Table of Contents

CEMon Service Reference Card for EMI-1

Daemons running

- tomcat (`/usr/lib/jvm/java/bin/java -server -Xms128m -Xmx512m -Dglite.log.path=/var/log/cream -Dcatalina.ext.dirs=/usr/share/tomcat5/shared/lib:/usr/share/tomcat5/commo -Djavax.sql.DataSource.Factory=org.apache.commons.dbcp.BasicDataSource`)

Init scripts and options (start|stop|restart|...)

- Init script for tomcat: `/etc/init.d/tomcat5 {start|stop|restart|condrestart|try-restart|reload|force-reload|status|ver`

Configuration files location with example or template

- CEMon configuration file (`/etc/glite-ce-monitor/cemonitor-config.xml`). This file is created by yaim-cream-ce. A template is installed as `/etc/glite-ce-monitor/cemonitor-config.xml.template`. An example of this configuration file is available here

Logfile locations (and management) and other useful audit information

The relevant log files are:

- The tomcat log file (`/usr/share/tomcat5/logs/catalina.out`)
- The trustmanager log file (`/usr/share/tomcat5/logs/trustmanager.log`)
- The CEMon log file (`/var/log/cream/glite-ce-monitor.log`). The verbosity of this file can be increased modifying the file `/etc/glite-ce-monitor/log4j.properties` replacing:

```
log4j.logger.org.glite=info, fileout
```

with:

```
log4j.logger.org.glite=debug, fileout
```

You may also change the attributes `log4j.appender.fileout.MaxFileSize` and `log4j.appender.fileout.MaxBackupIndex` to change the maximum file size and the maximum number of log files to be kept.

Open ports

Service	From node	From port	To node	To port	Other info
	UI	*	CREAM-CE	8443	

CEMon Service					
CREAM job sensor	CEMon host	*	CREAM-CE	9909	Specified by `CREAM_JOB_SENSOR_PORT` in CREAM conf file. CEMON Host is usually the CREAM CE

Possible unit test of the service

TBD

Where is service state held (and can it be rebuilt)

CEMon job related information are kept in the filesystem in the directory `/var/cemonitor`

Cron jobs

None

Security information

Access control Mechanism description (authentication & authorization)

Authentication

Authentication in CEMon is managed via the trustmanager.

The Trust Manager is the component responsible for carrying out authentication operations. It is an implementation of the J2EE security specifications. Authentication is based on PKI. Each user (and Grid service) wishing to access CEMon is required to present an X.509 format certificate. These certificates are issued by trusted entities, the Certificate Authorities (CA). The role of a CA is to guarantee the identity of a user. This is achieved by issuing an electronic document (the certificate) that contains the information about the user and is digitally signed by the CA with its private key. An authentication manager, such as the Trust Manager, can verify the user identity by decrypting the hash of the certificate with the CA public key. This ensures that the certificate was issued by that specific CA. The Trust Manager can then access the user data contained in the certificate and verify the user identity.

Authorization for the CEMon service

Authorization in CEMon can be implemented in two different ways (the choice is done at configuration time):

- Authorization with ARGUS
- Authorization with gJAF

Argus is a system meant to render consistent authorization decisions for distributed services (e.g. compute elements, portals). In order to achieve this consistency a number of points must be addressed. First, it must be possible to author and maintain consistent authorization policies. This is handled by the Policy Administration Point (PAP) component in the service. Second, authored policies must be evaluated in a consistent manner, a task performed by the Policy Decision Point (PDP). Finally, the data provided for evaluation against policies must be consistent (in form and definition) and this is done by the Policy Enforcement Point (PEP). Argus is also responsible to manage the Grid user - local user mapping.

gJAF (Grid Java Authorization Framework) provides a way to invoke a chain of policy engines and get a decision result about the authorization of a user. The policy engines are divided in two types, depending on

their functionality. They can be plugged into the framework in order to form a chain of policy engines as selected by the administrator in order to let him set up a complete authorization system. A policy engine may be either a PIP or a PDP. PIP collect and verify assertions and capabilities associated with the user, checking her role, group and VO attributes. PDP may use the information retrieved by a PIP to decide whether the user is allowed to perform the requested action, whether further evaluation is needed, or whether the evaluation should be interrupted and the user access denied. In CEMon VO based authorization is supported. In this scenario, implemented via the VOMS PDP, the administrator can specify authorization policies based on the VO the jobs' owners belong to (or on particular VO attributes).

How to block/ban a user

If ARGUS is used as authorization system, ARGUS can be used to ban users.

Security recommendations

- It is recommended to close port 9909 (that is `CREAM_JOB_SENSOR_PORT`) to all nodes except the one running CEMon (which by default is the CREAM_CE node)

-- MassimoSgaravatto - 2011-04-20

This topic: CEMon > ServiceReferenceCard
Topic revision: r2 - 2012-04-19 - MassimoSgaravatto