directly involve SEs (*Remote* option). The *File* and *Internal name* fields are very important: the first one allows you to specify the **actual** name of the file present locally in your local machine or in the SE; *Internal name* on the other hand is the name of the file **in the WN**. So your programme running in the WN will work with files identified by their *Internal name*, whereas outside of the WN they will be identified by the name given in the *File* field.

Notice that for EGRID, files in StoRM secure storage SE can only be accessed from WNs which have special software to do so. Although this special software has been installed around the grid so you can use all of the available raw computing power, it is important to add extra parameters to the job submission. This is further explained in following sections.

Now that the definition is complete, from the *Menu bar* choose **File**→**Save**; you will be prompted for a name to give to the workflow, any error in the workflow definition will be noted, and any local file that you specified will be transferred first to the portal in your personal area and from there to the WN through the

SandBox during Job submission. See Fig. 9, Fig. 10 and Fig. 11. The Workflow is then saved on the portal in your personal portal area. You can now close the *Workflow Editor*; and if you press the *Refresh* button as shown in Fig. 1, you will see listed the Workflow just created.

4.2.3 Launching a workflow

Once you return to the *Workflow tab* as in Fig. 1, you can press *Attach* to open the saved workflow for *editing* with the same tool used during its definition; or you can press *Submit* to launch the workflow and so submit the jobs to the grid. Remember to have a valid credential active as explained in previous chapters, or else the workflow will fail and a proper error message gets displayed as shown in Fig. 12. For monitoring workflow execution and to retrieve any file you specified as Sandbox output, click on *Details* button and a screen similar to Fig. 13 is displayed.

That's essentially all there is to know about a single job submission from the portal. In section 4.3 you can see applications of the concepts involved, into worked examples.

4.2.4 If the programme is already installed in the grid

Notice that only programmes that you have currently in your machine can be submitted as jobs to the grid. If the programme has already been installed around the grid, that is it's an *experiment software*, then there are two steps to follow:

- ◆ first, you must write a specific script and submit that;
- ◆ second, you must specify a job constraint.

The script must be written in any language supported by the WNs that will be running your jobs. Since all WNs of gLite grids and in particular for EGRID have Linux installed, this means that you could write *shell* scripts. Within this script, you specify the name of the programme to execute, together with any parameters it needs. You therefore use the portal to submit this script as *Job executable* and you leave the *Attributes* field empty (as you have everything setup inside the script).

The job constraint is needed to tell that only WNs with that programme must be considered valid for job submission. You do this by clicking on the *JDL Editor* button, which brings up the window shown in fig. 7. There, you go to the *Rank and Requirements* tab, and then you specify the following string:

```
Member( "programme_name" , other.GlueHostApplicationSoftwareRunTimeEnvironment )
```

Notice that *programme_name* stands for the name used to publish your programme in the grid: ask your grid administrator to find out the names of any of your software that has been installed in the grid. Notice the double quotes which you *must* use.

For EGRID it is useful to end up in a WNs that have ELFI installed, or on WNs that at least have the special EGRID clients, any one of which must be used to access files in EGRID's StoRM secure grid storage:

```
Member( "EGRID-elfi" , other.GlueHostApplicationSoftwareRunTimeEnvironment )
Member( "EGRID-storm-client" , other.GlueHostApplicationSoftwareRunTimeEnvironment )
```

4.3 gLite grids and the portal workflow

gLite production grids were envisioned to be made with dozens or perhaps hundreds of CEs, while behind each CE there would be hundreds or thousands WNs: at least an order of magnitude more WNs than CEs. Each job being executed on a WN was thought to last at least in the order of hours, so that the minutes it takes the RB to submit jobs, as well as the minutes it takes the Information System to update its view of the grid, do

not impose a heavy toll on the user given the time scale involved. Although there is support for MPI parallel jobs, the kind of problems for which gLite production grids were intended and designed, are more readily attacked by submission of very large numbers of the same programme but on different chunks of data, with the programmes that do not communicate among themselves.

The portal workflow, on the other hand, allows for submission of usual programmes and also of special MPI parallel ones: a node attribute in the graph tells the portal if it is an MPI job or not. For usual programmes, each node on the graph is *one* WN. When there is the need to launch hundreds of usual programmes, there is no alternative to drawing a workflow with hundreds of nodes. It is clearly impractical: the portal's current graphical formalism is not expressive enough to represent the kind of jobs to be run on production grids for which the gLite middleware is designed.

For MPI jobs, the portal workflow regains its usability since a node representing such jobs must specify the number of processors required. So the node ends up representing a *group* of WNs behind the CE, and a workflow with a handful of MPI nodes may actually have hundreds of WNs involved. Still, unlike usual programmes MPI ones must be purpose written so the user must wire in all the necessary communication logic between the parallel processes; gLite grids support for MPI is patchy; and more important still, if you submit an MPI job which requires 64 CPUs it will run only if all 64 CPUs are available at once, or else it will fail right away, unlike for usual programmes for which resubmission will be attempted periodically.

4.3.1 Example of a single job

4.3.2 Previous example rewritten as a Workflow of 5 jobs

4.3.3 Previous example rewritten as a Parametric job

4.3.4 Previous example rewritten as an MPI job

5. Information system

Il sistema informativo permette di controllare le risorse presenti in griglia e il loro stato di occupazione. Di default presenta tutti i siti, anche quelli che non appartengono alla nostra *Virtual Organisation*. Per potere visualizzare soltanto i dati che ci interessano bisogna selezionare la nostra VO tramite la combo box *Select VO* e premere *View*. E' possibile selezionare anche griglie differenti se il portale e' impostato per accedere a diverse griglie. Com'e' visibile in figura 5.1.

Figure 5.1

Le informazioni presentate sono raggruppate per sito. I dati visualizzati sono il numero di CPU del sito, quelle disponibili, i job in esecuzione, quelli in attesa, la capacita totale degli SE e la percentuale di spazio libero. Vengono inoltre mostrate le percentuali di relative alle CPU usate *Usage* e al numero di job in attesa rispetto al totale degli stessi, *Load*.

Ogni sito puo' essere costituito da piu' SE e CE. Nel caso ci siano piu' code sullo stesso CE queste vengono mostrate come CE distinti. Clickando sul nome del sito si accede al dettagli delle risorse del sito, come mostrato in figura 5.2.

Figure 5.2

Figure 5.3

In figura 5.2 e' visibile un sito con due CE distinti, mentre in figura 5.3 si vede un sito con un unico CE con tre code. Le statistiche in quesato caso riguardano i singoli CE ed SE.