



ID: INFNGRID20030926-1230

Version: v1.0.0

Date: September 29, 2003

Job Submission Tutorial

Andrea Caltroni, INFN-PD <caltroni@pd.infn.it>

Andrea Ferraro, INFN-BO <ferraro@bo.infn.it>

Enrico Ferro, INFN-PD <ferro@pd.infn.it>

Abstract: *Job submission is handled as a large batch system with commands to submit a job, check its status, and retrieve any output. Here you will learn how to submit a simple job on the Grid.*

Contents

1	Job Submission Commands	3
2	The Job Description File	3
2.1	Example 1 : HelloWorld.jdl	4
2.2	Command Sequence.....	4
2.3	Other Commands	7
3	Long Lived Jobs	7
4	Example 2 : Hello from Script	7
5	Example 3 : Specifying Job Requirements	8
6	Example 4 : Ranking Resources	9
	References	11

1 Job Submission Commands

Here you can find a brief description of the relevant commands to manage a job submission to the grid.

```
edg-job-submit <jdl_file>
edg-job-status <job_ID>
edg-job-get-output <job_ID>
edg-job-list-match <jdl_file>
edg-job-cancel <job_ID>
```

edg-job-submit This is the command to submit a job to the grid. The command requires as input a JDL (Job Description File) and returns a job ID (`edg_jobId`).

edg-job-status This command prints the status of a job previously submitted using `edg-job-submit`. The job status request is sent to the LB (Logging and Bookkeeping service) that provides the requested information.

edg-job-get-output The `edg-job-get-output` command can be used to retrieve the output files of a job that has been submitted through the `edg-job-submit` command with a job description file including the *OutputSandbox* attribute. After the submission, when the job has terminated its execution, the user can download the files generated by the job and temporarily stored on the RB (Resource Broker) machine as specified by the *OutputSandbox* attribute, issuing the `edg-job-get-output` with as input the job ID returned by the `edg-job-submit`.

edg-job-list-match Displays the list of identifiers of the resources on which the user is authorized and satisfying the job requirements included in the JDF.

edg-job-cancel This command cancels a job previously submitted using `edg-job-submit`. Before cancellation, it prompts the user for confirmation. The cancel request is sent to the Network Server that forwards it to the WM that fulfils it.

The Job Identifiers produced by the workload management software are of the form:

```
https://edt003.cnaf.infn.it:9000/NyIYrqE_a8igk4f0CLXNKA
```

The other commands take job IDs as input and perform the corresponding action on the listed jobs. The `edg-job-get-output` pulls back the output from the job from the resource broker machine where it is cached; the output can only be retrieved when a job has reached the *OutputReady* status.

2 The Job Description File

The key to the job submission and resource matching process is the job description file. This file describes the necessary inputs, generated outputs, and resource requirements of a job using the JDL (Job Description Language).

A typical example of a job description file:

```
Executable = "/bin/echo";
Arguments = "Hello World";
StdOutput = "message.txt";
```

```
StdError = "stderr";
OutputSandbox = {"message.txt", "stderr"};
Requirements = other.LRMSType=="PBS"
Rank          = other.FreeCPUs;
```

shows that a command is passed as input to a job, which then stores its output in a file which will be eventually transported back to the user (with a `edg-job-get-output`).

Important note: The input and output sandboxes are intended for relatively small files (few megabytes) like scripts, standard input, and standard output streams.

If you are using large input files or generating large output files, you should instead directly read from or write to a storage element. Abuse of the input and output sandboxes can fill the storage on the ResourceBroker and cause the broker to crash.

The parameters `Requirements` and `Rank` control the resource matching for the job. The expression given for the requirements specifies the constraints necessary for a job to run. In this case, a site running PBS is required. The job will only be submitted to resources which satisfy this condition. If more than one resource matches, then the rank is used to determine which is the most desirable resource and hence the one to which the job is submitted. (Higher values are more desirable.) Both of these can be arbitrary expressions which use the fields published by the resources in the MDS system. JDL is based on the Condor ClassAds library. More information about the supported functions and the syntax of these expressions can be found in the ClassAds documentation.

ClassAds are extensible and place no restrictions on the parameter names. A side-effect of this is that misspelled parameter names are still syntactically valid, but will not act as expected. Be extremely careful spelling the JDL parameter names.

It is often useful to check the results of the resource matching without submitting a job. For this, one can use the `edg-job-list-match` command. Given the JDL file it will return a ranked list of matching resources. The highest-ranked resource will appear first.

2.1 Example 1 : HelloWorld.jdl

Let's examine the Hello World task in more detail.
JDL file HelloWorld.jdl:

```
Executable = "/bin/echo";
Arguments = "Hello World";
StdOutput = "message.txt";
StdError = "stderr";
OutputSandbox = {"message.txt", "stderr"};
```

`stdout` (like its siblings) can be shortened in `std.out` (`stderr --> std.err`; `stdin --> std.in`). Note that the complete path of the command is given and that the standard output and standard error are specified in the output sandbox.

2.2 Command Sequence

Once the JDL file is ready, we can issue a `grid-proxy-init` to get a valid proxy certificate. A list of available CEids is the one returned from the `edg-job-list-match` command.

```
> grid-proxy-init
> edg-job-list-match HelloWorld.jdl
```

Connecting to host edt003.cnaf.infn.it, port 7772

```
*****
```

COMPUTING ELEMENT IDS LIST

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

CEId

```
grid20.bo.ingv.it:2119/jobmanager-pbs-infinite
grid20.bo.ingv.it:2119/jobmanager-pbs-long
grid20.bo.ingv.it:2119/jobmanager-pbs-short
gridba2.ba.infn.it:2119/jobmanager-lcgpbs-infinite
gridba2.ba.infn.it:2119/jobmanager-lcgpbs-long
gridba2.ba.infn.it:2119/jobmanager-lcgpbs-short
gridit001.pd.infn.it:2119/jobmanager-pbs-infinite
gridit001.pd.infn.it:2119/jobmanager-pbs-long
gridit001.pd.infn.it:2119/jobmanager-pbs-short
```

```
*****
```

```
> edg-job-submit HelloWorld.jdl
```

Connecting to host edt003.cnaf.infn.it, port 7772

Logging to host edt003.cnaf.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://edt003.cnaf.infn.it:9000/NyIYrqE_a8igk4f0CLXNKA

```
*****
```

Note that this returns the job identifier associated with this job (the string starting with https). The job Id is the unique GRID Job Identifier, assigned from the WMS (Workload Management System) to every job in order to be able to identify it in clear and unique way all over the GRID system scope.

```
> edg-job-status https://edt003.cnaf.infn.it:9000/NyIYrqE_a8igk4f0CLXNKA
```

```
*****
```

BOOKKEEPING INFORMATION:

```
Printing status info for the Job :
https://edt003.cnaf.infn.it:9000/NyIYrqE_a8igk4f0CLXNKA
Current Status:    Done (Success)
Exit code:        0
```

```

Status Reason:      Job terminated successfully
Destination:       gridit001.pd.infn.it:2119/jobmanager-lcgpbs-infinite
reached on:        Mon Sep 22 09:37:13 2003
*****

```

When the status was requested, the job had finished and the output had been pushed back to the resource broker. States seen in the normal processing of jobs are: *Accepted*, *Waiting*, *Running*, *Done*, and *OutputReady*. Abnormal execution usually ends with an *Aborted* status.

```
> edg-job-get-output https://edt003.cnaf.infn.it:9000/NyIYrqE_a8igk4f0CLXNKA
```

```
Retrieving files from host edt003.cnaf.infn.it
```

```

*****
                        JOB GET OUTPUT OUTCOME

Output sandbox files for the job:
- https://edt003.cnaf.infn.it:9000/NyIYrqE_a8igk4f0CLXNKA
have been successfully retrieved and stored in the directory:
/tmp/jobOutput/NyIYrqE_a8igk4f0CLXNKA

*****

```

The contents of the JDL file transformed by many different programs during the submission process. A side effect of this is that special characters in the arguments attribute must be preceded by triple backslash, e.g.:

```

Executable = "/usr/bin/tail";
Arguments = "-f file1\\\&file2";

```

and quotes in the arguments must be escaped with the backslash, character, e.g.:

```

Executable = "/bin/grep";
Arguments = "-i \"my name\" *.txt";

```

to be on the safe side or better create a script as in the next example.

Handling the job identifiers directly quickly becomes tedious. To avoid the need to handle directly the identifiers, the `edg-job-submit` will append the job ID to a named file when using the `-o` option. On the other side, the job submission commands which take job identifiers as an argument accept the `-i` which allows the job identifier to be read from a file.

Note that the job management system does not limit the rate of jobs submitted either by a single user or in total. The broker does not respond well to high rates and has a limit of 512 concurrently running jobs. Continuous serial submissions are fine for short periods of time keeping in mind that the 512 concurrent jobs are for all users of the broker.

2.3 Other Commands

To force the execution on a given CE, you can execute:

```
> edg-job-submit -resource CEid HelloWorld.jdl
> edg-job-status JobId
> edg-job-get-output JobId
```

You can try the `edg-job-get-logging-info` to get some informations for the job:

```
> edg-job-get-logging-info JobId
```

or cancel a given job with :

```
> edg-job-cancel JobId
```

3 Long Lived Jobs

It is possible that long jobs may outlive the validity of the initial proxy; if so and the proxy is not renewed, the job will die prematurely. To avoid this the workload management software allows the proxy to be renewed automatically if your credentials are managed by a proxy server.

To use the automatic proxy renewal mechanism, first register a proxy with the MyProxy server using the command

```
myproxy-init -s <server> -t <hours> -d -n
```

where `server` is the server address, `hours` is the number of hours the proxy should be valid on the server.

As this proxy is only copied to the server, you will need to create a local short-lived proxy using `grid-proxy-init` to do the job submissions. The resource broker will retrieve renewed proxies from the myproxy server for jobs which need them.

Information about your stored proxy can be obtained via the command

```
myproxy-info -s <server> -d
```

and the proxy can be removed with

```
myproxy-destroy -s <server> -d.
```

Once the proxy is removed from the server, running jobs will no longer receive renewed credentials.

4 Example 2 : Hello from Script

The next example simply sends a small script with the job, executes it and returns the results. Create an executable file called `HelloScript.sh` which contains the following:

```
#!/bin/sh
/bin/echo "Hello From Script"
/bin/ls 9485968.txt
```

This will simply echo the given phrase on the standard output and put an error into the standard error (unless you have a file named `9485968.txt` on the machine on which your job runs).

The appropriate JDL file for this job is the following.

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

If the script is not in the current directory, then you must give the full path in the input sandbox line.

If the job executed correctly, then the standard output and standard error should contain the following

```
Hello From Script
```

and

```
/bin/ls: 9485968.txt: No such file or directory
```

respectively.

The executable named in the JDL will automatically be set with executable permissions. Other executable files may need to have the permissions set explicitly.

5 Example 3 : Specifying Job Requirements

By specifying job requirements, the user can steer the job to sites which have the resources necessary to run the job correctly. Incompletely specifying the requirements may cause the job to fail, wasting both the resources and the user's time.

The requirements are specified with a `Requirements` attribute in the JDL description of the job. This value of this attribute is a boolean expression which specifies the necessary constraints. Nearly the full set of C operators and syntax are supported.

The values (or variables) which can be used in the requirements expression can be found by looking at the `ComputingElement` attributes in the information system (see Appendix E).

```
ldapsearch -x \
-H ldap://edt001.cnaf.infn.it:2135 \
-b 'mds-vo-name=local,o=grid' \
'(objectclass=computingelement)'
```

Attributes which are typically used are `Architecture`, `OpSys`, `RunTimeEnvironment`, `MaxCPUTime`, `MaxWallClockTime` and `FreeCPUs`. Most of the attributes are self-explanatory.

To express that a job requires at least 25 minutes of CPU time and 100 minutes of real time, the expression:

```
Requirements = other.MaxCPUtime>=1500 && other.MaxWallClockTime>=6000;
```

would limit the matching to viable sites. The times are given in seconds. Note that the attribute names are prefixed with `other.`; this is a remnant of the `ClassAds` syntax on which `JDL` is based. Note also that the values are not quoted. Using quotes around a numeric value will result in a string comparison which will produce an erroneous match (or none at all).

The `RunTimeEnvironment` is usually used to describe application software packages which are installed on a site. For example,

```
Requirements = Member(other.RunTimeEnvironment,"ALICE-3.07.01");
```

will choose a site with the `ALICE-3.07.01` tag defined. The `RunTimeEnvironment` is a multi-valued attribute and evaluates to a list. The `Member` function returns true if the given value is in the list.

Occasionally, one may wish to exclude or include a site manually. Forcing a job to a site can be accomplished with the `-resource` option of the `dg-job-submit` command. However, this entirely bypasses the matchmaking process and will not produce a `.BrokerInfo` file (see the `Workload Management Documentation` for information on the `BrokerInfo` file) with the matchmaking results. Instead one can use a clause like:

```
Requirements = other.CEId=="ccgridli03.in2p3.fr:2119/jobmanager-bqs-A";
```

to do the same thing. More interestingly one can select or exclude a site:

```
Requirements = RegExp(".*nikhef.*",other.CEId);  
Requirements = (!(RegExp(".*nikhef.*",other.CEId)));
```

which cannot be accomplished with the `-resource` option. Note that the `JDL` is very picky about the logical not syntax. Many sites also define a `RunTimeEnvironment` variable which identifies the site.

In the UI configuration file (`/opt/edg/etc/UI_ConfigENV.cfg`) there is a `requirements` clause which is added to all `JDL` files by default. By default this is `other.Active`. Users may create a UI configuration file of their own to specify habitual requirements (or to choose an alternate resource broker, etc.). To use a custom UI configuration file set the `edg_wl_ui_config_path` variable to the full path name, or specify the `-c` option when submitting a job with the `edg-job-submit` command.

6 Example 4 : Ranking Resources

If more than one resource matches the specified requirements, then the highest-ranked resource will be used. If the `Rank` attribute is not specified in the user's `JDL` description, then

```
Rank = -other.EstimatedTraversalTime;
```

will be used by default (also specified in the UI configuration file). The traversal time is the expected time in seconds that a job will take to begin executing at the site.

This ranking is not always ideal, and the user may wish to choose some other criteria for the ranking. For example,

Rank = other.FreeCPUs;

will choose the site with the largest number of free CPUs. The rule to remember is that the larger the rank, the more desirable the resource is. If more than one site has exactly the same rank, then the one which is used is chosen randomly.

References

- [1] INFN-GRID/EDG User Tutorial: Job Submission. Exercise 1
<http://hep-proj-grid-tutorials.web.cern.ch/hep-proj-grid-tutorials/JS-Exercise1.html>
- [2] EDG-Users-Guide.pdf
<http://marianne.in2p3.fr/datagrid/documentation/EDG-Users-Guide.pdf>
- [3] Man pages