# Table of Contents

# Table of Contents

# Table of Contents

# System Administrator Guide for CREAM for EMI-1 release

# 1 Installation and Configuration

## 1.1 Prerequisites

### 1.1.1 Operating system

A standard 64 bit SL(C)5 distribution is supposed to be properly installed.

### 1.1.2 Node synchronization

A general requirement for the Grid nodes is that they are synchronized. This requirement may be fulfilled in several ways. One of the most common one is using the `NTP` protocol with a time server.

### 1.1.3 Cron and logrotate

Many components deployed on the CREAM CE rely on the presence of `cron` (including support for `/etc/cron.*` directories) and `logrotate`. You should make sure these utils are available on your system.

### 1.1.4 Batch system

If you plan to use Torque as batch system for your CREAM CE, it will be installed and configured along with the middleware (i.e. you don't have to install and configure it in advance)

If you plan to use LSF as batch system for your CREAM CE, you have to install and configure it before installing and configuring the CREAM software. Since LSF is a commercial software it can't be distributed together with the middleware.

If you plan to use GE as batch system for your CREAM CE, you have to install and configure it before installing and configuring the CREAM software. The CREAM CE integration was tested with GE 6.2u5 but it should work with any forked version of the original GE software. The support of the GE batch system software (or any of its forked versions) is out of the scope of this activity.

More information abut batch system integration is available in the relevant section .

## 1.2 Plan how to deploy the CREAM CE

### 1.2.1 CREAM CE and gLite-cluster

glite-CLUSTER is a node type that can publish information about clusters and subclusters in a site, referenced by any number of compute elements. In particular it allows to deal with sites having multiple CREAM CE nodes and/or multiple subclusters (i.e. disjoint sets of worker nodes, each set having sufficiently homogeneous properties).

In Glue1, batch system queues are represented through GlueCE objectclasses. Each GlueCE refers to a Cluster, which can be composed by one or more SubClusters. However the gLite WMS requires the publication of exactly one SubCluster per Cluster (and hence per batch queue).

Thus sites with heterogeneous hardware have two possible choices:

- publish a SubCluster with a representative/minimum hardware description (e.g. the minimum memory on any node)
- define separate batch queues for each hardware configuration, e.g. low/high memory queues, and attach the corresponding GlueCE objects to separate Cluster/SubCluster pairs. For attributes with discrete values, e.g. SL4 vs SL5, this second option is the only one which makes sense.

However, without the use of the gLite-cluster, YAIM allows configuring a single Cluster per CREAM head node.

A second problem, addressed by gLite-cluster arises for larger sites which install multiple CE headnodes submitting to the same batch queues for redundancy or load-balancing. Without the use of gLite-cluster, YAIM generates a separate Cluster/SubCluster pair for each head node even though they all describe the same hardware. This causes no problems for job submission, but by default would overcount the installed capacity at the site by a multiple of the number of SubClusters. The workaround, before introducing the gLite-cluster, was to publish zero values for the installed capacity from all but one of the nodes (but this is clearly far from being an ideal solution).

The glite-CLUSTER node addresses this issue. It contains a subset of the functionality incorporated in the CREAM node types: the publication of the Glue1 GlueCluster and its dependent objects, the publication of the Glue1 GlueService object for the GridFTP endpoint, and the directories which store the RunTimeEnvironment tags, together with the YAIM functions which configure them.

So, gLite-CLUSTER should be considered:

- if in the site there are multiple CE head nodes, and/or
- if in the site there are multiple disjoint sets of worker nodes, each set having sufficiently homogeneous properties

When configuring a gLite-cluster, please consider that there should be:

- There should be one cluster for each set of worker nodes having sufficiently homogeneous properties
- There should be one subcluster for each cluster
- Each batch system queue should refer to the WNs of a single subcluster

glite-CLUSTER can be deployed in the same host of a CREAM-CE or in a different one.

The following deployment models are possible for a CREAM-CE:

- CREAM-CE can be configured without worrying about the glite-CLUSTER node. This can be useful for small sites who don't want to worry about cluster/subcluster configurations because they have a very simple setup. In this case CREAM-CE will publish a single cluster/subcluster. This is called **no cluster mode**. This is done as described below by defining the yaim setting `CREAMCE_CLUSTER_MODE=no` (or by no defining at all that variable).
- CREAM-CE can work on **cluster mode** using the glite-CLUSTER node type. This is done as described below by defining the yaim setting `CREAMCE_CLUSTER_MODE=yes`. The CREAM-CE can be in the same host or in a different host wrt the glite-CLUSTER node.

More information about glite-CLUSTER can be found at
https://twiki.cern.ch/twiki/bin/view/LCG/CLUSTER and in this note .

Information concerning glue2 publication is available here .

Please also note that, because of these bugs:

http://savannah.cern.ch/bugs/?87690

http://savannah.cern.ch/bugs/?87691

in the current implementation when configuring via yaim, all queues of a given CE head node gets mapped to a single cluster (i.e. it not possible to map different queues to different clusters). Waiting for the fixes for these bugs, the ldif files produced by yaim must be then manually edited.

## 1.2.2 Define a DNS alias to refer to set of CREAM CEs

In order to distribute load for job submissions, it is possible to deploy multiple CREAM CEs head nodes referring to the same set of resources. As explained in the previous section, this should be implemented with:

- a gLite-CLUSTER node
- multiple CREAM CEs configured in cluster mode

It is then also possible to define a DNS alias to refer to the set of CREAM head nodes: after the initial contact from outside clients to the CREAM-CE alias name for job submission, all further actions on that job are based on the jobid which contains the physical hostname of the CREAM-CE to which the job was submitted. This allows to switch the DNS alias in order to distribute load.

The alias shouldn't be published in the information service, but should be simply communicated to the relevant users.

There are various techniques to change an alias entry in the DNS. The choice depends strongly on the way the network is set up and managed. For example at DESY a self-written service called `POISE` is used; using metrics (which take into account in particular the load and the sandbox size ) it decides the physical instance the alias should point to. Another possibility to define aliases is to use commercial network techniques such as F5.

It must be noted that, as observed by Desy sysadmins, the proliferation of the aliases (C-records) is not well defined among DNS'. Therefore changes of an alias sometimes can take hours to be propagated to other sites.

The use of alias for job submission is a good solution to improve load balancing and availability of the service (the unavailability of a physical CREAM CE would be hidden by the use of the alias). It must however be noted that:

- The list operation ( `glite-ce-job-list` command of the CREAM CLI) issued on a alias returns the identifiers of the jobs submitted to the physical instance currently pointed to the alias, and not the identifiers of all the jobs submitted to all CREAM CEs instances
- The operations to be done on all jobs (e.g. cancel all jobs, return the status of all jobs, etc.), i.e. the ones issued using the options `-a -e` of the CREAM CLI, issued on a alias, refer to just the CREAM physical instance currently pointed by the alias (and not to all CREAM CE instances).
- The use of alias is not supported for submissions through the gLite-WMS and CondorG

## 1.2.3 Choose the authorization model

The CREAM CE can be configured to use as authorization system:

- the ARGUS authorization framework

OR

- the grid Java Authorization Framework (gJAF)

In the former case a ARGUS box (recommended to be at site level: it can of course serve multiple CEs of that site) where to define policies for the CREAM CE box is needed.

To use ARGUS as authorization system, yaim variable `USE_ARGUS` must be set in the following way:

```
USE_ARGUS=yes
```

In this case it is also necessary to set the following yaim variables:

- `ARGUS_PEPD_ENDPOINTS` The endpoint of the ARGUS box (e.g."https://cream-43.pd.infn.it:8154/authz")
- `CREAM_PEPC_RESOURCEID` The id of the CREAM CE in the ARGUS box (e.g. "http://pd.infn.it/cream-18")

If instead gJAF should be used as authorization system, yaim variable `USE_ARGUS` must be set in the following way:

```
USE_ARGUS=no
```

## 1.2.4 Choose the BLAH BLparser deployment model

The BLAH Blparser is the component of the CREAM CE responsible to notify CREAM about job status changes.

For LSF and PBS/Torque it is possible to configure the BLAH blparser in two possible ways:

- The new BLAH BLparser, which relies on the status/history batch system commands
- The old BLAH BLparser, which parses the batch system log files

For GE and Condor, only the configuration with the new BLAH blparser is possible

### 1.2.4.1 New BLAH Blparser

The new Blparser runs on the CREAM CE machine and it is automatically installed when installing the CREAM CE. The configuration of the new BLAH Blparser is done when configuring the CREAM CE (i.e. it is not necessary to configure the Blparser separately from the CREAM CE).

To use the new BLAH blparser, it is just necessary to set:

```
BLPARSER_WITH_UPDATER_NOTIFIER=true
```

in the siteinfo.def and then configure the CREAM CE. This is the default value.

The new BLParser doesn't parse the log files. However the `bhist` (for LSF) and `tracejob` (for Torque) commands (used by the new BLParser) require the batch system log files, which therefore must be available (in case e.g. via NFS in the CREAM CE node. Actually for Torque the blparser uses `tracejob` (which requires the log files) only when `qstat` can't find anymore the job. And this can happen if the job has been completed more than `keep_completed` seconds ago and the blparser was not able to detect before that the job completed/was cancelled/whatever. This can happen e.g. if keep_completed is too short or if the BLAH blparser for whatever reason didn't run for a while. If the log files are not available and the `tracejob` command is issued (for the reasons specified above), the BLAH blparser will not be able to find the job, which will considered "lost" (DONE-FAILED wrt CREAM).

The init script of the new Blparser is `/etc/init.d/glite-ce-blahparser`. Please note that it is not needed to explicitly start the new blparser: when CREAM is started, it starts also this new BLAH Blparser if it

is not already running.

When the new Blparser is running, you should see the following two processes on the CREAM CE node:

- `/usr/bin/BUpdaterxxx`
- `/usr/bin/BNotifier`

Please note that the user `tomcat` on the CREAM CE should be allowed to issue the relevant status/history commands (for Torque: `qstat`, `tracejob`, for LSF: `bhist`, `bjobs`). Some sites configure the batch system so that users can only see their own jobs (e.g. in torque:

```
set server query_other_jobs = False
```

). If this is done at the site, then the tomcat user will need a special privilege in order to be exempt from this setting (in torque:

```
set server operators += tomcat@creamce.yoursite.domain
```

).

#### 1.2.4.2 Old BLAH Blparser

The old BLAH blparser must be installed on a machine where the batch system log files are available (let's call this host `BLPARSER_HOST`. So the BLPARSER_HOST can be the batch system master or a different machine where the log files are available (e.g. they have been exported via NFS). There are two possible layouts:

- The BLPARSER_HOST is the CREAM CE host
- The BLPARSER_HOST is different than the CREAM CE host

If the BLPARSER_HOST is the CREAM CE host, after having installed and configured the CREAM CE, it is necessary to configure the old BLAH Blparser (as explained below) and then to restart tomcat.

If the BLPARSER_HOST is different than the CREAM CE host, after having installed and configured the CREAM CE it is necessary:

- to install the old BLAH BLparser software on this BLPARSER_HOST as explained below
- to configure thie old BLAH BLparser
- to restart tomcat on the CREAM-CE

After having configured CREAM, it is necessary to also configure the BLAH Blparser as explained below.

On the CREAM CE, to use the old BLAH blparser, it is necessary to set:

```
BLPARSER_WITH_UPDATER_NOTIFIER=false
```

in the siteinfo.def before configuring via yaim.

## 1.2.5 Deployment models for CREAM databases

The databases used by CREAM can be deployed in the CREAM CE host (which is the default scenario) or on a different machine.

Click here  for information how to deploy the databases on a machine different wrt the CREAM-CE.

# 1.3 CREAM CE Installation

This section explains how to install:

- a CREAM CE in no cluster mode
- a CREAM CE in cluster mode
- a glite-CLUSTER node

For all these scenarios, the setting of the repositories is the same.

## 1.3.1 Repositories

For a successful installation, you will need to configure your package manager to reference a number of repositories (in addition to your OS);

- the EPEL repository
- the EMI middleware repository
- the CA repository

and to **REMOVE (!!!)** or **DEACTIVATE (!!!)**

- the DAG repository

### 1.3.1.1 The EPEL repository

You can install the EPEL repository, issuing:

```
rpm -Uvh http://dl.fedoraproject.org/pub/epel/5/i386/epel-release-5-4.noarch.rpm
```

### 1.3.1.2 The EMI middleware repository

You can install the EMI-1 yum repository, issuing:

```
wget http://emisoft.web.cern.ch/emisoft/dist/EMI/1/sl5/x86_64/updates/emi-release-1.0.1-1.sl5.noa
yum install ./emi-release-1.0.1-1.sl5.noarch.rpm
```

### 1.3.1.3 The Certification Authority repository

The most up-to-date version of the list of trusted Certification Authorities (CA) is needed on your node. The relevant yum repo can be installed issuing:

```
wget http://repository.egi.eu/sw/production/cas/1/current/repo-files/EGI-trustanchors.repo -O /et
```

### 1.3.1.4 Important note on automatic updates

An update of an RPM not followed by configuration can cause problems. Therefore **WE STRONGLY RECOMMEND NOT TO USE AUTOMATIC UPDATE PROCEDURE OF ANY KIND**.

Running the script available at http://forge.cnaf.infn.it/frs/download.php/101/disable_yum.sh (implemented by Giuseppe Platania (INFN Catania) yum autoupdate will be disabled

## 1.3.2 Installation of a CREAM CE node in no cluster mode

First of all, install the `yum-protectbase` rpm:

```
yum install yum-protectbase
```

Then proceed with the installation of the CA certificates.

### 1.3.2.1 Installation of the CA certificates

The CA certificate can be installed issuing:

```
yum install ca-policy-egi-core
```

### 1.3.2.2 Installation of the CREAM CE software

The CREAM software itself can work with both Sun jdk and openjdk. However, the apel-core package, deployed in the CREAM CE node, requires `mm-mysql`, which explicitly requires Sun jdk. So to install the middleware software needed for the CREAM CE, install first of all Sun JDK ( `jdk`). This is actually not needed in a standard SL5 box, since in this case the Sun JDK rpm is available in the OS repo. Please note that this dependency on `mm-mysql` is being fixed by APEL developers.

Then install `xml-commons-apis`:

```
yum install xml-commons-apis java-1.6.0-openjdk-devel
```

This is due to a dependency problem within the Tomcat distribution

Then install the CREAM-CE metapackage:

```
yum install emi-cream-ce
```

### 1.3.2.3 Installation of the batch system specific software

After the installation of the CREAM CE metapackage it is necessary to install the batch system specific metapackage(s):

- If you are running Torque, and your CREAM CE node is the torque master, install the `emi-torque-server` and `emi-torque-utils` metapackages:

```
yum install emi-torque-server
yum install emi-torque-utils
```

- If you are running Torque, and your CREAM CE node is NOT the torque master, install the `emi-torque-utils` metapackage:

```
yum install emi-torque-utils
```

- If you are running LSF, install the `emi-lsf-utils` metapackage:

```
yum install emi-lsf-utils
```

- If you are running GE, install the emi-ge-utils metapackage:

```
yum install emi-ge-utils
```

## 1.3.3 Installation of a CREAM CE node in cluster mode

First of all, install the `yum-protectbase` rpm:

```
yum install yum-protectbase
```

Then proceed with the installation of the CA certificates.

### 1.3.3.1 Installation of the CA certificates

The CA certificate can be installed issuing:

```
yum install ca-policy-egi-core
```

### 1.3.3.2 Installation of the CREAM CE software

The CREAM software itself can work with both Sun jdk and openjdk. However, the apel-core package, deployed in the CREAM CE node, requires `mm-mysql`, which explicitly requires Sun jdk. So to install the middleware software needed for the CREAM CE, install first of all Sun JDK ( `jdk`). This is actually not needed in a standard SL5 box, since in this case the Sun JDK rpm is available in the OS repo. Please note that this dependency on `mm-mysql` is being fixed by APEL developers.

Then install `xml-commons-apis`:

```
yum install xml-commons-apis java-1.6.0-openjdk-devel
```

This is due to a dependency problem within the Tomcat distribution

Then install the CREAM-CE metapackage:

```
yum install emi-cream-ce
```

### 1.3.3.3 Installation of the batch system specific software

After the installation of the CREAM CE metapackage it is necessary to install the batch system specific metapackage(s).

- If you are running Torque, and your CREAM CE node is the torque master, install the `emi-torque-server` and `emi-torque-utils` metapackages:

```
yum install emi-torque-server
yum install emi-torque-utils
```

- If you are running Torque, and your CREAM CE node is NOT the torque master, install the `emi-torque-utils` metapackage:

```
yum install emi-torque-utils
```

- If you are running LSF, install the `emi-lsf-utils` metapackage:

```
yum install emi-lsf-utils
```

- If you are running GE, install the emi-ge-utils metapackage:

```
yum install emi-ge-utils
```

#### 1.3.3.4 Installation of the cluster metapackage

If the CREAM CE node has to host also the `glite-cluster`, install also this metapackage:

```
yum install emi-cluster
```

### 1.3.4 Installation of a glite-cluster node

First of all, install the `yum-protectbase` rpm:

```
  yum install yum-protectbase
```

Then proceed with the installation of the CA certificates.

#### 1.3.4.1 Installation of the CA certificates

The CA certificate can be installed issuing:

```
yum install ca-policy-egi-core
```

#### 1.3.4.2 Installation of the cluster metapackage

Install the glite-CLUSTER metapackage:

```
yum install emi-cluster
```

### 1.3.5 Installation of the BLAH BLparser

If the new BLAH Blparser must be used, there isn't anything to be installed for the BLAH Blparser (i.e. the installation of the CREAM-CE is enough).

This is also the case when the old BLAH Blparser must be used **AND** the BLPARSER_HOST is the CREAM-CE.

Only when the old BLAH Blparser must be used **AND** the BLPARSER_HOST is different than the CREAM-CE, it is necessary to install the BLParser software on this BLPARSER_HOST. This is done in the following way:

```
yum install glite-ce-blahp
yum install glite-ce-yaim-cream-ce
```

### 1.3.6 Installation of the CREAM CLI

The CREAM CLI is part of the EMI-UI. To install it please refer to https://twiki.cern.ch/twiki/bin/view/EMI/EMIui#Client_Installation_Configuratio .

## 1.4 CREAM CE update

To update the CREAM CE from EMI-1 Update x to EMI-1 Update y:

- Run `yum update`
- Reconfigure via yaim

# 1.5 CREAM CE configuration

## 1.5.1 Using the YAIM configuration tool

For a detailed description on how to configure the middleware with YAIM, please check the YAIM guide .

The necessary YAIM modules needed to configure a certain node type are automatically installed with the middleware.

## 1.5.2 Configuration of a CREAM CE node in no cluster mode

### 1.5.2.1 Install host certificate

The CREAM CE node requires the host certificate/key files to be installed. Contact your national Certification Authority (CA) to understand how to obtain a host certificate if you do not have one already.

Once you have obtained a valid certificate:

- hostcert.pem - containing the machine public key
- hostkey.pem - containing the machine private key

make sure to place the two files in the target node into the `/etc/grid-security` directory. Then set the proper mode and ownerships doing:

```
chown root.root /etc/grid-security/hostcert.pem
chown root.root /etc/grid-security/hostkey.pem
chmod 644 /etc/grid-security/hostcert.pem
chmod 400 /etc/grid-security/hostkey.pem
```

### 1.5.2.2 Configure the siteinfo.def file

Set your `siteinfo.def` file, which is the input file used by yaim. Documentation about yaim variables relevant for CREAM CE is available at
https://twiki.cern.ch/twiki/bin/view/LCG/Site-info_configuration_variables#cream_CE

Be sure that `CREAMCE_CLUSTER_MODE` is set to `no` (or not set at all).

### 1.5.2.3 Run yaim

After having filled the `siteinfo.def` file, run yaim:

```
/opt/glite/yaim/bin/yaim -c -s <site-info.def> -n creamCE -n <LRMSnode>
```

Examples:

- Configuration of a CREAM CE in no cluster mode using Torque as batch system, with the CREAM CE being also Torque server

```
/opt/glite/yaim/bin/yaim -c -s <site-info.def> -n creamCE -n TORQUE_server -n TORQUE_utils
```

- Configuration of a CREAM CE in no cluster mode using Torque as batch system, with the CREAM CE NOT being also Torque server

```
/opt/glite/yaim/bin/yaim -c -s <site-info.def> -n creamCE -n TORQUE_utils
```

- Configuration of a CREAM CE in no cluster mode using LSF as batch system

```
/opt/glite/yaim/bin/yaim -c -s <site-info.def> -n creamCE -n LSF_utils
```

- Configuration of a CREAM CE in no cluster mode using GE as batch system

```
/opt/glite/yaim/bin/yaim -c -s <site-info.def> -n creamCE -n SGE_utils
```

## 1.5.3 Configuration of a CREAM CE node in cluster mode

### 1.5.3.1 Install host certificate

The CREAM CE node requires the host certificate/key files to be installed. Contact your national Certification Authority (CA) to understand how to obtain a host certificate if you do not have one already.

Once you have obtained a valid certificate:

- hostcert.pem - containing the machine public key
- hostkey.pem - containing the machine private key

make sure to place the two files in the target node into the `/etc/grid-security` directory. Then set the proper mode and ownerships doing:

```
chown root.root /etc/grid-security/hostcert.pem
chown root.root /etc/grid-security/hostkey.pem
chmod 600 /etc/grid-security/hostcert.pem
chmod 400 /etc/grid-security/hostkey.pem
```

### 1.5.3.2 Configure the siteinfo.def file

Set your `siteinfo.def` file, which is the input file used by yaim.

Variables which are required in cluster mode are described below.

When the CREAM CE is configured in cluster mode it will stop publishing information about clusters and subclusters. That information should be published by the glite-CLUSTER node type instead. A specific set of yaim variables has been defined for configuring the information which is still required by the CREAM CE in cluster mode. The names of these variables follow this syntax:

- In general, variables based on hostnames, queues or VOViews containing '.' and '_' # should be transformed into '-'
- <host-name>: identifier that corresponds to the CE hostname in lower case. Example: ctb-generic-1.cern.ch -> ctb_generic_1_cern_ch
- <queue-name>: identifier that corresponds to the queue in upper case. Example: dteam -> DTEAM
- <voview-name>: identifier that corresponds to the VOView id in upper case. '/' and '=' should also be transformed into '_'. Example: /dteam/Role=admin -> DTEAM_ROLE_ADMIN

These new variables are:

| Variable Name | Description |
|---|---|
| `CREAM_CLUSTER_MODE` | Defines if the CE shou<br>mode. Default value is<br>be `yes` for cluster mod |
| `CE_HOST_<host-name>_CE_TYPE` | CE type: should be 'cre |

| | |
|---|---|
| CE_HOST_<host-name>_QUEUES | Space separated list of the CE. The old variab cluster mode) was: QU |
| CE_HOST_<host-name>_QUEUE_<queue-name>_CE_AccessControlBaseRule | Space separated list of which are allowed to a the CE. The old variab cluster mode) was: _G |
| CE_HOST_<host-name>_CE_InfoJobManager | The name of the job m The old variable name mode) was: JOB_MAN sge or condor |
| CE_HOST_<host-name>_CLUSTER_UniqueID | UniqueID of the cluste be obsoleted by QUEUE_<queue-na |
| QUEUE_<queue-name>_CLUSTER_UniqueID | UniqueID of the cluste obsolete CE_HOST_<host-n |
| CE_InfoApplicationDir | Prefix of the experimen The old variable name mode) was: VO_SW_D defined per CE, queue, |

Be sure that CREAMCE_CLUSTER_MODE is set to yes

#### 1.5.3.3 Run yaim

After having filled the siteinfo.def file, run yaim:

```
/opt/glite/yaim/bin/yaim -c -s <site-info.def> -n creamCE -n <LRMSnode> [-n glite-CLUSTER]
```

-n glite-CLUSTER must be specified only if the glite-CLUSTER is deployed in the same node of the CREAM-CE

Examples:

- Configuration of a CREAM CE in cluster mode (with glite-CLUSTER deployed on a different node) using LSF as batch system

```
/opt/glite/yaim/bin/yaim -c -s <site-info.def> -n creamCE -n LSF_utils
```

- Configuration of a CREAM CE in cluster mode (with glite-CLUSTER deployed on a different node) using GE as batch system

```
/opt/glite/yaim/bin/yaim -c -s <site-info.def> -n creamCE -n SGE_utils
```

- Configuration of a CREAM CE in cluster mode (with glite-CLUSTER deployed on a different node) using Torque as batch system, with the CREAM CE being also Torque server

```
/opt/glite/yaim/bin/yaim -c -s <site-info.def> -n creamCE -n TORQUE_server -n TORQUE_utils
```

- Configuration of a CREAM CE in cluster mode (with glite-CLUSTER deployed on a different node) using Torque as batch system, with the CREAM CE NOT being also Torque server

```
/opt/glite/yaim/bin/yaim -c -s <site-info.def> -n creamCE -n TORQUE_utils
```

- Configuration of a CREAM CE in cluster mode (with glite-CLUSTER deployed on the same node of the CREAM-CE) using LSF as batch system

```
/opt/glite/yaim/bin/yaim -c -s <site-info.def> -n creamCE -n LSF_utils -n glite-CLUSTER
```

- Configuration of a CREAM CE in cluster mode (with glite-CLUSTER deployed on the same node of the CREAM-CE) using GE as batch system

```
/opt/glite/yaim/bin/yaim -c -s <site-info.def> -n creamCE -n SGE_utils -n glite-CLUSTER
```

- Configuration of a CREAM CE in cluster mode (with glite-CLUSTER deployed on the same node of the CREAM-CE) using Torque as batch system, with the CREAM CE being also Torque server

```
/opt/glite/yaim/bin/yaim -c -s <site-info.def> -n creamCE -n TORQUE_server -n TORQUE_utils
```

- Configuration of a CREAM CE in cluster mode (with glite-CLUSTER deployed on the same node of the CREAM-CE)) using Torque as batch system, with the CREAM CE NOT being also Torque server

```
/opt/glite/yaim/bin/yaim -c -s <site-info.def> -n creamCE -n TORQUE_utils -n glite-CLUSTER
```

## 1.5.4 Configuration of a glite-CLUSTER node

### 1.5.4.1 Install host certificate

The glite-CLUSTER node requires the host certificate/key files to be installed. Contact your national Certification Authority (CA) to understand how to obtain a host certificate if you do not have one already.

Once you have obtained a valid certificate:

- hostcert.pem - containing the machine public key
- hostkey.pem - containing the machine private key

make sure to place the two files in the target node into the `/etc/grid-security` directory. Then set the proper mode and ownerships doing:

```
chown root.root /etc/grid-security/hostcert.pem
chown root.root /etc/grid-security/hostkey.pem
chmod 600 /etc/grid-security/hostcert.pem
chmod 400 /etc/grid-security/hostkey.pem
```

### 1.5.4.2 Configure the siteinfo.def file

Set your `siteinfo.def` file, which is the input file used by yaim. Documentation about yaim variables relevant for glite-CLUSTER is available at
https://twiki.cern.ch/twiki/bin/view/LCG/Site-info_configuration_variables#CLUSTER

### 1.5.4.3 Run yaim

After having filled the `siteinfo.def` file, run yaim:

```
/opt/glite/yaim/bin/yaim -c -s <site-info.def> -n glite-CLUSTER
```

## 1.5.5 Configuration of the BLAH Blparser

If the new BLAH Blparser must be used, there isn't anything to be configured for the BLAH Blparser (i.e. the configuration of the CREAM-CE is enough).

If the old BLparser must be used, it is necessary to configure it on the BLPARSER_HOST (which, as said above, can be the CREAM-CE node or on a different host). This is done in the following way:

```
/opt/glite/yaim/bin/yaim -r -s <site-info.def> -n creamCE -f config_cream_blparser
```

Then it is necessary to restart tomcat on the CREAM-CE node:

```
service tomcat5 restart
```

#### 1.5.5.1 Configuration of the old BLAH Blparser to serve multiple CREAM CEs

The configuration instructions reported above explains how to configure a CREAM CE and the BLAH blparser (old model) considering the scenario where the BLAH blparser has to "serve" a single CREAM CE.

Considering that the blparser (old model) has to run where the batch system log files are available, let's consider a scenario where there are 2 CREAM CEs ( `ce1.mydomain` and `ce2.mydomain`) that must be configured. Let's suppose that the batch system log files are not available on these 2 CREAM CEs machine. Let's assume they are available in another machine ( `blhost.mydomain`), where the old blparser has to be installed.

The following summarizes what must be done:

- In the `/services/glite-creamce` for `ce1.mydomain` set:

```
BLPARSER_HOST=blhost.mydomain
BLAH_JOBID_PREFIX=cre01_
BLP_PORT=33333
```

and configure `ce1.mydomain` via yaim:

```
/opt/glite/yaim/bin/yaim -c -s <site-info.def> -n creamCE -n <LRMSnode> [-n glite-CLUSTER]
```

- In the `/services/glite-creamce` for `ce2.mydomain` set:

```
BLPARSER_HOST=blhost.mydomain
BLAH_JOBID_PREFIX=cre02_
BLP_PORT=33334
```

and configure `ce2.mydomain` via yaim:

```
/opt/glite/yaim/bin/yaim -c -s <site-info.def> -n creamCE -n <LRMSnode> [-n glite-CLUSTER]
```

- In the `/services/glite-creamce` for `blhost.mydomain` sets:

```
CREAM_PORT=56565
```

and configure `blhost.mydomain` via yaim:

```
/opt/glite/yaim/bin/yaim -r -s <site-info.def> -n creamCE -f config_cream_blparser
```

- In `blhost.mydomain` edit the file `/etc/blparser.conf` setting (considering the pbs/torque scenario):

```
GLITE_CE_BLPARSERPBS_NUM=2

# ce01.mydomain
GLITE_CE_BLPARSERPBS_PORT1=33333
GLITE_CE_BLPARSERPBS_CREAMPORT1=56565
```

```
# ce02.mydomain
GLITE_CE_BLPARSERPBS_PORT2=33334
GLITE_CE_BLPARSERPBS_CREAMPORT2=56566
```

- Restart the blparser on `blhost.mydomain`:

```
/etc/init.d/glite-ce-blparser restart
```

- Restart tomcat on `ce01.mydomain` and `ce02.mydomain`

You can of course replace 33333, 33334, 56565, 56566 (reported in the above examples) with other port numbers

#### 1.5.5.2 Configuration of the new BLAH Blparser to to use cached batch system commands

With BLAH version >= 1.16-3, the new BLAH blparser can be configured in order to not interact directly with the batch system, but through a program (to be implemented by the site admin) which can implement some caching functionality. This is the case for example of `CommandProxyTools`, implement at Cern

To enable this feature, add in `/etc/blah.config` (the example below is for lsf, with `/usr/bin/runcmd.pl` as name of the "caching" program):

```
lsf_batch_caching_enabled=yes
batch_command_caching_filter=/usr/bin/runcmd.pl
```

So the blparser, insead of issuing `bjobs -u ....`, will issue `/usr/bin/runcmd.pl bjobs -u ...`" </verbatim>

## 1.5.6 Configuration of the CREAM databases on a host different than the CREAM-CE

To configure the CREAM databases on a host different than the CREAM-CE:

- Set in the siteinfo.def file the variable `CREAM_DB_HOST` to the remote host (where mysql must be already installed)
- Set in the siteinfo.def file the variable `MYSQL_PASSWORD` considering the mysql password of the remote host
- On this remote host, grant appropriate privs to `root@CE_HOST`
- Configure via yaim

Please note that, because of bug #88078 , the name of the databases are hardwired in some files to `creamdb` and `delegationdb`. This means that some manual hacks are needed if the same mysql server host should be used to host the databases of multiple CREAM CEs.

The files where the database names are specified and should be changed before configuring via yaim are:

- `/etc/glite-ce-cream/ce-cream.xml` (which is the template used by yaim to create `$CATALINA_HOME/conf/Catalina/localhost/ce-cream.xml`. In the following lines `creamdb` and `delegationdb` should be replaced with the new names:

```
 url="jdbc:mysql://localhost:3306/creamdb"
```

and:

```
url="jdbc:mysql://localhost:3306/delegationdb?autoReconnect=true"
```

You must NOT change in the following line:

```
<Resource name="jdbc/creamdb"
```

- /etc/glite-ce-cream/createAndPopulateDB.sh. In the following line creamdb should be replaces with the chosen db name:

```
 if [ $3 == "creamdb" -a -d /opt/glite/var/cream_sandbox ] ; then
```

- /etc/glite-ce-cream/populate_creamdb_mysql.sql. In the following lines creamdb should be replaces with the chosen db name:

```
/************ Drop: Database ***************/
DROP DATABASE IF EXISTS creamdb;

/************ Create: Database ***************/
CREATE DATABASE creamdb;

/************ Use: Database ***************/
USE creamdb;
```

- /etc/glite-ce-cream/populate_delegationdb.sql. in the following lines delegationdb should be replaced with the chosen db name :

```
/************ Drop: Database ***************/
DROP DATABASE IF EXISTS delegationdb;

/************ Create: Database ***************/
CREATE DATABASE delegationdb;

/************ Use Delegation DB ***************/

use delegationdb;
```

- /usr/bin/glite_cream_load_monitor. In the following line creamdb should be replaced with the chosen db name:

```
 my $querycmd= "mysql -B --skip-column-names -u" . $userdb . " --password=\"" . $passworddb . "\"
```

- /etc/glite-ce-cream/cream-config.xml.template (which is used by yaim to generate /etc/glite-ce-cream/cream-config.xml). In this file add:

```
<cream db name>_database_version="2.4"
<delegation db name>_database_version="2.4"
```

just after:

```
creamdb_database_version="2.4"
delegationdb_database_version="2.4"
```

- /opt/glite/yaim/functions/config_cream_db. In the following lines, creamdb and delegationdb should be replaced with the chosen db names:

```
  create_mysql_db creamdatabase ${CREAM_DB_USER} ${CREAM_DB_PASSWORD}  "${GLITE_CREAM_LOCATION_ET
    && create_mysql_db delegationdatabase ${CREAM_DB_USER} ${CREAM_DB_PASSWORD} "${GLITE_CREAM_LO
```

- /opt/glite/yaim/functions/config_cream_ce. In the following lines, creamdb and delegationdb should be replaced with the chosen db names:

```
-e "s/url=\"jdbc:mysql:\/\/localhost:3306\/delegationdb?autoReconnect=true/url=\"jdbc:mysql:\/\/$
```

```
-e "s/url=\"jdbc:mysql:\/\/localhost:3306\/creamdb\"/url=\"jdbc:mysql:\/\/${CREAM_DB_HOST}:3306\/
```

- `/opt/glite/yaim/functions/config_cream_drop`. In the following lines, `creamdb` and `delegationdb` should be replaced with the chosen db names:

```
 mysqlshow --password="$MYSQL_PASSWORD" | grep "creamdatabase" > /dev/null 2>&1


mysql -u root --password="$MYSQL_PASSWORD" -e "DROP DATABASE creamdatabase"


mysqlshow --password="$MYSQL_PASSWORD" | grep "delegationdatabase" > /dev/null 2>&1


mysql -u root --password="$MYSQL_PASSWORD" -e "DROP DATABASE delegationdatabase"
```

## 1.5.7 Configuration of the CREAM CLI

The CREAM CLI is part of the EMI-UI. To configure it please refer to https://twiki.cern.ch/twiki/bin/view/EMI/EMIui#Client_Installation_Configuratio .

## 1.5.8 Manual configuration

yaim allows to choose the most important parameters (via yaim variables) related to the CREAM-CE. It is then possible to tune some other attributes manually editing the relevant configuration files.

The following subsections describe some of the parameters that can be manually configured.

Please note that:

- After having manually modified a configuration file, it is then necessary to restart the service
- Manual changes done in the configuration files are scratched by following yaim reconfigurations

### 1.5.8.1 Tune the number of concurrent BLAH instances

To allow a parallelism when interacting with the batch system, in particular to have a good throughput when submitting jobs to the batch system, CREAM can run multiple BLAH instances. A new instance is created whenever needed, till the maximum number defined in the `etc/glite-ce-cream/cream-config.xml` configuration file is reached. The relevant attribute is `cream_concurrency_level`. The default value is 50. This value should be usually fine. You might need to decrease it if you notice an overload of the batch system and many jobs aborted because the submission to the batch system failed with "send command timeout" error message.

Please note that with BLAH version < 1.16.3, there are some memory issues: each BLAH instance uses too much memory and therefore a high value of this parameter can cause memory problems. The problem is fixed with BLAH version 1.16.3. Besides this version of BLAH, it is also needed to have in `blah.config`:

```
job_registry_use_mmap=yes
```

which is automatically added by yaim (starting with yaim-cream-ce version >= 4.2.2-1).

Starting with yaim-cream-ce version >= 4.2.3-1, this value for the number of concurrent BLAH instances is configurable via yaim. The name of the relevant yaim variable is `CREAM_CONCURRENCY_LEVEL`.

#### 1.5.8.2 Tune the BLAH BUpdater polling frequency

If the new BLAH Blparser is used (click here for instructions to check if you are using the old or the new blparser) the parameter `bupdater_loop_interval` attribute in `/etc/blah.config` defines how often the batch system is queried to check the status of the jobs. If a low value is used, job status changes are detected promptly, but this also means that several batch system queries are done, and this can cause a high load.

With yaim-cream-ce v >= 4.2.2-1, this parameter is configurable via yaim: the relevant yaim variable is `BUPDATER_LOOP_INTERVAL` which has 30 (secs) as default value.

With yaim-cream-ce v. < 4.2.2-1, this parameter is not configurable via yaim, and therefore it is needed to manually edit the blah configuration file, otherwise the default value (5 s.) is used. After having set this value in the blah configuration file it is necessary:

- to stop tomcat: `service tomcat5 stop`
- to restart the blparser: `/etc/init.d/glite-ce-blahparser restart`
- to start tomcat: `service tomcat5 start`

In particular for LSF it is necessary that this value is not too low. As a general rule it is recommended to set it to 1.5 * (average of time needed to execute `bhist -u all -d -l -n 1`).

---

# 1.6 Batch system integration

## 1.6.1 Torque

### 1.6.1.1 Installation

If the CREAM-CE has to be also the torque server, install the `emi-torque-server` metapackage: `yum install emi-torque-server`

In all cases (Torque server in the CREAM-CE or in a different host) then install the `emi-torque-utils` metapackage: `yum install emi-torque-utils`

### 1.6.1.2 Configuration

Set your `siteinfo.def` file, which is the input file used by yaim. Documentation about yaim variables relevant for CREAM CE is available at CREAM CE:
https://twiki.cern.ch/twiki/bin/view/LCG/Site-info_configuration_variables#cream_CE

The CREAM CE Torque integration is then configured running YAIM:

- no cluster mode with CREAM-CE being also Torque server: `/opt/glite/yaim/bin/yaim -c -s <site-info.def> -n creamCE -n TORQUE_server -n TORQUE_utils`
- no cluster mode with CREAM-CE not being also Torque server: `/opt/glite/yaim/bin/yaim -c -s <site-info.def> -n creamCE -n TORQUE_utils`
- cluster mode with glite-CLUSTER deployed on a different node with CREAM-CE being also Torque server: `/opt/glite/yaim/bin/yaim -c -s <site-info.def> -n creamCE -n TORQUE_server -n TORQUE_utils`
- cluster mode with glite-CLUSTER deployed on a different node with CREAM-CE not being also Torque server: `/opt/glite/yaim/bin/yaim -c -s <site-info.def> -n creamCE -n TORQUE_utils`

- cluster mode with glite-CLUSTER deployed on the same node of the CREAM-CE with CREAM-CE being also Torque server : `/opt/glite/yaim/bin/yaim -c -s <site-info.def> -n creamCE -n TORQUE_server -n TORQUE_utils -n glite-CLUSTER`
- cluster mode with glite-CLUSTER deployed on the same node of the CREAM-CE with CREAM-CE not being also Torque server : `/opt/glite/yaim/bin/yaim -c -s <site-info.def> -n creamCE -n TORQUE_utils -n glite-CLUSTER`

## 1.6.2 LSF

### 1.6.2.1 Requirements

You have to install and configure the LSF batch system software before installing and configuring the CREAM software.

### 1.6.2.2 Installation

If you are running LSF, install the `emi-lsf-utils` metapackage: `yum install emi-lsf-utils`

### 1.6.2.3 Configuration

Set your `siteinfo.def` file, which is the input file used by yaim. Documentation about yaim variables relevant for CREAM CE is available at CREAM CE:
https://twiki.cern.ch/twiki/bin/view/LCG/Site-info_configuration_variables#cream_CE

The CREAM CE LSF integration is then configured running YAIM:

- no cluster mode: `/opt/glite/yaim/bin/yaim -c -s <site-info.def> -n creamCE -n LSF_utils`
- cluster mode with glite-CLUSTER deployed on a different node: `/opt/glite/yaim/bin/yaim -c -s <site-info.def> -n creamCE -n LSF_utils`
- cluster mode with glite-CLUSTER deployed on the same node of the CREAM-CE: `/opt/glite/yaim/bin/yaim -c -s <site-info.def> -n creamCE -n LSF_utils -n glite-CLUSTER`

## 1.6.3 Grid Engine

### 1.6.3.1 Requirements

You have to install and configure the GE batch system software before installing and configuring the CREAM software. The CREAM CE integration was tested with GE 6.2u5 but it should work with any forked version of the original GE software. The support of the GE batch system software (or any of its forked versions) is out of the scope of this activity.

Before proceeding, please take note of the following remarks:

1. CREAM CE must be installed in a separate node from the GE SERVER (GE QMASTER).
2. CREAM CE must work as a GE submission host (use `qconf -as <CE.MY.DOMAIN>` in the GE QMASTER to set it up).

### 1.6.3.2 Integration plugins

The GE integration with CREAM CE consists in deploying specific BLAH plugins and configure them to properly interoperate with Grid Engine batch system. The following GE BLAH plugins are deployed with CREAM CE installation: **BUpdaterSGE, sge_hold.sh, sge_submit.sh, sge_resume.sh, sge_status.sh and**

**sge_cancel.**

#### 1.6.3.3 Installation

If you are running GE, install the `emi-ge-utils` metapackage: `yum install emi-ge-utils`

#### 1.6.3.4 Configuration

Set your `siteinfo.def` file, which is the input file used by yaim. Documentation about yaim variables relevant for CREAM CE and GE is available at

- CREAM CE: https://twiki.cern.ch/twiki/bin/view/LCG/Site-info_configuration_variables#cream_CE
- SGE / GE Utils: https://twiki.cern.ch/twiki/bin/view/LCG/Site-info_configuration_variables#SGE_GE_utils

The most relevant GE YAIM variables to set in your site-info.def are:

1. BLPARSER_WITH_UPDATER_NOTIFIER= `"true"`
2. JOB_MANAGER= `sge`
3. CE_BATCH_SYS= `sge`
4. SGE_ROOT= `<Path to your SGE installation>. Default: "/usr/local/sge/pro"`
5. SGE_CELL= `<Path to your SGE CELL>. Default: "default"`
6. SGE_QMASTER= `<SGE QMASTER PORT>. Default: "536"`
7. SGE_EXECD= `<SGE EXECD PORT>. Defaul: "537"`
8. SGE_SPOOL_METH= `"classic"`
9. BATCH_SERVER= `<FQDN of your QMASTER>`
10. BATCH_LOG_DIR= `<Path for the GE accounting file>`
11. BATCH_BIN_DIR= `<Path for the GE binaries>`
12. BATCH_VERSION= `<GE version>`

Some sites use GE installations shared via NFS (or equivalent) in the CREAM CE. In order to prevent changes in that setup when YAIM is executed, define `SGE_SHARED_INSTALL=yes` in your `site-info.def`, otherwise YAIM may change your setup according to the definitions in your `site-info.def`.

The CREAM CE GE integration is then configured running YAIM:

- no cluster mode: `/opt/glite/yaim/bin/yaim -c -s <site-info.def> -n creamCE -n SGE_utils`
- cluster mode with glite-CLUSTER deployed on a different node: `/opt/glite/yaim/bin/yaim -c -s <site-info.def> -n creamCE -n SGE_utils`
- cluster mode with glite-CLUSTER deployed on the same node of the CREAM-CE: `/opt/glite/yaim/bin/yaim -c -s <site-info.def> -n creamCE -n SGE_utils -n glite-CLUSTER`

#### 1.6.3.5 Important notes

##### 1.6.3.5.1 File transfers

Besides the input/output sandbox files (transfered via GFTP) there are some other files that need to be transferred from/to the CREAM sandbox directory on the CE node to/from the Worker Node, namely:

- The CREAM job wrapper and the user proxies, that are staged from the CE node to the WN where the job will run
- The standard output and error files of the Cream job wrapper, that are copied from the WN to the CE when the job completes its execution.

Since GE does not implement staging capabilitites by default, we distribute the **sge_filestaging** file with the GE CREAM software. In order to enable the copy of the previous files:

1. Copy the **sge_filestaging** file to all your WNs (or to a shared directory mounted on your WNs)
2. Add `<path>/sge_filestaging --stagein` and `<path>/sge_filestaging --stageout` to your prolog and epilog defined in GE global configuration (use `qconf -mconf`), or alternatively, in each queue configuration `(qconf -mq <QUEUE>)`.
3. If you do **not share** the CREAM sanbox area between the CREAM CE node and the Worker Node, the **sge_filestaging** file requires configuring the ssh trust between CE and WNs.
4. If you share the CREAM sanbox area between the CREAM CE node and the Worker Node, the **sge_filestaging has to be changed according to:**

```
# diff -Nua sge_filestaging.modified sge_filestaging.orig
--- sge_filestaging.modified    2010-03-25 19:38:11.000000000 +0000
+++ sge_filestaging.orig    2010-03-25 19:05:43.000000000 +0000
@@ -21,9 +21,9 @@
     my $remotefile    = $3;

     if ( $STAGEIN ) {
-    system( 'cp', $remotefile, $localfile );
+    system( 'scp', "$remotemachine:$remotefile", $localfile );
     } else {
-    system( 'cp', $localfile, $remotefile );
+    system( 'scp', $localfile, "$remotemachine:$remotefile" );
     }
 }
```

#### 1.6.3.5.2 GE accounting file

BUpdaterSGE needs to consult the GE accounting file to determine how did a given job ended. Therefore, the GE accounting file must be shared between the GE SERVER / QMASTER and the CREAM CE.

Moreover, to guarantee that the accounting file is updated on the fly, the GE configuration should be tunned (using qconf -mconf) in order to add under the reporting_params the following definitions:
`accounting=true accounting_flush_time=00:00:00`

#### 1.6.3.5.3 GE SERVER (QMASTER) tuning

The following suggestions should be implemented to achieve better performance when integrating with CREAM CE:

1. The Cream CE machine must be set as a submission machine
2. The GE QMASTER configuration should have the definition `execd_params INHERIT_ENV=false` (use `qconf -mconf` to set it up). This setting allows to propagate the environment of the submission machine (CREAM CE) into the execution machine (WN).

# 2 Postconfiguration

Have a look at the Known issue page .

In particular consider the workaround needed for this problem .

# 3 Operating the system

## 3.1 Tomcat configuration guidelines

In `/etc/tomcat5/tomcat5.conf`, there are some settings related to heap. They are in the `JAVA_OPTS` setting (see `-Xms` and `-Xmx`).

It is suggested to customize such settings taking into account how much physical memory is available, as indicated in the following table (which refers to 64bit architectures):

| Memory | < 2 GB | 2 - 4 GB | > 4 GB |
|---|---|---|---|
| JAVA_OPTS setting | -Xms128m -Xmx512m | -Xms512m -Xmx1024m | -Xms512m -Xmx2048m |

After having done the changes, it is necessary to restart tomcat

## 3.2 MySQL database configuration guidelines

Default values of some MySQL settings are likely to be suboptimal especially for large machines. In particular some parameters could improve the overall performance if carefully tuned.
In this context one relevant parameter to be set is the `innodb_buffer_poll_size` which specifies the size of the buffer pool (the default value is 8MB).

The benefits obtained by using a proper value for this parameter are principally: an appreciable performance improving and the reduced amount of disk I/O needed for accessing the data in the tables. The optimal value depends on the amount of physical memory and the CPU architecture available in the host machine.

The maximum value depends on the CPU architecture, 32-bit or 64-bit. For 32-bit systems, the CPU architecture and operating system sometimes impose a lower practical maximum size.
Larger this value is set, less disk I/O is needed to access data in tables. On a dedicated database server, it is possible to set this to up to 80% of the machine physical memory size.
Scale back this value whether one of the following issues occur:

- competition for physical memory might cause paging in the operating system;
- innoDB reserves additional memory for buffers and control structures, so that the total allocated space is approximately 10% greater than the specified size.

In `/etc/my.cnf`, in particular within the [mysqld] section, it is suggested to customize the innodb_buffer_pool_size parameter taking into account how much physical memory is available.

Example:

```
[mysqld]
innodb_buffer_pool_size=512MB
```

After that, it's necessary to restart the mysql service for applying the change:

```
/etc/init.d/mysqld restart
```

Finally, the following sql command (root rights are needed) could be used for checking if the new value was applied successfully:

```
SHOW VARIABLES like 'innodb_buffer_pool_size';
```

# 3.3 MySQL database: How to resize Innodb log files

If the following error occurs (see the mysql log file: /var/log/mysqld.log)

```
InnoDB: ERROR: the age of the last checkpoint is ,
InnoDB: which exceeds the log group capacity .
InnoDB: If you are using big BLOB or TEXT rows,you must set the
InnoDB: combined size of log files at least 10 times bigger than the
InnoDB: largest such row.
```

then you must resize the innodb log files.

Follow these steps:

- check for the value of the `innodb_log_file_size`.

```
SHOW VARIABLES  like "innodb_log_file_size";
```

- Stop the MySQL server and make sure it shuts down without any errors. You can have a look at the error log to see if there are no errors.

```
service mysqld stop
```

- Once the server has stopped, edit the configuration file ( `/etc/my.cnf`) and insert or change the value of `innodb_log_file_size` to your desired value (64M should be a proper value). Example:

```
[mysqld]
innodb_log_file_size=64M
```

- Move the log file sizes `ib_log*` to some place out of the the directory where the log files reside. Example:

```
mv /var/lib/mysql/ib_logfile* /tmp
```

- Now restart the server.

```
service mysqld start
```

- Check for errors in /var/log/mysqld.log file

- Verify the correct size of the log files

```
ls -lrth /var/lib/mysql/ib_logfile*
```

# 3.4 How to start the CREAM service

A site admin can start the CREAM service just starting the CREAM container:

```
/etc/init.d/tomcat5 start
```

In case the new BLAH blparser is used, this will also start it (if not already running).

If for some reason it necessary to explicitly start the new BLAH blparser, the following command can be used:

```
/etc/init.d/glite-ce-blahparser start
```

If instead the old BLAH blparser is used, before starting tomcat it is necessary to start it on the BLPARSER_HOST using the command:

```
/etc/init.d/glite-ce-blparser start
```

To stop the CREAM service, it is just necessary to stop the CREAM container:

```
/etc/init.d/tomcat5 stop
```

# 3.5 Daemons

Information about daemons running in the CREAM CE is available in http://wiki.italiangrid.org/twiki/bin/view/CREAM/ServiceReferenceCard#Daemons_running

# 3.6 Init scripts

Information about init scripts in the CREAM CE is available in the http://wiki.italiangrid.org/twiki/bin/view/CREAM/ServiceReferenceCard#Init_scripts_and_options_start_s

# 3.7 Configuration files

Information about configuration files in the CREAM CE is available in http://wiki.italiangrid.org/twiki/bin/view/CREAM/ServiceReferenceCard#Configuration_files_location_wit

# 3.8 Log files

Information about log files in the CREAM CE is available in http://wiki.italiangrid.org/twiki/bin/view/CREAM/ServiceReferenceCard#Logfile_locations_and_management

# 3.9 Network ports

Information about ports used in the CREAM CE is available in http://wiki.italiangrid.org/twiki/bin/view/CREAM/ServiceReferenceCard#Open_ports

# 3.10 Cron jobs

Information about cron jobs used in the CREAM CE is available in http://wiki.italiangrid.org/twiki/bin/view/CREAM/ServiceReferenceCard#Cron_jobs

# 3.11 Security related operations

## 3.11.1 How to enable a certain VO for a certain CREAM CE in Argus

Let's consider that a certain CREAM CE has been configured to use ARGUS as authorization system.

Let's suppose that we chose `http://pd.infn.it/cream-18` as the id of the CREAM CE (i.e. yaim variable `CREAM_PEPC_RESOURCEID` is `http://pd.infn.it/cream-18`).

On the ARGUS box (identified by the yaim variable `ARGUS_PEPD_ENDPOINTS`) to enable the VO XYZ, it is necessary to define the following policy:

```
resource "http://pd.infn.it/cream-18" {
    obligation "http://glite.org/xacml/obligation/local-environment-map" {}
    action ".*" {
        rule permit { vo = "XYZ" }
    }
}
```

### 3.11.2 Security recommendations

Security recommendations relevant for the CREAM CE is available in http://wiki.italiangrid.org/twiki/bin/view/CREAM/ServiceReferenceCard#Security_recommendations

### 3.11.3 How to block/ban a user

Information about how to ban users is available in http://wiki.italiangrid.org/twiki/bin/view/CREAM/ServiceReferenceCard#How_to_block_ban_a_user

### 3.11.4 How to block/ban a VO

To ban a VO, it is suggested to reconfigure the service via yaim without that VO in the `siteinfo.def`

### 3.11.5 How to define a CREAM administrator

A CREAM administrator (aka super-user) can manage (e.g. cancel, check the status, etc.) also the jobs submitted by other people.

Moreover he/she can issue some privileged operations, in particular the ones to disable the new job submissions ( `glite-ce-disable-submission`) and then to re-enable them ( `glite-ce-disable-submission`)

To define a CREAM CE administrator for a specific CREAM CE, the DN of this person must be specified in the `/etc/grid-security/admin-list` of this CREAM CE node, e.g.:

```
"/C=IT/O=INFN/OU=Personal Certificate/L=Padova/CN=Massimo Sgaravatto"
```

Please note that including the DN between " is important

## 3.12 Input and Output Sandbox files transfer between the CREAM CE and the WN

The input and output sandbox files (unless they have to be copied from/to remote servers) are copied between the CREAM CE node and the Worker Node.

These files transfers can be done in two possible ways:

- Using gridftp
- Using the staging capabilities of the batch system

The choice is done at configuration time setting the yaim variable `SANDBOX_TRANSFER_METHOD_BETWEEN_CE_WN`. Possible values are:

- GSIFTP to use gridftp. This is the default value
- LRMS to use the staging capabilities of the batch system

# 3.13 Sharing of the CREAM sandbox area between the CREAM CE and the WN

Besides the input/output sandbox files there are some other files that need to be transferred from/to the CREAM sandbox directory on the CE node to/from the Worker Node:

- The CREAM job wrapper and the user proxies, that are staged from the CE node to the WN where the job will run
- The standard output and error files of the Cream job wrapper, that are copied from the WN to the CE when the job completes its execution.

To manage that, there are two possible options:

- Use the staging capabilities of the batch system (e.g. for Torque this requires configuring the ssh trust between CE and WNs)
- Share the CREAM sanbox area between the CREAM CE node and the Worker Node and configure appropriately the batch system

Please note:

- If you want to have several CREAM CEs sharing the same WNs, you need to mount each CE sandbox area to a different mount point on the WN, such as /cream_sandbox/ce_hostname.

- The CREAM sandbox directory name (default /var/cream_sandbox) can be changed using the yaim variable CREAM_SANDBOX_DIR

## 3.13.1 Sharing of the CREAM sandbox area between the CREAM CE and the WN for Torque

When Torque is used as batch system, to share the CREAM sandbox area between the CREAM CE node and the WNs:

- Mount the cream_sandbox directory also in the WNs. Let's assume that in the CE node the cream sandbox directory is called /var/cream_sandbox and on the WN is mounted as /cream_sandbox)
- On the WNs, add the following to the Torque client config file (generally /var/spool/pbs/mom_priv/config):

```
$usecp <CE node>://var/cream_sandbox /cream_sandbox
```

This $usecp line means that every time Torque will have to copy a file from/t the cream_sandbox directory on the CE (which is the case during the stage in/stage out phase), it will have to use a cp from /cream_sandbox instead.

# 3.14 Self-limiting CREAM behavior

CREAM is able to protect itself if the load, memory usage, etc. is too high. This happens disabling new job submissions, while the other commands are still allowed.

The whole stuff is implemented via a limiter script ( `/usr/bin/glite_cream_load_monitor`) very similar to the one used in the WMS.

Basically this limiter script check the values for some system and CREAM specific parameters, and compare them against some thresholds. If one or more threshold is exceeded, new job submissions get disabled. If a new submission is attempted when submissions are disabled, an error message is returned, e.g.:

```
$ glite-ce-job-submit -a -r cream-35.pd.infn.it:8443/cream-lsf-creamtest2 myjob.jdl

MethodName=[jobRegister] ErrorCode=[0] Description=[The CREAM service cannot accept jobs at the m
FaultCause=[Threshold for Memory Usage: 95 => Detected value for Memory Usage: 96.71%] Timestamp=
```

The limiter script is run every 10 minutes.

To disable the limiter, it is necessary to edit the CREAM configuration file `/etc/glite-ce-cream/cream-config.xml` setting `JOB_SUBMISSION_MANAGER_ENABLE` to false and restarting tomcat.

The values that are currently taken into account are the following:

| Value | Default threshold |
|---|---|
| Load average (1 min) | 40 |
| Load average (5 min) | 40 |
| Load average (15 min) | 20 |
| Memory usage | 95 % |
| Swap usage | 95 % |
| Free file descriptors | 500 |
| File descriptors used by tomcat | 800 |
| Number of FTP connections | 30 |
| Number of active jobs | no limit |
| Number of pending commands (commands still to be executed) | no limit |

If needed, the thresholds can be modified editing the limiter script itself (they are defined in the section starting with:

```
 # Default Values
```

If needed, the limiter script can be easily augmented to take into account some other parameters.

# 3.15 How to drain a CREAM CE

The administrator of a CREAM CE can decide to drain a CREAM CE, that is disabling new job submissions while allowing the other commands. This can be useful for example because of scheduled shutdown of the CREAM CE.

This can be achieved via the `glite-ce-disable-submission command` (provided by the CREAM CLI package installed on the UI), that can be issued only by a CREAM CE administrator, that is the DN of this person must be listed in the `/etc/grid-security/admin-list` file of the CE.

If newer job submissions are attempted, users will get an error message such as:

```
> glite-ce-job-submit -a -r grid006.pd.infn.it:8443/cream-lsf-grid02 lnl_test.j\dl
MethodName=[jobRegister] ErrorCode=[0] Description=[The CREAM2 service cannot
accept jobs anymore] FaultCause=[The CREAM2 service cannot accept jobs anymore]\
Timestamp=[Tue 22 Jan 2008 16:28:47]
```

It is possible to then resume new job submissions calling the `glite-ce-enable-submission` command.

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

It is possible to resume the job submission calling the proper operation ( `glite-ce-enable-submission`).

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
```

# 3.16 How to trace a specific job

To trace a specific job, first of all get the CREAMjobid.

If the job was submitted through the WMS, you can get its CREAMjobdid in the following way:

```
glite-wms-job-logging-info -v 2 <gridjobdid> | grep "Dest jobid"
```

If the job is not yours and you are not LB admin, you can get the CREAMjobid of that gridjobid if you have access to the CREAM logs doing:

```
grep <gridjobid> /var/log/cream.glite-ce-cream.log*
```

Grep the "last part" of the CREAMjobid in the CREAM log file (e.g. if the CREAMjobid is https://cream-07.pd.infn.it:8443/CREAM383606450 considers CREAM383606450):

```
grep CREAM383606450 /var/log/cream/glite-ce-cream.log*
```

This will return all the information relevant for this job

# 3.17 How to check if you are using the old or the new blparser

If you want to quickly check if you are using the old or the new BLAH Blparser, do a `grep registry /etc/blah.config`. If you see something like:

```
# grep registry /etc/blah.config
job_registry=/var/blah/user_blah_job_registry.bjr
```

you are using the new BLAH blparser. Otherwise you are using the old one.

# 3.18 Job purging

Purging a CREAM job means removing it from the CREAM database and removing from the CREAM CE any information relevant for that job (e.g. the job sandbox area).

When a job has been purged, it is not possible to manage it anymore (e.g. it is not possible to check anymore its status).

A job can be purged:

- Explicitly by the user who submitted that job, using e.g. the `glite-ce-job-purge` command provided by the CREAM CLI
- Automatically by the automatic CREAM job purger, which is responsible to purge old - forgotten jobs, according to a policy specified in the CREAM configuration file ( `/etc/glite-ce-cream/cream-config.xml`).

A user can purge only jobs she submitted. Only a CREAM CE admin can purge jobs submitted by other users.

For jobs submitted to a CREAM CE through the WMS, the purging is done by the ICE component of the WMS when it detects the job has reached a terminal status. The purging operation is not done if in the WMS conf file ( `/etc/glite_wms.conf`) the attribute `purge_jobs` in the ICE section is set to `false`.

## 3.18.1 Automatic job purging

The automatic CREAM job purger is responsible to purge old - forgotten jobs, according to a policy specified in the CREAM configuration file ( `/etc/glite-ce-cream/cream-config.xml`).

This policy is specified by the attribute `JOB_PURGE_POLICY`.

For example, if `JOB_PURGE_POLICY` is the following:

```
<parameter name="JOB_PURGE_POLICY" value="ABORTED 1 days; CANCELLED 2 days; DONE-OK 3 days; DONE-
```

then the job purger will purge jobs which are:

- in ABORTED status for more than 1 day
- in CANCELLED status for more than 2 days
- in DONE-OK status for more than 3 days
- in DONE-FAILED status more than 4 days
- in REGISTERED status for more than 5 days

## 3.18.2 Purging jobs in a non terminal status

The (manual or automatic) purge operation can be issued only for jobs which are in a terminal status. If it is necessary to purge a job which has been terminated but which is for CREAM in a non terminal status (e.g. RUNNING, REALLY_RUNNING) because of some bugs/problems/..., a specific utility ( `JobAdminPurger`) provided with the `glite-ce-cream` package can be used.

JobAdminPurger allows to purge jobs based on their CREAM jobids and/or their status (considering how long the job is in that status).

Usage:

```
JobDBAdminPurger.sh [-c|--conf CREAMConfPath] -u|--userDB userDB -p|--pswDB pswDB [-j|--jobids jo
```

Options:

- `-c | --conf` : the CREAM conf file (to be specified only if it is not the standard value `/etc/glite-ce-cream/cream-config.xml`)
- `-u | --userDB`: the cream DB user (as specified in the `/etc/tomcat5/Catalina/localhost/ce-cream.xml` file)
- `-p | --pswDB`: the cream DB password (as specified in the `/etc/tomcat5/Catalina/localhost/ce-cream.xml` file)
- `-j | --jobids`: the IDs (list of values separated by ':') of the jobs to be purged
- `-f | --filejobIds`: the file containing a list of jobids (one per line) to be purged
- `-s | --status`: the list of `state,deltatime` (list of values separated by ':') of the jobs to be purged. A job is purged if it is in specified `state` for more that `deltatime` days. deltatime can be omitted (which means that all jobs in that status will be purged). The possible states are:

- REGISTERED
- PENDING
- IDLE
- RUNNING
- REALLY-RUNNING
- CANCELLED
- HELD
- DONE-OK
- DONE-FAILED
- PURGED
- ABORTED

Examples:

```
JobDBAdminPurger.sh -u xyz -p abc -j CREAM217901296:CREAM324901232

JobDBAdminPurger.sh -u xyz -p abc -s registered:pending:idle

JobDBAdminPurger.sh u xyz -p abc -s registered,3:pending:idle,5

JobDBAdminPurger.sh u xyz -p abc -f /tmp/jobIdsToPurge.txt
```

Please note that this script should be run just to clean the CREAM DB in case of problems (i.e. jobs reported in a non terminal status while this is not the case)

Please also note that this script purges jobs from the CREAM DB. The relevant job sandbox directories are also deleted.

# 3.19 Proxy purging

Expired delegation proxies are automatically purged:

- from the DelegationDB
- from the file system ( `<cream-sandbox-dir>/<group>/<DN>/proxy`)

In the CREAM configuration file ( `/etc/glite-ce-cream/cream-config.xml`) there is a property called `delegation_purge_rate` which defines how often the proxy purger is run. The default value is 720 (720 minutes, that is 12 hours).

If the value is changed, it is then necessary to restart tomcat.

Setting that value to -1 means disabling the proxy purger.

# 3.20 Job wrapper management

## 3.20.1 Customization points

The CREAM JobWrapper running on the WN execute some scripts (to be provided by the local administrators) if they exist. These are called `customization points`.

There are 3 customization points:

- `${GLITE_LOCAL_CUSTOMIZATION_DIR}/cp_1.sh`. This is executed in the beginning of the CREAM job wrapper execution, before the creation of the temporary directory where the job is executed.
- `${GLITE_LOCAL_CUSTOMIZATION_DIR}/cp_2.sh`. This is executed just after the execution of the user job, before executing the epilogue (if any)
- `${GLITE_LOCAL_CUSTOMIZATION_DIR}/cp_3.sh`. This is executed just before the end of the JobWrapper execution.

If setting the customization point is not enough, the administrator can also customize the CREAM job wrapper, as explained in the following section

## 3.20.2 Customization of the CREAM Job wrapper

To customize the CREAM job wrapper it is just necessary to edit as appropriate the template file `/usr/share/tomcat5/webapps/ce-cream/WEB-INF/jobwrapper.tpl`.

When done, tomcat must be restarted.

## 3.20.3 Customization of the Input/Output Sandbox file transfers

The CREAM job wrapper, besides running the user payload, is also responsible for other operations, such as the transfer of the input and output sandbox files from/to remote gridftp servers.

If in such transfers there is a failure, the operation is retried after a while. The sleep time between the first attempt and the second one is the "initial wait time" specified in the CREAM configuration file. In every next attempt the sleep time is doubled.

In the CREAM configuration file ( `/etc/glite-ce-cream/cream-config.xml`) it is possible to set:

- the maximum number of file transfers that should be tried
- the initial wait time (i.e. the wait time between the first attempt and the second one)

Different values can be used for input (ISB) and output (OSB) files.

The relevant section in the CREAM configuration file is this one:

```
<parameter name="JOB_WRAPPER_COPY_RETRY_COUNT_ISB" value="2" />
<parameter name="JOB_WRAPPER_COPY_RETRY_FIRST_WAIT_ISB" value="60" /> <!-- sec. -->
<parameter name="JOB_WRAPPER_COPY_RETRY_COUNT_OSB" value="6" />
```

```
<parameter name="JOB_WRAPPER_COPY_RETRY_FIRST_WAIT_OSB" value="300" /> <!-- sec. -->
```

If one or more of these values are changed, it is then necessary to restart tomcat.

# 3.21 Managing the forwarding of requirements to the batch system

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

From a site administrator point of view, this requires creating and properly filling some scripts ( `/usr/bin/xxx_local_submit_attributes.sh`).

The relevant documentation is available at
http://wiki.italiangrid.org/twiki/bin/view/CREAM/UserGuide#1_Forward_of_requirements_to_the

# 3.22 Querying the CREAM Database

## 3.22.1 Check how many jobs are stored in the CREAM database

The following mysql query can be used to check how many jobs (along with their status) are reported in the CREAM database:

```
mysql> select jstd.name, count(*) from job, job_status_type_description jstd, job_status AS statu
```

# 3.23 How to clean the CREAM databases, sandboxes, log files?

[1] stop the service

```
service tomcat5 stop
```

[2] delete cream log files

```
rm -rf /var/log/cream/*
```

[3] delete tomcat log files

```
rm -rf /var/log/tomcat5/*
```

[4] drop databases

```
DROP DATABASE creamdb;
DROP DATABASE delegationdb;
```

[5] delete sandboxes (see the value of the "CREAM_SANDBOX_DIR" parameter in the /etc/glite-ce-cream/cream-config.xml file)

```
rm -rf "CREAM_SANDBOX_DIR"
```

[6] configure via yaim

see:
https://wiki.italiangrid.it/twiki/bin/view/CREAM/SystemAdministratorGuideForEMI1#1_4_CREAM_CE_configuratio

-- MassimoSgaravatto - 2011-04-07

---

This topic: CREAM > SystemAdministratorGuideForEMI1
Topic revision: r69 - 2013-08-09 - SaraBertocco