Table of Contents

CREAM User's Guide for EMI-2

CREAM Command Line Interface Guide

This section briefly explains the sequence of operations to be performed by a user to submit and then manage jobs on CREAM based CEs, referring to the C++ Command Line Interface (CLI).

Before starting: get your user proxy

Before using any of the CREAM client commands, it is necessary to have a valid proxy credential available on the client machine. You can create it using the `voms-proxy-init` command. If you already have a valid proxy available on your machine just make the `X509_USER_PROXY` environment variable point to it.

In order to get a proxy certificate issued by VOMS, you should have in the directory `/etc/vomses` the proper VOMS file containing a line as follows:

```
"EGEE" "kuiken.nikhef.nl" "15001" "/O=dutchgrid/O=hosts/OU=nikhef.nl/CN=kuiken.nikhef.nl" "EGEE"
```

or the corresponding line for your VO. You also need to install the VO related `.lsc` files in the `/etc/grid-security/vomsdir/<VO>` directory. In a standard EMI UI installation, these settings should be already there.

Make moreover sure you have in the directory `$HOME/.globus` your certificate/key pair, i.e. the following files:

```
usercert.pem
userkey.pem
```

Note that file permissions are important: the two files must have respectively 0600 and 0400 permissions.

Then you can issue the VOMS client command (you will be prompted for the pass-phrase):

```
$ voms-proxy-init -voms dteam
Enter GRID pass phrase:
Your identity: /C=IT/O=INFN/OU=Personal Certificate/L=Padova/CN=Massimo Sgaravatto
Creating temporary proxy ..................................................................
Contacting  voms2.hellasgrid.gr:15004 [/C=GR/O=HellasGrid/OU=hellasgrid.gr/CN=voms2.hellasgrid.gr
Creating proxy ............................. Done

Your proxy is valid until Sat Apr 30 05:05:49 2011

$ voms-proxy-info -all
subject   : /C=IT/O=INFN/OU=Personal Certificate/L=Padova/CN=Massimo Sgaravatto/CN=proxy
issuer    : /C=IT/O=INFN/OU=Personal Certificate/L=Padova/CN=Massimo Sgaravatto
identity  : /C=IT/O=INFN/OU=Personal Certificate/L=Padova/CN=Massimo Sgaravatto
type      : proxy
strength  : 1024 bits
path      : /tmp/x509up_u500
timeleft  : 11:59:55
key usage : Digital Signature, Key Encipherment, Data Encipherment
=== VO dteam extension information ===
VO        : dteam
subject   : /C=IT/O=INFN/OU=Personal Certificate/L=Padova/CN=Massimo Sgaravatto
issuer    : /C=GR/O=HellasGrid/OU=hellasgrid.gr/CN=voms2.hellasgrid.gr
attribute : /dteam/Role=NULL/Capability=NULL
attribute : /dteam/italy/Role=NULL/Capability=NULL
attribute : /dteam/italy/INFN-PADOVA/Role=NULL/Capability=NULL
timeleft  : 11:59:55
uri       : voms2.hellasgrid.gr:15004
```

CREAM CLI commands

The most relevant commands to interact with CREAM based CEs are:

- glite-ce-job-submit <jdl_file_1> <jdl_file_2> ... <jdl_file_N>
- glite-ce-delegate-proxy <delegation_Id>
- glite-ce-job-status <job_Id_1> <job_Id_2> ... <job_Id_N>
- glite-ce-job-list <host[:port]>
- glite-ce-job-cancel <job_Id_1> <job_Id_2> ... <job_Id_N>
- glite-ce-job-suspend <job_Id_1> <job_Id_2> ... <job_Id_N>
- glite-ce-job-resume <job_Id_1> <job_Id_2> ... <job_Id_N>
- glite-ce-job-output <job_Id_1> <job_Id_2> ... <job_Id_N>
- glite-ce-job-purge <job_Id_1> <job_Id_2> ... <job_Id_N>
- glite-ce-proxy-renew <delegation_Id_1> <delegation_Id_2> ... <delegation_Id_N>
- glite-ce-service-info <host[:port]>
- glite-ce-get-cemon-url <host[:port]>
- glite-ce-enable-submission <host[:port]>
- glite-ce-disable-submission <host[:port]>
- glite-ce-allowed-submission <host[:port]>
- glite-ce-job-lease <lease_identifier> --endpoint <cream_endpoint> --leaseTime <lease_time>

Man pages are available for all the CREAM client commands. You can also access information about the usage of each command by issuing:

```
$ <command> --help
```

`glite-ce-job-submit` submits N jobs (N must be >=1) to a CREAM based CE. It requires N JDL files as input and returns N CREAM job identifiers.

`glite-ce-delegate-proxy` allows the user to delegate her proxy credential to the CREAM service. This delegated credential can then be used for job submissions.

`glite-ce-job-status` displays information (in particular the states) of N jobs (N must be >=1) previously submitted to CREAM based CEs.

`glite-ce-job-list` lists the identifiers of jobs submitted to a CREAM based CE by the user issuing the command.

`glite-ce-job-cancel` cancels N jobs (N must be >=1) previously submitted to CREAM based CEs.

`glite-ce-job-suspend` suspends the execution of N jobs (N must be >=1) previously submitted to CREAM based CEs.

`glite-ce-job-resume` resumes the execution of N jobs (N must be >=1) which have been previously suspended.

`glite-ce-job-output` retrieves the output sandbox files of N jobs (N must be >=1) previously submitted to CREAM based CEs.

`glite-ce-job-purge` clears N jobs (N must be >=1) from CREAM based CEs. After this operation the purged jobs can't be managed anymore.

`glite-ce-proxy-renew` renews N delegations (N must be >=1), and therefore refreshes the proxy of the jobs submitted to CREAM based CEs using the considered delegations.

`glite-ce-service-info` returns information about the CREAM service (version, status, etc.).

`glite-ce-get-cemon-url` returns the end-point of the CEMon service coupled with the considered CREAM CE.

`glite-ce-enable-submission` (re-)enables job submissions on the specified CREAM CE.

`glite-ce-disable-submission` disables job submissions on the specified CREAM CE.

`glite-ce-allowed-submission` checks if jobs submissions on the specified CREAM CE are allowed or have been disabled.

`glite-ce-job-lease` create a lease identifier in the CREAM server and associate a time duration to it.

All these commands are described in the following sections.

Submitting jobs to CREAM based CEs

To submit jobs to CREAM based CEs, the command `glite-ce-job-submit` must be used. The `glite-ce-job-submit` command requires as input one or more job description files; each file describes the job characteristics and requirements through the JDL (Job Description Language). A typical example of a JDL job description file is:

```
[
Type = "Job";
JobType = "Normal";
Executable = "myexe";
StdInput = "myinput.txt";
StdOutput = "message.txt";
StdError = "error.txt";
InputSandbox = {"/users/seredova/example/myinput.txt",
"/users/seredova/example/myexe"};
OutputSandbox = {"message.txt", "error.txt"};
OutputSandboxBaseDestUri = "gsiftp://se.pd.infn.it/data/seredova";
]
```

Such a JDL would make the `myexe` executable be transferred on the remote CREAM CE and be run taking the `myinput.txt` file (also copied from the client node) as input. The standard streams of the job are redirected to files `message.txt` and `error.txt`, and when job completes its execution they are automatically uploaded on `gsiftp://se.pd.infn.it/data/seredova`.

A detailed description of the available JDL attributes and of the rules for building correct JDL files is provided at http://wiki.italiangrid.org/twiki/bin/view/CREAM/JdlGuide .

The jobs submitted to a CREAM based CE are given the delegated credentials of the user who submitted it. These credentials can then be used when operations requiring security support has to be performed by the job.

There are two possible options to deal with proxy delegation:

- asking the "automatic" delegation of the credentials during the submission operation;
- explicitly delegating credentials, and then asking to rely on these previously delegated credentials on the actual submission operations.

It is highly suggested to rely on this latter mechanism, using the same delegated proxy for multiple job submissions, instead of delegating each time a proxy. Delegating a proxy, in fact, is an operation that can require a non negligible time.

The command `glite-ce-delegate-proxy` is the command to be used to explicitly delegate the user credentials to a CREAM CE.

The following shows an example of job submission, performed explicitly delegating credentials. So first of all the credentials are delegated to a CREAM based CE (whose endpoint is specified with the option `--endpoint` (`-e`):

```
> glite-ce-delegate-proxy -e cream-ce-01.pd.infn.it mydelid
2006-02-26 15:03:37,286 NOTICE - Proxy with delegation id [mydelid] successfully
delegated to endpoint [https://cream-ce-01.pd.infn.it:8443//ce-cream/services/CREAMDelegation]
```

The identifier of the delegation is then specified with the `--delegationId` (`-D`) option in the job submit operation:

```
> glite-ce-job-submit -D mydelid -r cream-ce-01.pd.infn.it:8443/cream-lsf-grid02 myjob1.jdl myjob
```

The option `-r` (`--resource`) has been used to specify the identifier of the CREAM CE where the job has to be submitted to.

`myjob1.jdl` `myjob2.jdl` `myjob3.jdl` are the 3 JDL files describing the jobs to be submitted.

The command returns the CREAM job identifiers associated with these jobs (e.g. `https://cream-ce-01.pd.infn.it:8443/CREAM116j9vgnf`) which identify them in clear and unique way all over the Grid system scope.

In addition the user can associate a lease that she/he has previously created with the command `glite-ce-job-lease` by mean of the option `--leaseId` :

```
> glite-ce-job-submit -D mydelid -r cream-ce-01.pd.infn.it:8443/cream-lsf-grid02 --leaseId <my_le
```

To create a lease in the CREAM service, with a certain duration of time (expressed in seconds), issue the command:

```
glite-ce-job-lease --endpoint cream-27.pd.infn.it --leaseTime 3600 myLID
You requested lease time [3600] for lease ID [myLID]
CREAM negotiated the lease time to [3600]
```

The above command has created a lease on `cream-27.pd.infn.it` named "myLID" and lasting 1 hour.

Monitoring jobs

Passing the CREAM job identifiers returned by the `glite-ce-job-submit` command to the `glite-ce-job-status` command, it is possible to monitor the submitted jobs. Several (static and dynamic) information can be shown, depending on the chosen verbosity level. The verbosity level can be 0 (less verbosity), 1 or 2 (most verbosity).

Please note that specifying 0 as verbosity level means calling on the CREAM service a faster operation than when using 1 or 2 as verbosity level.

The most relevant attribute is the job status.

The following is an example of job status operation, specifying 1 as verbosity level:

```
$ glite-ce-job-status -L 1 https://cream-02.pd.infn.it:8443/CREAM738582717
****** JobID=[https://cream-02.pd.infn.it:8443/CREAM738582717]
Current Status = [DONE-FAILED]
```

```
ExitCode = [N/A]
FailureReason = [lsf_reason=256; Cannot move ISB (${globus_transfer_cmd}
gsiftp://cream-02.pd.infn.it//CREAMTests/Exe1/ssh1.sh file:///home/infngrid001/home_cream_7385827
error: globus_ftp_client: the server responded with an error 500 500-Command failed. : globus_l_g
500-globus_xio: Unable to open file //CREAMTests/Exe1/ssh1.sh
500-globus_xio: System error in open: No such file or directory
500-globus_xio: A system call failed: No such file or directory 500 End.]
Grid JobID = [N/A]

Job status changes:
-------------------
Status = [REGISTERED] - [Tue 22 Jan 2008 15:55:08] (1201013708)
Status = [PENDING] - [Tue 22 Jan 2008 15:55:08] (1201013708)
Status = [IDLE] - [Tue 22 Jan 2008 15:55:11] (1201013711)
Status = [RUNNING] - [Tue 22 Jan 2008 15:55:18] (1201013718)
Status = [DONE-FAILED] - [Tue 22 Jan 2008 16:03:10] (1201014190)

Issued Commands:
-------------------
*** Command Name = [JOB_REGISTER]
Command Category = [JOB_MANAGEMENT]
Command Status = [SUCCESSFULL]
*** Command Name = [JOB_START]
Command Category = [JOB_MANAGEMENT]
Command Status = [SUCCESSFULL]
```

In this example it is interesting to note that the job failed (as reported by the `Current Status` field) for the problem reported in the `FailureReason` field: the file to be transferred was not found.

Instead of explicitly specifying the identifiers of the jobs to monitor, the user can also ask to monitor all her jobs, in case specifying conditions (on the submission date and/or on the job status) that must be met.

For example to monitor all jobs, whose status is DONE-OK or DONE-FAILED, submitted to the `grid005.pd.infn.it` CREAM CE between July 23, 2005 10:00 and July 28, 2005 11:00, the following command must be issued:

```
> glite-ce-job-status --all -e grid005.pd.infn.it:8443 --from  2005-07-23 10:00:00  --to  2005-07
```

Retrieving output of jobs

User can choose to save the output sandbox (OSB) files on a remote server, or save them in the CREAM CE node. In the latter case these files can then be retrieved using the `glite-ce-job-output` command.

For example the following command retrieves the output sandbox files of the specified job from the relevant CREAM CE node:

```
> glite-ce-job-output https://cream-38.pd.infn.it:8443/CREAM295728364
2011-01-29 10:09:50,394 INFO - For JobID [https://cream-38.pd.infn.it:8443/CREAM295728364]
output will be stored in the dir ./cream-38.pd.infn.it_8443_CREAM295728364
```

This command can be used also to retrieve output produced by multiple jobs, by specifying multiple job identifiers as command's arguments .

Getting job identifiers

If a user is interested to get the identifiers of all her jobs submitted to a specific CREAM CE, she can use the `glite-ce-job-list` command. For example the following command returns the identifiers of all the jobs submitted to the specified CREAM CE, owned by the user issuing the command:

```
> glite-ce-job-list grid005.pd.infn.it:8443
```

Cancelling jobs

In some cases it might be needed to cancel jobs which have been previously submitted to CREAM based CEs. This can be achieved via the `glite-ce-job-cancel` command.

E.g., the command:

```
> glite-ce-job-cancel https://grid005.pd.infn.it:8443/CREAM115j5vfnf
```

cancels the specified job.

Suspending and resuming jobs

A running or idle job can be suspended (i.e. its execution will be stopped), and be resumed (i.e. it will run again) later. This can be achieved with the `glite-ce-job-suspend` and `glite-ce-job-resume` commands.

The following example shows that after having issued the `glite-ce-job-suspend` command, after a while the job status becomes `HELD`.

```
> glite-ce-job-suspend https://cream-ce-01.pd.infn.it:8443/CREAM11a79tnb2
Are you sure you want to suspend specified job(s) [y/n]: y
> glite-ce-job-status -L 0 https://cream-ce-01.pd.infn.it:8443/CREAM11a79tnb2
****** JobID=[https://cream-ce-01.pd.infn.it:8443/CREAM11a79tnb2]
Status = [HELD]
```

Issuing the `glite-ce-job-resume` command, the job will run/will be idle again:

```
> glite-ce-job-resume https://cream-ce-01.pd.infn.it:8443/CREAM11a79tnb2
Are you sure you want to resume specified job(s) [y/n]: y
> glite-ce-job-status -L 0 https://cream-ce-01.pd.infn.it:8443/CREAM11a79tnb2
****** JobID=[https://cream-ce-01.pd.infn.it:8443/CREAM11a79tnb2]
Status = [REALLY-RUNNING]
```

Purging jobs

A CREAM job can be monitored (via the `glite-ce-job-status`) even after it has completed its execution. A job gets "lost" (i.e. it is not possible to monitor or manage it anymore) only when the user who submitted it decides to explicitly clear it, or when the CREAM system administrator decides to do this purging operation. A user can purge her own jobs, using the `glite-ce-job-purge` command.

E.g., after having issued the command:

```
> glite-ce-job-purge https://cream-ce-01.pd.infn.it:8443/CREAM116jbi4o0
```

the specified job can't be managed anymore (e.g. it is not possible to check its status anymore).

Renewing proxies

It is possible that long jobs may outlive the validity of the initial delegated credentials; if so the job will die prematurely. To avoid this it is possible to renew the proxy of jobs submitted to CREAM CEs with the `glite-ce-proxy-renew` command.

E.g. the following command:

```
> glite-ce-proxy-renew -e cream-ce-01.pd.infn.it:8443 mydelid
```

renews the proxy of all the jobs having `mydelid` as delegation id.

It must be stressed that for jobs submitted to CREAM based CEs via the Workload Management System (WMS), proxy renewal is automatically dealt by the middleware.

Handling job identifiers

Handling the job identifiers directly quickly becomes tedious. To avoid this, you can make the `glite-ce-job-submit` and `glite-ce-job-list` commands append the job Id(s) to a named file using the `--output` (`-o`) option. On the other side, the CREAM client commands which take job identifier(s) as argument accept also the `--input` (`-i`) option which allows the job identifier(s) to be read from a file.

The following shows an example:

```
> glite-ce-job-submit -a -r cream-ce-01.pd.infn.it:8443/cream-lsf-grid02 -o idfile myjob.jdl
https://cream-ce-01.pd.infn.it:8443/CREAM116jbs5b9
```

The returned job id got also inserted in the specified file (`idfile`), which can be specified with the `--input` (`-i`) option e.g. with the `glite-ce-job-status` command:

```
> glite-ce-job-status -i idfile
****** JobID=[https://cream-ce-01.pd.infn.it:8443/CREAM116jbs5b9]
Status=[REALLY-RUNNING]
```

Restricting job submissions

In order to prevent that a CREAM CE gets overloaded, the CREAM CE administrator can set a specific policy to disable new job submissions when certain conditions are met.

If submissions are disabled because of that, if newer job submissions are attempted, users will get an error message such as:

```
> glite-ce-job-submit -a -r cream-38.pd.infn.it:8443/cream-pbs-creamtest1 oo.jdl
MethodName=[jobRegister] ErrorCode=[0] Description=[The CREAM service cannot accept jobs at the m
FaultCause=[Threshold for Load Average(1 min): 30 => Detected value for Load Average(1 min): 31.1
Timestamp=[Sat 29 Jan 2011 11:55:18]
```

In order to avoid degrading the performance of the system, the specified policy is not evaluated for each job submission, but instead it is evaluated and imposed from time to time (so it might happen that for a short time job submissions are allowed even if the specified threshold has been reached).

CREAM "super-users" can also disable newer job submissions via the command `glite-ce-disable-submission`. Submissions can then be re-enabled by a CREAM "super-user" via the command `glite-ce-enable-submission`.

To check if job submissions on a specific CREAM CE are allowed, the command `glite-ce-allowed-submission` can be used.

E.g.:

```
> glite-ce-disable-submission grid006.pd.infn.it:8443
```

```
Operation for disabling new submissions succeeded
>
> glite-ce-allowed-submission grid006.pd.infn.it:8443
Job Submission to this CREAM CE is disabled
>
> glite-ce-enable-submission grid006.pd.infn.it:8443
Operation for enabling new submissions succeeded
>
> glite-ce-allowed-submission grid006.pd.infn.it:8443
Job Submission to this CREAM CE is enabled
```

It must be stressed that if job submissions to a specific CREAM CE are disabled, all other operations (job status, job cancellations, etc.) can still be performed.

Getting information about the CREAM service

It is possible to get information about the CREAM service (interface and service version, status, etc) using the `glite-ce-service-info` command, e.g.:

```
> glite-ce-service-info cream-13.pd.infn.it:8443
Interface Version = [2.1]
Service Version = [1.12]
Description = [CREAM 2]
Started at = [Tue Nov 10 14:42:12 2009]
Submission enabled = [YES]
Status = [RUNNING]
Service Property = [SUBMISSION_THRESHOLD_MESSAGE]->
[Threshold for Load Average
(1 min): 10 => Detected value for Load Average(1 min): 0.03
Threshold for Load Average(5 min): 10 => Detected value for Load Average(5 min): 0.03
Threshold for Load Average(15 min): 10 => Detected value for Load Average(15 min): 0.00
Threshold for Memory Usage: 95 => Detected value for Memory Usage: 57.41%
Threshold for Swap Usage: 95 => Detected value for Swap Usage: 2.02%
Threshold for Free FD: 500 => Detected value for Free FD: 204500
Threshold for tomcat FD: 800 => Detected value for Tomcat FD: 107
Threshold for FTP Connection: 30 => Detected value for FTP Connection: 1
Threshold for Number of active jobs: -1 => Detected value for Number of active jobs: 0
Threshold for Number of pending commands: -1 => Detected value for Number of pending commands: 0
```

A CREAM CE is usually coupled with a CEMon service, which can be queried to get information about the CE and/or can notify clients with specific CE events. The command `glite-ce-get-cemon-url` can be used to get the end-point of this CEMon service, e.g.:

```
> glite-ce-get-cemon-url grid005.pd.infn.it:8443
https://grid005.pd.infn.it:8443/ce-monitor/services/CEMonitor
```

CREAM CLI configuration files

The configuration of the CREAM UI is accomplished via three possible configuration files:

- A general configuration file. This file is looked for in `/etc/glite_cream.conf`
- A VO specific configuration file. This file is looked for in `/etc/<VO> /glite_cream.conf`
- A user specific configuration file. This file is looked for in the following order:
  - The file specified with the `--conf` option of the considered command
  - The file referenced by the `$GLITE_CREAM_CLIENT_CONFIG` environment variable
  - `$HOME/.glite/<VO> /glite_cream.conf` (if the VO is defined), or `$HOME/.glite/glite_cream.conf` otherwise

Each of these files is a classad containing definitions.

If the same attribute is defined in more configuration file, the definition in the user specific configuration file (if any) is considered. Likewise the definitions in the VO specific configuration file have higher priority than the ones specified in the general configuration file.

It must be noted that one or more (even all) of these three configuration files can be missing.

CREAM CLI configuration file attributes

We list here the possible attributes that can be specified in the configuration files:

- `CREAM_URL_PREFIX`: the prefix to the `<hostname>:<port>` to build the CREAM service endpoint. The default is `https://`.
- `CREAMDELEGATION_URL_PREFIX`: the prefix to the `<hostname>:<port>` to build the CREAM delegation service endpoint. The default is `https://`.
- `DEFAULT_CREAM_TCPPORT`: the port to be appended to the hostname (if not specified by the user) to build the CREAM and CREAM delegation service endpoint. The default is 8443.
- `CREAM_URL_POSTFIX`: the postfix to be appended to the `<hostname>:<port>` to build the CREAM service endpoint. The default is `/ce-cream/services/CREAM2`.
- `CREAMDELEGATION_URL_POSTFIX`: the postfix to be appended to the `<hostname>:<port>` to build the CREAM delegation service endpoint. The default is `/ce-cream/services/gridsite-delegation`.
- `JDL_DEFAULT_ATTRIBUTES`: the classad that must be included by default in the user's JDLs. The default is an empty classad.
- `STATUS_VERBOSITY_LEVEL`: the default verbosity level to be used for the `glite-ce-job-status` command. The default value is 0.
- `UBERFTP_CLIENT` is the pathname of the `uberftp` client executable. The default value is `/usr/bin/uberftp`.
- `SUBMIT_LOG_DIR`: the directory where by default the log file `glite-ce-job-submit_CREAM_<username>_<date>_<time>.log` (created when the `--debug` option is used with the `glite-ce-job-submit` command) is created. The default is `/tmp/glite_cream_cli_logs`.
- `DELEGATE_LOG_DIR`: the directory where by default the log file `glite-ce-delegate-proxy_CREAM_<username>_<date>_<time>.log` (created when the `--debug` option is used with the `glite-ce-delegate-proxy` command) is created. The default is `/tmp/glite_cream_cli_logs`.
- `STATUS_LOG_DIR`: the directory where by default the log file `glite-ce-job-status_CREAM_<username>_<date>_<time>.log` (created when the `--debug` option is used with the glite-ce-job-status command) is created. The default is `/tmp/glite_cream_cli_logs`.
- `LIST_LOG_DIR`: the directory where by default the log file `glite-ce-job-list_CREAM_<username>_<date>_<time>.log` (created when the

--debug option is used with the glite-ce-job-list command) is created. The default is /tmp/glite_cream_cli_logs.

- SUSPEND_LOG_DIR: the directory where by default the log file glite-ce-job-suspend_CREAM_<username>_<date>_<time>.log (created when the --debug option is used with the glite-ce-job-suspend command) is created. The default is /tmp/glite_cream_cli_logs.

- RESUME_LOG_DIR: the directory where by default the log file glite-ce-job-resume_CREAM_<username>_<date>_<time>.log (created when the --debug option is used with the glite-ce-job-resume command) is created. The default is /tmp/glite_cream_cli_logs.

- CANCEL_LOG_DIR: the directory where by default the log file glite-ce-job-cancel_CREAM_<username>_<date>_<time>.log (created when the --debug option is used with the glite-ce-job-cancel command) is created. The default is /tmp/glite_cream_cli_logs.

- JOBOUTPUT_LOG_DIR: the directory where by default the log file glite-ce-job-output_CREAM_<username>_<date>_<time>.log (created when the --debug option is used with the glite-ce-job-output command) is created. The default is /tmp/glite_cream_cli_logs.

- PURGE_LOG_DIR: the directory where by default the log file glite-ce-job-purge_CREAM_<username>_<date>_<time>.log (created when the --debug option is used with the glite-ce-job-purge command) is created. The default is /tmp/glite_cream_cli_logs.

- ALLOWEDSUB_LOG_DIR: the directory where by default the log file glite-ce-allowed-submission_CREAM_<username>_<date>_<time>.log (created when the --debug option is used with the glite-ce-allowed-submission command) is created. The default is /tmp/glite_cream_cli_logs.

- ENABLE_LOG_DIR: the directory where by default the log file glite-ce-enable-submission_CREAM_<username>_<date>_<time>.log (created when the --debug option is used with the glite-ce-enable-submission command) is created. The default is /tmp/glite_cream_cli_logs.

- DISABLE_LOG_DIR: the directory where by default the log file glite-ce-disable-submission_CREAM_<username>_<date>_<time>.log (created when the --debug option is used with the glite-ce-disable-submission command) is created. The default is /tmp/glite_cream_cli_logs.

- PROXYRENEW_LOG_DIR: the directory where by default the log file glite-ce-proxy-renew_CREAM_<username>_<date>_<time>.log (created when the --debug option is used with the glite-ce-proxy-renew command) is created. The default is /tmp/glite_cream_cli_logs.

- GETSERVICEINFO_LOG_DIR: the directory where by default the log file glite-ce-service-info_CREAM_<username>_<date>_<time>.log (created when the --debug option is used with the glite-ce-service-info command) is created. The default is /tmp/glite_cream_cli_logs.

- GETCEMONURL_LOG_DIR: the directory where by default the log file

`glite-ce-get-cemon-url_CREAM_<username>_<date>_<time>.log` (created when the `--debug` option is used with the `glite-ce-get-cemon-url` command) is created. The default is `/tmp/glite_cream_cli_logs`.

As mentioned above, if the same attribute is defined in more than a configuration file, the definition in the user specific configuration file (if any) has higher priority than the definition in the VO specific configuration file (if any), which has higher priority than the definition in the generic configuration file. If an attribute is not defined anywhere, the default value is considered.

- `LEASE_LOG_DIR`: the directory where by default the log file `glite-ce-job-lease_CREAM_<username>_<date>_<time>.log` (created when the `--debug` option is used with the `glite-ce-job-lease` command) is created. The default is `/tmp/glite_cream_cli_logs`.

Example of CREAM CLI configuration file

The following represents an example of a CREAM UI configuration file:

```
[
JDL_DEFAULT_ATTRIBUTES = [
JobType=" Normal" ;
Type="job"
];
STATUS_VERBOSITY_LEVEL = 2;
CANCEL_LOG_DIR="tmp/CREAMLogs"
PURGE_LOG_DIR="tmp/CREAMLogs"
RESUME_LOG_DIR="tmp/CREAMLogs"
STATUS_LOG_DIR="tmp/CREAMLogs"
SUBMIT_LOG_DIR="tmp/CREAMLogs"
SUSPEND_LOG_DIR="tmp/CREAMLogs"
LIST_LOG_DIR="tmp/CREAMLogs"
DELEGATE_LOG_DIR="tmp/CREAMLogs"
LEASE_LOG_DIR="tmp/CREAMLogs"
]
```

Man pages for CREAM Command Line Interface

- glite-ce-job-submit
- glite-ce-delegate-proxy
- glite-ce-job-status
- glite-ce-job-list
- glite-ce-job-suspend
- glite-ce-job-resume
- glite-ce-job-cancel
- glite-ce-job-output
- glite-ce-job-purge
- glite-ce-proxy-renew
- glite-ce-allowed-submission
- glite-ce-enable-submission
- glite-ce-disable-submission
- glite-ce-service-info
- glite-ce-get-cemon-url
- glite-ce-job-lease

Use specific functionality of the CREAM CE

Submission on multi-core resources

As explained in the CREAM JDL guide , the following JDL attributes:

- CPUNumber
- SMPGranularity
- WholeNodes
- HostNumber

are relevant for submissions to multi core resources.

This section explains how this is actually implemented for LSF and Torque/PBS:

First scenario

With a JDL such as:

```
WholeNodes=true;
SMPGranularity=G;
 Hostnumber=H;
```

with H > 1.

In the submission script there will be:

- for LSF:

```
BSUB -n S*H
BSUB -R "span[ptile=S]
BSUB -x
```

- for PBS:

```
PBS -l nodes=H:ppn=S
PBS -W x=NACCESSPOLICY:SINGLEJOB
```

with S equal to the value published as `GlueHostArchitectureSMPSize`.

Second scenario

With a JDL such as:

```
WholeNodes=true;
SMPGranularity=G;
```

in the submission script there will be:

- for LSF:

```
BSUB -n S
BSUB -R "span[hosts=1]"
BSUB -x
```

- for PBS:

```
PBS -l nodes=1:ppn=S
PBS -W x=NACCESSPOLICY:SINGLEJOB
```

Third scenario

With a JDL such as:

```
WholeNodes=true;
HostNumber=H;
```

with H>1.

in the submission script there will be:

- for LSF:

```
BSUB -n S*H
BSUB -R "span[ptile=S]"
BSUB -x
```

- for PBS:

```
PBS -l nodes=H:ppn=S
PBS -W x=NACCESSPOLICY:SINGLEJOB
```

with S equal to the value published as `GlueHostArchitectureSMPSize`.

Forth scenario

With a JDL such as:

```
WholeNodes=false;
SMPGranularity=G;
CPUNumber=C;
```

in the submission script there will be:

- for LSF:

```
BSUB -n C
BSUB -R "span[ptile=G]"
```

- for PBS:

```
 PBS  -l nodes=N:ppn=G { [+1:ppn=R] if r>0 }
```

with:

```
N = C / G
R = C % G
```

Fifth scenario

With a JDL such as:

```
WholeNodes=false;
HostNumber=H;
CPUNumber=C;
```

with H>=1.

in the submission script there will be:

- for LSF:

```
BSUB -n C
BSUB -R "span[ptile={ N if R=0 ; N+1 if R>0 }]"
```

- for PBS:

```
PBS -l nodes=H-R:ppn=N { [+R:ppn=N+1] if R>0 }
```

with:

```
N = C / H
R = C % H
```

Sixth scenario

With a JDL such as:

```
WholeNodes=false;
CPUNumber=C;
```

in the submission script there will be:

- for LSF:

```
BSUB -n C
```

- for PBS:

```
PBS -l nodes=C
```

Forward of requirements to the batch system

The CREAM CE allows to forward, via tha BLAH component, requirements to the batch system.

For this purpose the JDL `CERequirements` attribute, described at http://wiki.italiangrid.org/twiki/bin/view/CREAM/JdlGuide#3_27_CERequirements , can be used.

For direct submissions to the CREAM CE (e.g. jobs submitted to the CREAM CE using the CREAM CLI `glite-ce-job-submit` command) the CeRequirements attribute is supposed to be filled by the end-user.

For jobs submitted to the CREAM CE via the WMS, the `CeRequirements` attribute is instead filled by the WMS, considering the JDL `Requirements` expression and the value of the `CeForwardParameters` attribute in the WMS configuration file.

For example, if in the user JDL there is :

```
Requirements= "other.GlueHostMainMemoryRAMSize > 100 && other.GlueCEImplementationName==\"CREAM\"
```

and if the WMS configuration file there is:

```
CeForwardParameters  = {"GlueHostMainMemoryVirtualSize","GlueHostMainMemoryRAMSize", "GlueCEPolic
```

in the JDL sent by the WMS to CREAM there will be:

```
CeRequirements= "other.GlueHostMainMemoryRAMSize > 100";
```

The `CERequirements` expression received by CREAM is then forwarded to BLAH. Basically BLAH manages the `CERequirements` expression setting some environment variables, which are available and can be properly used by the `/usr/libexec/xxx_local_submit_attributes.sh` script (e.g. `/usr/libexec/pbs_local_submit_attributes.sh` for PBS/Torque, `/usr/libexec/lsf_local_submit_attributes.sh` for LSF). This script must be properly created by the site admin.

For example, considering the following CeRequirements expression:

```
CeRequirements="other.GlueHostMainMemoryRAMSize > 100 && other.GlueCEStateWaitingJobs <10 && othe
```

the following settings will be available in `$USR_LOCATION/libexec/xxx_local_submit_attributes.sh`:

```
GlueHostMainMemoryRAMSize_Min='100'
GlueCEStateWaitingJobs_Max='10'
GlueCEImplementationName='CREAM'
GlueHostProcessorClockSpeed_Min='2800'
GlueHostApplicationSoftwareRuntimeEnvironment='"FDTD"'
```

where the value for $USR_LOCATION in a standard installation of a CREAM CE is "/usr".

What is printed by the `/usr/libexec/xxx_local_submit_attributes.sh` script is automatically added to the submit command file.

For example if the JDL `CeRequirements` expression is:

```
CeRequirements = "(Member(\"FDTD\", other.GlueHostApplicationSoftwareRuntimeEnvironment))";
```

and the `/usr/libexec/pbs_local_submit_attributes.sh` is:

```
#!/bin/sh
if [ "$other.GlueHostApplicationSoftwareRuntimeEnvironment" == "FDTD" ]; then
 echo "#PBS -l software=FDTD"
fi
```

then the PBS submit file that will be used will include:

```
...
...
# PBS directives:
#PBS -S /bin/bash
#PBS -o /dev/null
#PBS -e /dev/null
#PBS -l software=FDTD
....
....
```

where the line:

```
#PBS -l software=FDTD
```

is set via the `/usr/libexec/pbs_local_submit_attributes.sh` script.

Please note that there are no differences if in `CeRequirements` expresssion there is e.g.

```
CeRequirements = other.xyz==\"ABC\"
```

or:

```
CeRequirements = "xyz==\"ABC\"";
```

In both cases in `/usr/libexec/xxx_local_submit_attributes.sh` the variable `xyz` will be set.

As shown above, having `x>a` or `x>=a` doesn't make any difference in the setting of the environment variable `x` in the `/usr/libexec/xxx_local_submit_attributes.sh` script. It will be in both cases:

```
x_Min='a'
```

Starting with EMI-2 (i.e. BLAH v. >= 1.18) it is possible to forward to the batch system also other attributes not included in the `CeRequiments` JDL attribute.

This can be done adding in `/etc/blah.config` the line:

```
blah_pass_all_submit_attributes=yes
```

In this way the `xxx_local_submit_attributes.sh` will see the following environment variables set:

- gridType
- x509UserProxyFQAN
- uniquejobid
- queue
- ceid
- VirtualOrganisation
- ClientJobId
- x509UserProxySubject

It is also possible to specify that only some attributes must be forwarded in the batch system setting in `blah.config` e.g.:

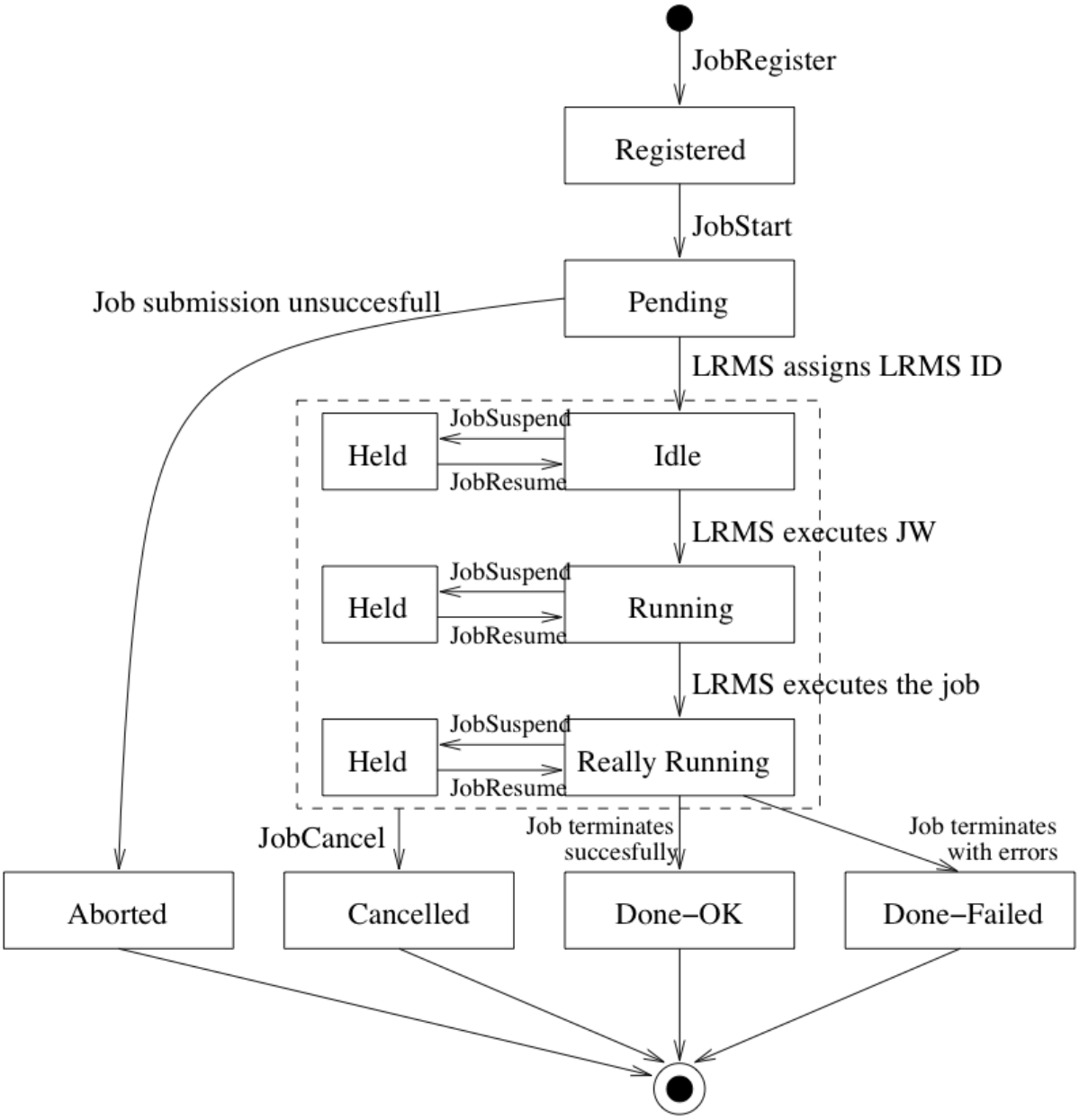
```
blah_pass_submit_attributes[0]="x509UserProxySubject"
blah_pass_submit_attributes[1]="x509UserProxyFQAN"
```

CREAM job states

Here below is provided a brief description of the meaning of each possible state a CREAM job can enter:

- `REGISTERED`: the job has been registered but it has not been started yet.
- `PENDING` the job has been started, but it has still to be submitted to the LRMS abstraction layer module (i.e. BLAH).
- `IDLE`: the job is idle in the Local Resource Management System (LRMS).
- `RUNNING`: the job wrapper, which "encompasses" the user job, is running in the LRMS.
- `REALLY-RUNNING`: the actual user job (the one specified as Executable in the job JDL) is running in the LRMS.
- `HELD`: the job is held (suspended) in the LRMS.
- `CANCELLED`: the job has been cancelled.
- `DONE-OK`: the job has successfully been executed.
- `DONE-FAILED`: the job has been executed, but some errors occurred.
- `ABORTED`: errors occurred during the "management" of the job, e.g. the submission to the LRMS abstraction layer software (BLAH) failed.
- `UNKNOWN`: the job is an unknown status.

The following figure shows the possible job states transitions.



he CE

The previous example was about a simple activity that doesn't involve sandboxes to move around; if the user needs to send one or more files to the CE, or needs that the CE sends files after activity termination, then a more complex activity must be written and sent to the CE; before to create the activity a delegation must be created on the CE (or "a proxy must be delegated on the CE"), because moving the sandbox files could imply to contact remote authenticated services, like GridFTP servers. To delegate his/her own proxy on the CE, the user has to issue this command:

```
$ glite-es-delegate-proxy cream-10.pd.infn.it
DelegationID = 04669318871724504
```

(this command **`glite-es-delegate-proxy`** will be explained more deeply later).

The returned delegation identifier **`04669318871724504`** (that is an handle to the real delegated proxy residing on the CE **`cream-10.pd.infn.it`**, whose lifetime is equal to the user's proxy lifetime) must be inserted in the proper ADL's section, like in this example:

```
$ cat ~/JDLs/activity_files.adl

<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<CreateActivities>
    <ActivityDescription>
        <ActivityIdentification>
            <Name>CustomJob</Name>
            <Description>A job that to run needs a file to be sent and writes its elaboration on
            <Type>single</Type>
        </ActivityIdentification>
        <Application>
            <Executable>
                <Path>myjob.sh</Path>
                <Argument></Argument>
            </Executable>
            <Environment>
              <Name>MY_ENV</Name>
              <Value>"my env"</Value>
            </Environment>
            <Error>JobError.txt</Error>
            <Output>JobOutput.txt</Output>
        </Application>
        <Resources>
            <QueueName>creamtest2</QueueName>
        </Resources>
        <DataStaging>
          <OutputFile>
            <Name>JobError.txt</Name>
            <Target>
              <URI>gsiftp://cream-23.pd.infn.it//tmp/JobError.txt</URI>
              <DelegationID>04669318871724504</DelegationID>
            </Target>
          </OutputFile>
          <OutputFile>
            <Name>JobOutput.txt</Name>
            <Target>
              <URI>gsiftp://cream-23.pd.infn.it//tmp/JobOutput.txt</URI>
              <DelegationID>04669318871724504</DelegationID>
            </Target>
          </OutputFile>
          <InputFile>
            <IsExecutable>true</IsExecutable>
            <Name>myjob.sh</Name>
            <Source>
              <URI>gsiftp://cream-23.pd.infn.it//tmp/myjob.sh</URI>
              <DelegationID>04669318871724504</DelegationID>
            </Source>
          </InputFile>
        </DataStaging>
    </ActivityDescription>
</CreateActivities>
```

When the proxy is delegated, and the ADL does contain the delegation identifier associated to the files to move, the activity can be created:

```
$ glite-es-activity-create -e cream-10.pd.infn.it ~/JDLs/activity_files.adl
****************************************
ActivityID     = CR_ES292405349
ActivityMgrURI = https://cream-10.pd.infn.it:8443/ce-cream-es/services/ActivityManagementService
Status         = PREPROCESSING
Status Attrs   = {}
Timestamp      = Tue Mar 20 10:59:15 2012
Description    =
ETNSC          =
STAGEIN Dir    = {gsiftp://cream-10.pd.infn.it/var/cream-es_sandbox/dteam/CN_Alvise_Dorigo_L_Pado
```

```
SESSION Dir    = {}
STAGEOUT Dir   = {gsiftp://cream-10.pd.infn.it/var/cream-es_sandbox/dteam/CN_Alvise_Dorigo_L_Pado
```

In the last example, the activity represents a job to execute, and it is not a system executable (like the **/bin/sleep** of the previous example); the executable is built up by the user, **myjob.sh**, and must be staged out from a storage server (**cream-23.pd.infn.it** in this example; see again the **<Source>** node in the ADL above) running a GridFTP daemon, into the activity's directory in the CE. We do not enter into the job's details: simply note that the user's executable produces a standard output, and a standard error that will be redirected to dedicated files (the **JobOutput.txt** and **JobError.txt** specified in the ADL above). The standard output will contain the print of a particular user-defined environment variable (MY_ENV, see the ADL above) and an **echo** message; the standard error will contain an error message triggered by operations that the user cannot perform on the destination worker node.

In the ADL it is written that the two files **JobOutput.txt** and **JobError.txt** must be sent to the storage server (the same GridFTP server seen before for the executable stage-out) **cream-23.pd.infn.it**; of course it could be a different storage server. The user can retrieve these output files by mean of his/her preferred GridFTP client. For example:

```
$ globus-url-copy gsiftp://cream-23.pd.infn.it/tmp/JobOutput.txt JobOutput.txt
$ globus-url-copy gsiftp://cream-23.pd.infn.it/tmp/JobError.txt JobError.txt
$ ls -l Job*
-rw------- 1 dorigoa dorigoa 120 Mar 21 10:31 JobError.txt
-rw------- 1 dorigoa dorigoa  73 Mar 21 10:31 JobOutput.txt
```

Please note that the destination CE **cream-10.pd.infn.it** has sent the two files **JobOutput.txt** and **JobError.txt** to the storage server **cream-23.pd.infn.it** using the proxy that the user delegated before (identified by the string **04669318871724504**, that has been put in the ADL file) for authentication. The delegated proxy must be valid when the stage-in occurs. Then it is up to the user to delegate a proxy into the CE, that has a lifetime long enough for the entire time of the job's life and files stage-in.

Stage-in performed by the user

Consider this ADL:

```
<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<CreateActivities>
    <ActivityDescription>
        <ActivityIdentification>
            <Name>CustomJob</Name>
            <Description>A job that to run needs a file to be sent and writes its elaboration on
            <Type>single</Type>
        </ActivityIdentification>
        <Application>
            <Executable>
                <Path>myjob.sh</Path>
                <Argument></Argument>
            </Executable>
            <Environment>
              <Name>MY_ENV</Name>
              <Value>"my env"</Value>
            </Environment>
            <Error>JobError.txt</Error>
            <Output>JobOutput.txt</Output>
        </Application>
        <Resources>
            <QueueName>creamtest2</QueueName>
        </Resources>
```

```
        <DataStaging>
          <ClientDataPush>true</ClientDataPush>
          <OutputFile>
            <Name>JobError.txt</Name>
            <Target>
              <URI>gsiftp://cream-23.pd.infn.it//tmp/JobError.txt</URI>
              <DelegationID>04352064184726456</DelegationID>
            </Target>
          </OutputFile>
          <OutputFile>
            <Name>JobOutput.txt</Name>
            <Target>
              <URI>gsiftp://cream-23.pd.infn.it//tmp/JobOutput.txt</URI>
              <DelegationID>04352064184726456</DelegationID>
            </Target>
          </OutputFile>
        </DataStaging>
    </ActivityDescription>
</CreateActivities>
```

Note the new XML node **<ClientDataPush>** and the lack of the **<InputFile>** node. This ADL tells the CE that the ISB file (in this case the user's executable **myjob.sh**) will be sent by the user by hand and the CE has to wait until this operation is performed.

```
$ glite-es-activity-create -e cream-10.pd.infn.it ~/JDLs/activity_files_push.adl
****************************************
ActivityID     = CR_ES613269972
ActivityMgrURI = https://cream-10.pd.infn.it:8443/ce-cream-es/services/ActivityManagementService
Status         = PREPROCESSING
Status Attrs   = {CLIENT_STAGEIN_POSSIBLE}
Timestamp      = Wed Mar 21 13:36:55 2012
Description    =
ETNSC          =
STAGEIN Dir    = {gsiftp://cream-10.pd.infn.it/var/cream-es_sandbox/dteam/CN_Alvise_Dorigo_L_Pado
SESSION Dir    = {}
STAGEOUT Dir   = {gsiftp://cream-10.pd.infn.it/var/cream-es_sandbox/dteam/CN_Alvise_Dorigo_L_Pado
```

Note the status attribute **CLIENT_STAGEIN_POSSIBLE**: the CE is saying that it is waiting for the user to send the required ISB by hand (by mean of the usual GridFTP client for example). Also note that the CE is communicating the complete address where to send the ISB (and the protocol supported, gsiftp):

```
$ globus-url-copy myjob.sh gsiftp://cream-10.pd.infn.it/var/cream-es_sandbox/dteam/CN_Alvise_Dori
```

When the stage-in is finished, the user has to notify the service that the file shipment is complete:

```
$ glite-es-notify-service -e cream-10.pd.infn.it CR_ES613269972:CLIENT-DATAPUSH-DONE
```

The activity is started by the CE and the OSB are available as usual where specified in the ADL (**cream-23.pd.infn.it**).

In this example a new command has been introduced: **glite-es-notify-service**, that will be described more deeply later.

Both stage-in and stage-out performed by the user

Consider this ADL:

```
<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<CreateActivities>
    <ActivityDescription>

        <ActivityIdentification>
            <Name>CustomJob</Name>
            <Description>A job that to run needs a file to be sent and writes its elaboration on
            <Type>single</Type>
        </ActivityIdentification>

        <Application>
            <Executable>
                <Path>myjob.sh</Path>
                <Argument></Argument>
                <FailIfExitCodeNotEqualTo>0</FailIfExitCodeNotEqualTo>
            </Executable>
            <Environment>
              <Name>MY_ENV</Name>
              <Value>"my env"</Value>
            </Environment>
            <Error>JobError.txt</Error>
            <Output>JobOutput.txt</Output>
        </Application>
        <Resources>
            <QueueName>creamtest2</QueueName>
        </Resources>

        <DataStaging>
          <ClientDataPush>true</ClientDataPush>
          <OutputFile>
            <Name>JobError.txt</Name>
          </OutputFile>
          <OutputFile>
            <Name>JobOutput.txt</Name>
          </OutputFile>
        </DataStaging>

    </ActivityDescription>
</CreateActivities>
```

This ADL doesn't contain specification of **<Target>** node, which means that the CE doesn't have to send any files to any remote server; it will be up to the user to retrieve the **JobOutput.txt** and **JobError.txt** files. As before, just after activity creation, the CE will send back the client the paths for stage-in and stage out. Let's skip the description of stage-in of executabe **myjob.sh** already described above. The activity creation with this ADL will return (as usual):

```
$ glite-es-activity-create -e cream-05.pd.infn.it ~/JDLs/activity_files_push_pull.adl
****************************************
ActivityID     = CR_ES920948151
ActivityMgrURI = https://cream-05.pd.infn.it:8443/ce-cream-es/services/ActivityManagementService
Status         = PREPROCESSING
Status Attrs   = {CLIENT_STAGEIN_POSSIBLE}
Timestamp      = Thu Mar 22 13:28:32 2012
Description    =
ETNSC          =
STAGEIN Dir    = {gsiftp://cream-05.pd.infn.it/var/cream-es_sandbox/dteam/CN_Alvise_Dorigo_L_Pado
SESSION Dir    = {}
```

```
STAGEOUT Dir   = {gsiftp://cream-05.pd.infn.it/var/cream-es_sandbox/dteam/CN_Alvise_Dorigo_L_Pado
```

This time, in addition to the "STAGEIN Dir", the user has to consider (and remember) the value of "STAGEOUT Dir"; when the activity will be finished (use **`glite-es-activity-status`** to check that), he/she will have to retrieve the two files by mean of the usual GridFTP client:

```
$ glite-es-activity-status  -e cream-05.pd.infn.it CR_ES920948151

****************************************
ActivityID  = CR_ES920948151
Status      = TERMINAL
Attributes  = {CLIENT_STAGEOUT_POSSIBLE}
Timestamp   = Thu Mar 22 13:29:11 2012
Description = reason=0

$ globus-url-copy gsiftp://cream-05.pd.infn.it/var/cream-es_sandbox/dteam/CN_Alvise_Dorigo_L_Pado
$ globus-url-copy gsiftp://cream-05.pd.infn.it/var/cream-es_sandbox/dteam/CN_Alvise_Dorigo_L_Pado
```

When output file retrieve is finished the user should notify the server with the command:

```
$ glite-es-notify-service CR_ES920948151:CLIENT-DATAPULL-DONE -e cream-05.pd.infn.it
```

Now the status's attributes CLIENT_STAGEOUT_POSSIBLE disappeared:

```
dorigoa@lxgrid05 14:46:46 ~/emi/creamui_emi2>stage/usr/bin/glite-es-activity-status CR_ES92094815
****************************************
ActivityID  = CR_ES920948151
Status      = TERMINAL
Attributes  = {}
Timestamp   = Thu Mar 22 13:29:11 2012
Description = reason=0
```

Using an output file for the command `glite-es-activity-create`

The activity creation command has the ability to write the activity identifier(s) into an output file. If this file already exists, the user has to make sure that the output file has been already used for the same endpoint of the current activity creation. En example is better than any further explanation (the special ADL file **`~/JDLs/activity_sleep60_multiple.adl`** cointains 3 descriptions for 3 activities):

```
$ glite-es-activity-create -e cream-05.pd.infn.it ~/JDLs/activity_sleep60_multiple.adl -o ids_cre
****************************************
ActivityID     = CR_ES588248744
ActivityMgrURI = https://cream-05.pd.infn.it:8443/ce-cream-es/services/ActivityManagementService
Status         = PREPROCESSING
Status Attrs   = {}
Timestamp      = Thu Mar 22 14:03:48 2012
Description    =
ETNSC          =
STAGEIN Dir    = {gsiftp://cream-05.pd.infn.it/var/cream-es_sandbox/dteam/CN_Alvise_Dorigo_L_Pado
SESSION Dir    = {}
STAGEOUT Dir   = {gsiftp://cream-05.pd.infn.it/var/cream-es_sandbox/dteam/CN_Alvise_Dorigo_L_Pado
****************************************
ActivityID     = CR_ES477186035
```

```
ActivityMgrURI = https://cream-05.pd.infn.it:8443/ce-cream-es/services/ActivityManagementService
Status         = PREPROCESSING
Status Attrs   = {}
Timestamp      = Thu Mar 22 14:03:48 2012
Description    =
ETNSC          =
STAGEIN Dir    = {gsiftp://cream-05.pd.infn.it/var/cream-es_sandbox/dteam/CN_Alvise_Dorigo_L_Pado
SESSION Dir    = {}
STAGEOUT Dir   = {gsiftp://cream-05.pd.infn.it/var/cream-es_sandbox/dteam/CN_Alvise_Dorigo_L_Pado
****************************************
ActivityID     = CR_ES276464760
ActivityMgrURI = https://cream-05.pd.infn.it:8443/ce-cream-es/services/ActivityManagementService
Status         = PREPROCESSING
Status Attrs   = {}
Timestamp      = Thu Mar 22 14:03:49 2012
Description    =
ETNSC          =
STAGEIN Dir    = {gsiftp://cream-05.pd.infn.it/var/cream-es_sandbox/dteam/CN_Alvise_Dorigo_L_Pado
SESSION Dir    = {}
STAGEOUT Dir   = {gsiftp://cream-05.pd.infn.it/var/cream-es_sandbox/dteam/CN_Alvise_Dorigo_L_Pado
```

Note the option **`-o ids_cream-05`** that instructs the command to write (only) the activity identifiers on the output file **`ids_cream-05`**. Let's take a look at this file:

```
$ cat ids_cream-05
##ACTIVITY_IDS@cream-05.pd.infn.it:8443##
CR_ES588248744
CR_ES477186035
CR_ES276464760
```

The header **`##ACTIVITY_IDS@cream-05.pd.infn.it:8443##`** indicates the endpoint where the following identifiers has been created. If the user tries to create activites to a different endpoint but wants to write the output identifiers on the file **`ids_cream-05`**, he/she will get an error:

```
$ glite-es-activity-create -e cream-10.pd.infn.it ~/JDLs/activity_sleep60.adl -o ids_cream-05
****************************************
ActivityID     = CR_ES341612391
ActivityMgrURI = https://cream-10.pd.infn.it:8443/ce-cream-es/services/ActivityManagementService
Status         = PREPROCESSING
Status Attrs   = {}
Timestamp      = Thu Mar 22 14:08:52 2012
Description    =
ETNSC          =
STAGEIN Dir    = {gsiftp://cream-10.pd.infn.it/var/cream-es_sandbox/dteam/CN_Alvise_Dorigo_L_Pado
SESSION Dir    = {}
STAGEOUT Dir   = {gsiftp://cream-10.pd.infn.it/var/cream-es_sandbox/dteam/CN_Alvise_Dorigo_L_Pado
This output file [ids_cream-05] contains activity identifiers created a on different endpoint (cr
```

Note the error message at the end of the output: **This output file [ids_cream-05] contains activity identifiers created a on different endpoint (cream-05.pd.infn.it:8443). Will not write Activity ID on this file.**. In fact the endpoint of this command was **`cream-10.pd.infn.it`**.

Using an input file for the command `glite-es-activity-status`

We've seen above that the command **`glite-es-activity-status`** accepts one or more activity identifiers as arguments. Alternatively, if the identifiers are many, it could be convenient to load them from an input file (by mean of the **`-i`** option), previously created by the **`glite-es-activity-create`** command with the **`-o`** option. In this case the user must not specify the endpoint where the identifiers have been created, because the endpoint will be extracted from the input file:

```
$ glite-es-activity-status -e cream-05.pd.infn.it -i ids_cream-05
Options --endpoint and --input are mutually exclusive. Stop.

$ glite-es-activity-status -i ids_cream-05
****************************************
ActivityID  = CR_ES276464760
Status      = TERMINAL
Attributes  = {}
Timestamp   = Thu Mar 22 14:05:15 2012
Description = reason=0
****************************************
ActivityID  = CR_ES477186035
Status      = TERMINAL
Attributes  = {}
Timestamp   = Thu Mar 22 14:05:17 2012
Description = reason=0
****************************************
ActivityID  = CR_ES588248744
Status      = TERMINAL
Attributes  = {}
Timestamp   = Thu Mar 22 14:05:05 2012
Description = reason=0
```

Another possibility is to mix identifiers specified as arguments and an input file:

```
$ glite-es-activity-status CR_ES920948151 -i ids_cream-05
****************************************
ActivityID  = CR_ES276464760
Status      = TERMINAL
Attributes  = {}
Timestamp   = Thu Mar 22 14:05:15 2012
Description = reason=0
****************************************
ActivityID  = CR_ES477186035
Status      = TERMINAL
Attributes  = {}
Timestamp   = Thu Mar 22 14:05:17 2012
Description = reason=0
****************************************
ActivityID  = CR_ES588248744
Status      = TERMINAL
Attributes  = {}
Timestamp   = Thu Mar 22 14:05:05 2012
Description = reason=0
****************************************
ActivityID  = CR_ES920948151
Status      = TERMINAL
Attributes  = {CLIENT_STAGEOUT_POSSIBLE}
Timestamp   = Thu Mar 22 13:29:11 2012
Description = reason=0
```

Obtaining information about activities

Activity status

The command **`glite-es-activity-status`** has already been explained here and here .

Extended activity's information

To obtain extended information about an activity the command **`glite-es-activity-info`** can be used with exactly the same syntax and options than **`glite-es-activity-status`** :

```
$ glite-es-activity-info CR_ES920948151 -e cream-05.pd.infn.it

**** ActivityID = CR_ES920948151

ID=https://cream-05.pd.infn.it:8443/ce-cream-es/services/ActivityManagementService?CR_ES920948151
Name=CustomJob
OtherInfo={}
CreationTime=1332419311 (Thu Mar 22 13:28:31 2012)
Validity=N/A
***** Extension #0:
      LocalID=CR_ES920948151
      Key=ACTIVITY_HISTORY
      Value=ACTIVITY_HISTORY
***** Extension #1:
      LocalID=CR_ES920948151
      Key=STAGE_IN_URI
      Value=gsiftp://cream-05.pd.infn.it/var/cream-es_sandbox/dteam/CN_Alvise_Dorigo_L_Padova_OU_
***** Extension #2:
      LocalID=CR_ES920948151
      Key=STAGE_OUT_URI
      Value=gsiftp://cream-05.pd.infn.it/var/cream-es_sandbox/dteam/CN_Alvise_Dorigo_L_Padova_OU_
BaseType=
Type=SINGLE
IDFromEndpoint=https://cream-05.pd.infn.it:8443/ce-cream-es/services/ActivityManagementService?CR
LocalIDFromManager=lsf/20120322/620153
JobDescription=emi:adl
State={ACCEPTED, PREPROCESSING, PROCESSING-ACCEPTING, PROCESSING-QUEUED, PROCESSING-RUNNING, TERM
RestartState={}
ExitCode=1
ComputingManagerExitCode=N/A
Error={}
WaitingPosition=N/A
UserDomain=dteam
Owner=CN_Alvise_Dorigo_L_Padova_OU_Personal_Certificate_O_INFN_C_IT_dteam_Role_NULL_Capability_NU
LocalOwner=dteam002
RequestedTotalWallTime=N/A
RequestedTotalCPUTime=N/A
RequestedSlots=N/A
RequestedApplicationEnvironment={}
StdIn=JobOutput.txt
StdOut=N/A
StdErr=JobError.txt
LogDir=N/A
ExecutionNode=prod-wn-001.pn.pd.infn.it
Queue=creamtest2
UsedTotalWallTime=N/A
UsedMainMemory=N/A
SubmissionTime=1332419311 (Thu Mar 22 13:28:31 2012)
ComputingManagerSubmissionTime=1332419344 (Thu Mar 22 13:29:04 2012)
StartTime=1332419351 (Thu Mar 22 13:29:11 2012)
ComputingManagerEndTime=1332419351 (Thu Mar 22 13:29:11 2012)
EndTime=1332419351 (Thu Mar 22 13:29:11 2012)
WorkingAreaEraseTime=N/A
ProxyExpirationTime=N/A
SubmissionHost=N/A
SubmissionClientName=N/A
OtherMessages=
```

Refer to the EMI-ES documentation for a complete explanation of this output.

As the status command, also **glite-es-activity-info** supports more than one activity identifier as argument, or an input file (option **-i**), or both.

Obtaining the activity identifier created on an endpoint

To get a list of activities the user has created on a certain endpoint, the command **glite-es-activity-list** can be used:

```
$ glite-es-activity-list cream-05.pd.infn.it
CR_ES075634021
CR_ES117045296
CR_ES225992668
CR_ES276464760
CR_ES330911211
CR_ES342891609
CR_ES477186035
CR_ES556329337
CR_ES588248744
CR_ES800305261
CR_ES832095061
CR_ES920948151
```

Remember that a certain user can see **ONLY** his/her activity identifiers (those ones created with his/her proxy certificate).

glite-es-activity-list can redirect its output into a file; if this file already exists, it must be of the same type of the **glite-es-activity-create** 's one, i.e. it must contain the usual header line:

```
$ cat ids_cream-05
##ACTIVITY_IDS@cream-05.pd.infn.it:8443##
CR_ES588248744
CR_ES477186035
CR_ES276464760
```

The option to use is again **-o** :

```
$ glite-es-activity-list cream-05.pd.infn.it -o cream-05_activity_list
CR_ES075634021
CR_ES117045296
CR_ES225992668
CR_ES276464760
CR_ES330911211
CR_ES342891609
CR_ES477186035
CR_ES556329337
CR_ES588248744
CR_ES800305261
CR_ES832095061
CR_ES920948151

$ cat cream-05_activity_list
##ACTIVITY_IDS@cream-05.pd.infn.it:8443##
CR_ES075634021
CR_ES117045296
CR_ES225992668
```

```
CR_ES276464760
CR_ES330911211
CR_ES342891609
CR_ES477186035
CR_ES556329337
CR_ES588248744
CR_ES800305261
CR_ES832095061
CR_ES920948151
```

Once created (or updated), this file can be used to feed **glite-es-activity-status** or **glite-es-activity-info** (and other commands we will se later). As for the activity creation, if the output file already exists, it must contain activity identifiers created on the same endpoint where the user is asking the list:

```
$ head -1 cream-10_activity_list
##ACTIVITY_IDS@cream-10.pd.infn.it:8443##

$ glite-es-activity-list cream-05.pd.infn.it -o cream-10_activity_list
CR_ES075634021
CR_ES117045296
CR_ES225992668
CR_ES276464760
CR_ES330911211
CR_ES342891609
CR_ES477186035
CR_ES556329337
CR_ES588248744
CR_ES800305261
CR_ES832095061
CR_ES920948151
This output file [cream-10_activity_list] contains activity identifiers created a on different en
```

Managing activities

Managing activities means: cancel them, pause them, resume them, restart them, wipe them when they're in TERMINAL state. All these operations are performed by the commands: **glite-es-activity-cancel**, **glite-es-activity-pause**, **glite-es-activity-resume**, **glite-es-activity-restart**, **glite-es-activity-wipe**. These commands have exactly the same syntax, then we will explain just once and we will used a symbolic name **<command>** for all of them.

The **<command>** 's syntax is simple:

```
<command> -e <endpoint> <Activity_ID_1> .. <Activity_ID_N>
```

Where N must be >= 1 (i.e. at least one activity ID is a mandatory argument); unless the user specify an input file as explained below.

The user can specify an input file (as for the **glite-es-activity-status**) instead of using **<endpoint>** and activity identifiers; or he/she can mixes input file and activity identifiers. Then, the 3 possibilities of usage are:

```
<command> -e cream-05.pd.infn.it CR_ES075634021 CR_ES342891609 CR_ES832095061
```

or

```
<command> -i cream-10_activity_list
```

or

```
<command> -i cream-10_activity_list CR_ES081635022
```

Notifying service about user's operations

When the user's ADL contains files to be sent to the CE before the activity starts, or files that need to be retrieved after activity's termination, just after activity creation the CE puts the activity in a particular waiting state as we have seen above:

```
$ cat ~/JDLs/activity_files_push.adl
<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<CreateActivities>
    <ActivityDescription>

        <ActivityIdentification>
            <Name>CustomJob</Name>
            <Description>A job that to run needs a file to be sent and writes its elaboration on
            <Type>single</Type>
        </ActivityIdentification>

        <Application>
            <Executable>
                <Path>myjob.sh</Path>
                <Argument></Argument>
                <FailIfExitCodeNotEqualTo>0</FailIfExitCodeNotEqualTo>
            </Executable>
            <Environment>
              <Name>MY_ENV</Name>
              <Value>"my env"</Value>
            </Environment>
            <Error>JobError.txt</Error>
            <Output>JobOutput.txt</Output>
        </Application>
        <Resources>
            <QueueName>creamtest2</QueueName>
        </Resources>

        <DataStaging>

          <ClientDataPush>true</ClientDataPush>

          <OutputFile>
            <Name>JobError.txt</Name>
            <Target>
              <URI>gsiftp://cream-23.pd.infn.it//tmp/JobError.txt</URI>
              <DelegationID>04352064184726456</DelegationID>
            </Target>
          </OutputFile>
          <OutputFile>
            <Name>JobOutput.txt</Name>
            <Target>
              <URI>gsiftp://cream-23.pd.infn.it//tmp/JobOutput.txt</URI>
              <DelegationID>04352064184726456</DelegationID>
            </Target>
          </OutputFile>
        </DataStaging>
```

```
    </ActivityDescription>
</CreateActivities>
```

When the user submits this ADL the CE returns a particular status attribute saying that the CE is waiting for the user action (sending a file):

```
$ glite-es-activity-create --endpoint cream-52.pd.infn.it ~/JDLs/activity_files_push.adl
****************************************
ActivityID     = CR_ES510328951
ActivityMgrURI = https://cream-52.pd.infn.it:8443/ce-cream-es/services/ActivityManagementService
Status         = PREPROCESSING
Status Attrs   = {CLIENT_STAGEIN_POSSIBLE}
Timestamp      = Tue Mar 27 09:22:05 2012
Description    =
ETNSC          =
STAGEIN Dir    = {gsiftp://cream-52.pd.infn.it/var/cream_es_sandbox/dteam/CN_Alvise_Dorigo_L_Pado
SESSION Dir    = {}
STAGEOUT Dir   = {gsiftp://cream-52.pd.infn.it/var/cream_es_sandbox/dteam/CN_Alvise_Dorigo_L_Pado
```

We've already seen above that the user will have to send the **`myjob.sh`** executable into the "STAGEIN Dir" and then notify the CE:

```
$ glite-es-notify-service -e cream-52.pd.infn.it CR_ES510328951:CLIENT-DATAPUSH-DONE
```

The syntax of the **`glite-es-notify-service`** command is simple: the endpoint is the same for all the other commands; the argument **`CR_ES510328951:CLIENT-DATAPUSH-DONE`** is simply the **`<Activity_Identifier>:<ACTION_TO_NOTIFY>`**. The **`<Activity_Identifier>`** is to indicate which activity in the CE is affected by the action performed by the user; the **`<ACTION_TO_NOTIFY>`** can have at the moment only these two values:

- CLIENT-DATAPUSH-DONE
- CLIENT-DATAPULL-DONE

We've just seen the meaning of the former. The CLIENT-DATAPULL-DONE is to be sent when the activity is terminated, and output files, available on the CE in the "STAGEOUT Dir", have been retrieved by the user.

Creating and handling a delegation

A delegation (or delegated proxy) is a temporary proxy certificate created by the server and signed by the user with his/her personal certificate. This delegation resides on the server and is used by ES to perform operations that require authentication. Then, these operations will be performed on behalf of the user that signed the delegation. Actually the only operations that need a delegation are file transmission to/from storage elements that require authentication. In fact a delegation must be specified in the ADL only if there're Intput/Output sandboxes to move around that need authentication to get access (see above). If the sandboxes do not need authentication (e.g. they're available through "simple" HTTP protocol) the delegation is not needed.

A delegation is always associated to a delegation identifier string (1:1). The delegation identifier is the handle the user can use in the ADL, or as argument to ask information about the related delegation or to renew it.

Creating a delegation

The command to create a delegation is **`glite-es-delegate-proxy`**; it requires only one argument: the endpoint; it returns the delegation identifier string:

```
$ glite-es-delegate-proxy cream-52.pd.infn.it
```

```
DelegationID = 8070042623506658
```

The user's proxy must be valid before to create a delegation otherwise an error message like this will occur:

```
$ glite-es-delegate-proxy cream-05.pd.infn.it
There's a problem with proxyfile [/tmp/x509up_u501]: The proxy has EXPIRED!
```

---+++ Asking information about a delegation with the command **`glite-es-delegation-info`** the user can ask for delegation's information:

```
$ glite-es-delegation-info -e cream-52.pd.infn.it 8070042623506658
Lifetime = Wed Mar 28 21:31:19 2012
Issuer   = CN=proxy,CN=proxy,CN=Alvise Dorigo,L=Padova,OU=Personal Certificate,O=INFN,C=IT
Subject  = CN=Alvise Dorigo,L=Padova,OU=Personal Certificate,O=INFN,C=IT
```

Delegation renew

If the user renewed recently his proxy, he/she can also renew the delegation residing on the CE machine, by mean of the command **`glite-es-delegation-renew`**:

```
dorigoa@lxgrid05 9:39:16 ~/emi/creamui_emi2>stage/usr/bin/glite-es-delegation-renew -e cream-52.p
Delegation with identifier [8070042623506658] successfully renewed
```

Advanced usage of commands

In this section we will describe the options the user can use to improve his/her usage of the ES client commands; they can apply to all the commands.

- **`--debug|-d`** activates a highly verbose output on the console. Please take into account that this output is only for debugging purposes and could change with future updates
- **`--proxyfile|-p <alternate_proxyfile>`** instructs the command to use the file **`<alternate_proxyfile>`** as proxy file instead of the default one (**`/tmp/x509up_u<USER_UNIX_ID>`**)
- **`--timeout|-t N`** sets the SOAP's connection timeout to N seconds (the default is 30 seconds); when the CLI is connecting to (and exchanging data with) a remote ES service, if the transmission hungs up for a time greater than the SOAP connection timeout, the command will exit with an error message
- **`--certfile|-c <user_certificate_file>`** and **`--keyfile|-k <user_key_file>`** set the couple certificate/key to use for authentication (instead of the proxyfile). Some ES servers do not support yet (and probably will not support never) the proxy certificates. These 2 options must be use together or not at all; they are mutually exclusive with the **`--proxyfile`** option.
- **`--donot-verify-ac-sign|-A`** disable the certification authority check

Modifying the connection EPR

Any WebService is reachable at a certain endpoint (hostname and TCPPORT) and EPR (EndPoint Reference) like this:

```
https://cream-52.pd.infn.it:8443/ce-cream-es/services/CreationService
```

`https://` is the protocol, **`cream-52.pd.infn.it:8443`** is the endpoint (as explained above), **`/ce-cream-es/services/CreateActivityService`** is the EPR. A WebService like CREAM/ES can publish more operations (activity creation, activity status, activity list, etc.); different operations can be linked to different EPRs (but same protocol/endpoint). In the case of CREAM/ES this is the mapping:

- Activity creation: **/ce-cream-es/services/CreationService**
- Activity status, cancel, pause, resume, restart, wipe, info: **/ce-cream-es/services/ActivityManagementService**
- Activity list: **/ce-cream-es/services/ActivityInfoService**
- Delegation operations: **/ce-cream-es/services/DelegationService**

These EPRs in principle can differ from the EPRs of another ES implementation, then the ES command line has the possibility to specify a different EPR for a certain operation. In the following example we create an activity to a CREAM/ES implementation (cream-10.pd.infn.it) and to a Unicore/ES implementation (zam052v02.zam.kfa-juelich.de:8080) specifying two different EPRs; note that we will use the **`-d`** (**`--debug`**) option in order to print extra information (e.g. the complete address to contact, including the EPR):

```
$ glite-es-activity-create --endpoint zam052v02.zam.kfa-juelich.de:8080 --epr /EMI-ES/services/Cr
Parsing XML file [/home/dorigoa/JDLs/activity_sleep120.adl] and converting to SOAP objects...
There're 1 activity/ies to submit
Sending CreateActivity request to service [https://zam052v02.zam.kfa-juelich.de:8080/EMI-ES/servi
******************************************
ActivityID     = 5c9b6f1e-1a3b-409f-aea4-d550c57bf155
ActivityMgrURI = https://zam052v02.zam.kfa-juelich.de:8080/EMI-ES/services/ActivityManagementServ
Status         = ACCEPTED
Status Attrs   = {CLIENT_STAGEIN_POSSIBLE}
Timestamp      = Thu Jan  1 01:00:00 1970
Description    =
ETNSC          =
STAGEIN Dir    = {gsiftp://zam052v07.zam.kfa-juelich.de:2811/home15/unicore/FILESPACE/5c9b6f1e-1a
SESSION Dir    = {}
STAGEOUT Dir   = {}

$ glite-es-activity-create --endpoint cream-10.pd.infn.it --epr /ce-cream-es/services/CreationSer
Parsing XML file [/home/dorigoa/JDLs/activity_sleep120.adl] and converting to SOAP objects...
There're 1 activity/ies to submit
Sending CreateActivity request to service [https://cream-10.pd.infn.it:8443/ce-cream-es/services/
******************************************
ActivityID     = CR_ES569960280
ActivityMgrURI = https://cream-10.pd.infn.it:8443/ce-cream-es/services/ActivityManagementService
Status         = PREPROCESSING
Status Attrs   = {}
Timestamp      = Fri Mar 30 09:57:14 2012
Description    =
ETNSC          =
STAGEIN Dir    = {gsiftp://cream-10.pd.infn.it/var/cream-es_sandbox/dteam/CN_Alvise_Dorigo_L_Pado
SESSION Dir    = {}
STAGEOUT Dir   = {gsiftp://cream-10.pd.infn.it/var/cream-es_sandbox/dteam/CN_Alvise_Dorigo_L_Pado
```

Note the different EPRs used for the two activity creations. The ES UI has defaults built-in for all the EPRs needed to contact a CREAM/ES implementation (in fact, this guide refers to the CREAM/ES UI, conceived mainly as CREAM/ES client, but able to 'talk' also to other ES implementations thanks to the standardized ES interface).

Also note that for the Unicore/ES implementation (that does not support yet the proxy certificates) we had to specify our certificate and key files with the corresponding options described above.

In the following example we will ask for activity's status on both CREAM/ES and Unicore/ES implementations using, again, different EPRs:

```
$ glite-es-activity-status --endpoint zam052v02.zam.kfa-juelich.de:8080 --epr /EMI-ES/services/Ac
Sending ActivityStatus request for ActivityID(s) {5c9b6f1e-1a3b-409f-aea4-d550c57bf155} to servic
******************************************
ActivityID  = 5c9b6f1e-1a3b-409f-aea4-d550c57bf155
Status      = TERMINAL
```

```
Attributes  = {CLIENT_STAGEOUT_POSSIBLE}
Timestamp   = Fri Mar 30 10:00:34 2012
Description =

$ glite-es-activity-status --endpoint cream-10.pd.infn.it CR_ES778754308 -d
Sending ActivityStatus request for ActivityID(s) {CR_ES778754308} to service [https://cream-10.pd
****************************************
ActivityID  = CR_ES778754308
Status      = TERMINAL
Attributes  = {}
Timestamp   = Fri Mar 30 09:49:16 2012
Description = reason=0
```

Note that in this last example, for the status request to cream-10.pd.infn.it (CREAM/ES implementation) we omitted the **`--epr`** option and the command used the (correct) built-in EPR value (**`/ce-cream-es/services/ActivityManagementService`**). If we try to get activity status on the Unicore/ES and we forget the EPR specification, the built-in will be used and an error message will be returned by Unicore:

```
$ glite-es-activity-status --endpoint zam052v02.zam.kfa-juelich.de:8080 -c ~/.globus/usercert.pem
Sending ActivityStatus request for ActivityID(s) {5c9b6f1e-1a3b-409f-aea4-d550c57bf155} to servic
Received NULL fault; the error is due to another cause:  FaultString=[Cannot send message to http
```

-->

-- MassimoSgaravatto - 2011-04-07

-- MassimoSgaravatto - 2012-03-14

-- AlviseDorigo - 2012-03-30

This topic: CREAM > UserGuideEMI2
Topic revision: r25 - 2013-06-07 - AlviseDorigo