LHC COMPUTING GRID

LCG - CE_TORQUE - GENERIC CONFIGURATION REFERENCE

Document identifier:	**LCG-GIS-CR-CE_torque**
EDMS id:	**none**
Version:	
Date:	**January 16, 2006**
Section:	**LCG Grid Infrastructure Support**
Document status:	**ACTIVE**
Author(s):	LCG Deployment - GIS team;Retico,Antonio;Vidic,Valentin;
File:	**CE_torque**

Abstract: *Configuration steps done by the YAIM script 'configure_CE_torque'*

CONTENTS

1. INTRODUCTION

This document lists the manual steps for the installation and configuration of a LCG CE_torque Node. Furthermore it provides a specification of the YAIM functions used to configure the node with the script-based configuration.

The configuration has been tested on a standard Scientific Linux 3.0 Installation.

Link to this document:
This document is available on the *Grid Deployment* web site

`http://www.cern.ch/grid-deployment/gis/lcg-GCR/index.html`

2. VARIABLES

In order to set-up a CE_torque node, you need at least the following variables to be correctly configured in the site configuration file (site-info.def):

APEL_DB_PASSWORD : database password for apel.

BATCH_LOG_DIR : Your batch system log directory.

BDII_FCR : Set the URL of the Freedom of Choice for Rescources URL.

BDII_HOST : BDII Hostname.

BDII_HTTP_URL : URL pointing to the BDII configuration file.

BDII_REGIONS : List of node types publishing information on the bdii. For each item listed in the BDII_REGIONS variable you need to create a set of new variables as follows:

BDII_<REGION>_URL : URL of the information producer (e.g.: BDII_CE_URL="URL of the CE information producer", BDII_SE_URL="URL of the SE information producer".

CE_BATCH_SYS : Implementation of site batch system. Available values are “torque”, “lsf”, “pbs”, “condor” etc.

CE_CPU_MODEL : Model of the CPU used by the WN (WN specification). This parameter is a string whose domain is not defined yet in the GLUE Schema. The value used for Pentium III is "PIII".

CE_CPU_SPEED : Clock frequency in Mhz (WN specification).

CE_CPU_VENDOR : Vendor of the CPU. used by the WN (WN specification). This parameter is a string whose domain is not defined yet in the GLUE Schema. The value used for Intel is “intel”.

CE_HOST : Computing Element Hostname.

CE_INBOUNDIP : TRUE if inbound connectivity is enabled at your site, FALSE otherwise (WN specification).

CE_MINPHYSMEM : RAM size in kblocks (WN specification).

CE_MINVIRTMEM : Virtual Memory size in kblocks (WN specification).

CE_OS : Operating System name (WN specification).

CE_OS_RELEASE : Operating System release (WN specification).

CE_OUTBOUNDIP : TRUE if outbound connectivity is enabled at your site, FALSE otherwise (WN specification).

CE_RUNTIMEENV : List of software tags supported by the site. The list can include VO-specific software tags. In order to assure backward compatibility it should include the entry ’LCG-2’, the current middleware version and the list of previous middleware tags.

CE_SF00 : Performance index of your fabric in SpecFloat 2000 (WN specification). For some examples of Spec values see http://www.specbench.org/osg/cpu2000/results/cint2000.html.

CE_SI00 : Performance index of your fabric in SpecInt 2000 (WN specification). For some examples of Spec values see http://www.specbench.org/osg/cpu2000/results/cint2000.html.

CE_SMPSIZE : Number of cpus in an SMP box (WN specification).

CLASSIC_HOST : The name of your SE_classic host.

CLASSIC_STORAGE_DIR : The root storage directory on CLASSIC_HOST.

CRON_DIR : Yaim writes all cron jobs to this directory. Change it if you want to turn off Yaim's management of cron.

DCACHE_ADMIN : Host name of the server node which manages the pool of nodes.

DPMDATA : Directory where the data is stored (absolute path, e.g./storage).

DPM_HOST : Host name of the DPM host, used also as a default DPM for the lcg-stdout-mon .

EDG_WL_SCRATCH : Optional scratch directory for jobs.

EDG_WL_SCRATCH : Optional scratch directory for jobs.

GLOBUS_TCP_PORT_RANGE : Port range for Globus IO.

GRIDICE_SERVER_HOST : GridIce server host name (usually run on the MON node).

GRIDMAP_AUTH : List of ldap servers in edg-mkgridmap.conf which authenticate users.

GRID_TRUSTED_BROKERS : List of the DNs of the Resource Brokers host certificates which are trusted by the Proxy node (ex: /O=Grid/O=CERN/OU=cern.ch/CN=host/testbed013.cern.ch).

GROUPS_CONF : Path to the groups.conf file which contains information on mapping VOMS groups and roles to local groups. An example of this configuration file is given in /opt/lcg/yaim/examples/groups.conf.

GSSKLOG : yes or no, indicating whether the site provides an AFS authentication server which maps gsi credentials into Kerberos tokens .

GSSKLOG_SERVER : If GSSKLOG is yes, the name of the AFS authentication server host.

INSTALL_ROOT : Installation root - change if using the re-locatable distribution.

JAVA_LOCATION : Path to Java VM installation. It can be used in order to run a different version of java installed locally.

JOB_MANAGER : The name of the job manager used by the gatekeeper.

LFC_CENTRAL : A list of VOs for which the LFC should be configured as a central catalogue.

LFC_HOST : Set this if you are building an LFC_HOST, not if you're just using clients.

LFC_LOCAL : Normally the LFC will support all VOs in the VOS variable. If you want to limit this list, add the ones you need to LFC_LOCAL. For each item listed in the VOS variable you need to create a set of new variables as follows:

VO_<VO-NAME>_QUEUES : The queues that the VO can use on the CE.

VO_<VO-NAME>_SE : Default SE used by the VO. WARNING: VO-NAME must be in capital cases.

VO_<VO-NAME>_SGM : ldap directory with VO software managers list. WARNING: VO-NAME must be in capital cases.

VO_<VO-NAME>_STORAGE_DIR : Mount point on the Storage Element for the VO. WARNING: VO-NAME must be in capital cases.

VO_<VO-NAME>_SW_DIR : Area on the WN for the installation of the experiment software. If on the WNs a predefined shared area has been mounted where VO managers can pre-install software, then these variable should point to this area. If instead there is not a shared area and each job must install the software, then this variables should contain a dot (.).Anyway the mounting of shared areas, as well as the local installation of VO software is not managed by *yaim* and should be handled locally by Site Administrators. WARNING: VO-NAME must be in capital cases.

VO_<VO-NAME>_USERS : ldap directory with VO users list. WARNING: VO-NAME must be in capital cases.

VO_<VO-NAME>_VOMS_POOL_PATH : If necessary, append this to the VOMS_SERVER URL for the pool account list .

VO_<VO-NAME>_VOMS_SERVERS : A list of VOMS servers for the VO.

MON_HOST : MON Box Hostname.

MY_DOMAIN : site's domain name.

PX_HOST : PX hostname.

QUEUES : The name of the queues for the CE. These are by default set as the VO names.

RB_HOST : Resource Broker Hostname.

REG_HOST : RGMA Registry hostname.

SE_LIST : A list of hostnames of the SEs available at your site.

SITE_EMAIL : The e-mail address as published by the information system.

SITE_LAT : Site latitude.

SITE_LOC : "City, Country".

SITE_LONG : Site longitude.

SITE_NAME : Your GIIS.

SITE_SUPPORT_SITE : Support entry point ; Unique Id for the site in the GOC DB and information system.

SITE_TIER : Site tier.

SITE_WEB : Site site.

TORQUE_SERVER : Set this if your torque server is on a different host from the CE. It is ingored for other batch systems.

USERS_CONF : Path to the file containing a list of Linux users (pool accounts) to be created. This file should be created by the Site Administrator, which contains a plain list of the users and IDs. An example of this configuration file is given in /opt/lcg/yaim/examples/users.conf.

VOBOX_HOST : VOBOX hostname.

VOBOX_PORT : The port the VOBOX gsisshd listens on.

VOS : List of supported VOs.

VO_SW_DIR : Directory for installation of experiment software.

WN_LIST : Path to the list of Worker Nodes. The list of Worker Nodes is a file to be created by the Site Administrator, which contains a plain list of the batch nodes. An example of this configuration file is given in /opt/lcg/yaim/examples/wn-list.conf.

3. CONFIGURE LIBRARY PATHS

Author(s): Retico,Antonio
Email : support-lcg-manual-install@cern.ch

This chapter describes the configuration steps done by the *yaim* function *'config_ldconf'*.

In order to allow the middleware libraries to be looked up and dinamically linked, the relevant paths need to be configured.

- If not already there, append the following lines to the file */etc/ld.so.conf*

```
<INSTALL_ROOT>/globus/lib
<INSTALL_ROOT>/edg/lib
<INSTALL_ROOT>/lcg/lib
/usr/local/lib
/usr/kerberos/lib
/usr/X11R6/lib
/usr/lib/qt-3.1/lib
/opt/gcc-3.2.2/lib
```

  where <INSTALL_ROOT> is the installation root of the lcg middleware (*/opt* by default).

- Run the command:

```
> /sbin/ldconfig -v
```

  (this command produces a huge amount of output)

3.1. SPECIFICATION OF FUNCTION: CONFIG_LDCONF

The function *'config_ldconf'* needs the following variables to be set in the configuration file:

INSTALL_ROOT : Installation root - change if using the re-locatable distribution.

The original code of the function can be found in:

```
/opt/lcg/yaim/functions/config_ldconf
```

The code is reproduced also in 27.1..

4. SET-UP EDG CONFIGURATION VARIABLES

Author(s): Retico,Antonio
Email : support-lcg-manual-install@cern.ch

This chapter describes the configuration steps done by the *yaim* function *'config_sysconfig_edg'*.

The EDG configuration file is parsed by EDG daemons to locate the EDG root directory and various other global properties.

Create and edit the file */etc/sysconfig/edg* as follows:

```
EDG_LOCATION=<INSTALL_ROOT>/edg
EDG_LOCATION_VAR=<INSTALL_ROOT>/edg/var
EDG_TMP=/tmp
X509_USER_CERT=/etc/grid-security/hostcert.pem
X509_USER_KEY=/etc/grid-security/hostkey.pem
GRIDMAP=/etc/grid-security/grid-mapfile
GRIDMAPDIR=/etc/grid-security/gridmapdir/
```

where <INSTALL_ROOT> is the installation root of the lcg middleware (*/opt* by default).

NOTE: it might be observed that some of the variables above listed dealing with the GSI (Grid Security Interface) are needed just on service nodes (e.g. CE, RB) and not on others. Nevertheless, for sake of simplicity, *yaim* uses the same definitions on all node types, which has been proven not to hurt.

4.1. SPECIFICATION OF FUNCTION: CONFIG_SYSCONFIG_EDG

The function *'config_sysconfig_edg'* needs the following variables to be set in the configuration file:

INSTALL_ROOT : Installation root - change if using the re-locatable distribution.

The original code of the function can be found in:

```
/opt/lcg/yaim/functions/config_sysconfig_edg
```

The code is reproduced also in 27.2..

5. SET-UP GLOBUS CONFIGURATION VARIABLES

Author(s): Retico,Antonio
Email : support-lcg-manual-install@cern.ch

This chapter describes the configuration steps done by the *yaim* function '*config_sysconfig_globus*'.

Create and edit the file */etc/sysconfig/globus* as follows:

```
GLOBUS_LOCATION=<INSTALL_ROOT>/globus
GLOBUS_CONFIG=/etc/globus.conf
GLOBUS_TCP_PORT_RANGE="20000 25000"
export LANG=C
```

where <INSTALL_ROOT> is the installation root of the lcg middleware (*/opt* by default).

5.1. SPECIFICATION OF FUNCTION: CONFIG_SYSCONFIG_GLOBUS

The function *'config_sysconfig_globus'* needs the following variables to be set in the configuration file:

GLOBUS_TCP_PORT_RANGE : Port range for Globus IO.

INSTALL_ROOT : Installation root - change if using the re-locatable distribution.

The original code of the function can be found in:

```
/opt/lcg/yaim/functions/config_sysconfig_globus
```

The code is reproduced also in 27.3..

6. SET-UP LCG CONFIGURATION VARIABLES

Author(s): Retico,Antonio
Email : support-lcg-manual-install@cern.ch

This chapter describes the configuration steps done by the *yaim* function '*config_sysconfig_lcg*'.

Create and edit the file */etc/sysconfig/lcg* as follows:

```
LCG_LOCATION=<INSTALL_ROOT>/lcg
LCG_LOCATION_VAR=<INSTALL_ROOT>/lcg/var
LCG_TMP=/tmp
```

where <INSTALL_ROOT> is the installation root of the lcg middleware (*/opt* by default).

6.1. SPECIFICATION OF FUNCTION: CONFIG_SYSCONFIG_LCG

The function *'config_sysconfig_lcg'* needs the following variables to be set in the configuration file:

INSTALL_ROOT : Installation root - change if using the re-locatable distribution.

SITE_NAME : Your GIIS.

The original code of the function can be found in:

```
/opt/lcg/yaim/functions/config_sysconfig_lcg
```

The code is reproduced also in 27.4..

7. SET-UP UPDATING OF CRLS

Author(s): Vidic,Valentin
Email : support-lcg-manual-install@cern.ch

This chapter describes the configuration steps done by the *yaim* function '*config_crl*'.

Cron script is installed to fetch new versions of CRLs four times a day. The time when the script is run is randomized in order to distribute the load on CRL servers. If the configuration is run as root, the cron entry is installed in */etc/cron.d/edg-fetch-crl*, otherwise it is installed as a user cron entry.

CRLs are also updated immediately by running the update script (*<INSTALL_ROOT>/edg/etc/cron/edg-fetch-crl-cron*).

Logrotate script is installed as */etc/logrotate.d/edg-fetch-crl* to prevent the logs from growing indefinitely.

7.1. SPECIFICATION OF FUNCTION: CONFIG_CRL

The function *'config_crl'* needs the following variables to be set in the configuration file:

INSTALL_ROOT : Installation root - change if using the re-locatable distribution.

The original code of the function can be found in:

```
/opt/lcg/yaim/functions/config_crl
```

The code is reproduced also in 27.5..

8. SET-UP RFIO

Author(s): Vidic,Valentin
Email : support-lcg-manual-install@cern.ch

This chapter describes the configuration steps done by the *yaim* function '*config_rfio*'.

rfiod is configured on SE_classic nodes by adding the appropriate ports (5001 TCP and UDP) to */etc/services* and restarting the daemon.

For SE_dpm nodes, *rfiod* is configured by *config_DPM_rfio* so no configuration is done here.

All other nodes don't run *rfiod*. However, *rfiod* might still be installed from *CASTOR-client* RPM. If this is the case, we make sure it's stopped and disabled.

8.1. SPECIFICATION OF FUNCTION: CONFIG_RFIO

The function *'config_rfio'* needs the following variables to be set in the configuration file:

INSTALL_ROOT : Installation root - change if using the re-locatable distribution.

The original code of the function can be found in:

```
/opt/lcg/yaim/functions/config_rfio
```

The code is reproduced also in 27.6..

9. SET-UP HOST CERTIFICATES

Author(s): Retico,Antonio
Email : support-lcg-manual-install@cern.ch

This chapter describes the configuration steps done by the *yaim* function '*config_host_certs*'.

The CE_torque node requires the host certificate/key files to be put in place before you start the installation.

Contact your national Certification Authority (CA) to understand how to obtain a host certificate if you do not have one already.

Instruction to obtain a CA list can be found in

`http://markusw.home.cern.ch/markusw/lcg2CAlist.html`

From the CA list so obtained you should choose a CA close to you.

Once you have obtained a valid certificate, i.e. a file

hostcert.pem

containing the machine public key and a file

hostkey.pem

containing the machine private key, make sure to place the two files into the directory

/etc/grid-security

with the following permissions

```
> chmod 400 /etc/grid-security/hostkey.pem

> chmod 644 /etc/grid-security/hostcert.pem
```

It is IMPORTANT that permissions be set as shown, as otherwise certification errors will occur.

If the certificates don't exist, the function exits with an error message and the calling process is interrupted.

9.1. SPECIFICATION OF FUNCTION: CONFIG_HOST_CERTS

The function *'config_host_certs'* needs the following variables to be set in the configuration file:

The original code of the function can be found in:

```
/opt/lcg/yaim/functions/config_host_certs
```

The code is reproduced also in 27.7..

10. CREATE POOL ACCOUNTS

Author(s): Vidic,Valentin
Email : support-lcg-manual-install@cern.ch

This chapter describes the configuration steps done by the *yaim* function *'config_users'*.

config_users creates pool accounts for grid users defined in *users.conf*. Each line in this file describes one user:

```
UID:LOGIN:GID:GROUP:VO:SGM_FLAG:
```

First, the format of the *users.conf* file is checked (VO and SGM fields were added recently).

Groups are then created for the supported VOs (listed in <*VOS*> variable) using *groupadd*.

For each of the lines in *users.conf*, a user account is created (with *useradd*) if that user's VO is supported.

Finally, grid users are denied access to *cron* and *at* by adding their usernames to */etc/at.deny* and */etc/cron.deny*.

10.1. SPECIFICATION OF FUNCTION: CONFIG_USERS

The function *'config_users'* needs the following variables to be set in the configuration file:

INSTALL_ROOT : Installation root - change if using the re-locatable distribution.

USERS_CONF : Path to the file containing a list of Linux users (pool accounts) to be created. This file should be created by the Site Administrator, which contains a plain list of the users and IDs. An example of this configuration file is given in /opt/lcg/yaim/examples/users.conf.

VOS : List of supported VOs.

The original code of the function can be found in:

```
/opt/lcg/yaim/functions/config_users
```

The code is reproduced also in 27.8..

11. CREATE EDG USERS

Author(s): Retico,Antonio
Email : support-lcg-manual-install@cern.ch

This chapter describes the configuration steps done by the *yaim* function '*config_edgusers*'.

Many of the services running on LCG service nodes are owned by the user *edguser*. The user edguser belongs to the group *edguser* and it has got a home directory in */home*.

The user *edginfo* is required on all the nodes publishing information on the Information System. The user belongs to the group *edginfo* and it has got a home directory in */home*.

No special requirements exists for the ID of the above mentioned users and groups.

The function creates both*edguser* and *edginfo* groups and users.

- group *edguser*: the group is created with group ID 995.
- user *edguser*: the user is created with group ID 995 and its home is */home/edguser*.
- group *edginfo*: the group is created with group ID 999.
- user *edginfo*: the user is created with group ID 999 and its home is */home/edguser*.

11.1. SPECIFICATION OF FUNCTION: CONFIG_EDGUSERS

The function *'config_edgusers'* needs the following variables to be set in the configuration file:

INSTALL_ROOT : Installation root - change if using the re-locatable distribution.

USERS_CONF : Path to the file containing a list of Linux users (pool accounts) to be created. This file should be created by the Site Administrator, which contains a plain list of the users and IDs. An example of this configuration file is given in /opt/lcg/yaim/examples/users.conf.

VOS : List of supported VOs.

The original code of the function can be found in:

```
/opt/lcg/yaim/functions/config_edgusers
```

The code is reproduced also in 27.9..

12. SET-UP POOL ACCOUNT MAPPINGS

Author(s): Vidic,Valentin
Email : support-lcg-manual-install@cern.ch

This chapter describes the configuration steps done by the *yaim* function '*config_mkgridmap*'.

Format of the *users.conf* file is checked first. This file should have six colon separated fields. Using this file, */etc/grid-security/gridmapdir* pool directory is created and initialized with pool accounts.

Next, configuration for *edg-mkgridmap* is generated in *<INSTALL_ROOT>/edg/etc/edg-mkgridmap.conf*. *edg-mkgridmap* generates */etc/grid-security/grid-mapfile* using VO membership information in VOMS and/or LDAP. The following lines are generated for each of the supported VOs:

```
group <VO_<vo>_SERVER>/Role=lcgadmin sgmuser
group <VO_<vo>_SERVER>/<VO_<vo>_VOMS_EXTRA_MAPS>
group <VO_<vo>_SERVER><VO_<vo>_VOMS_POOL_PATH> .user_prefix

group <VO_<vo>_SGM> sgmuser
group <VO_<vo>_USERS> .user_prefix
```

where *sgmuser* is SGM for the <vo> and *user_prefix* is the prefix for <vo> pool accounts (both values are inferred from *users.conf*). Multiple VOMS servers and extra maps can be defined.

Authentication URLs and site specific mappings are appended to the end of the file:

```
auth <GRIDMAP_AUTH>

gmf_local <INSTALL_ROOT>/edg/etc/grid-mapfile-local
```

If authentication URLs are not defined in <GRIDMAP_AUTH>, *ldap://lcg-registrar.cern.ch/ou=users,o=registrar,dc* is used.

Site specific grid user mappings can be defined in <INSTALL_ROOT>/edg/etc/grid-mapfile-local. Contents of this file are included verbatim in the output of *edg-mkgridmap*.

<INSTALL_ROOT>/edg/etc/lcmaps/gridmapfile is generated with the following contents for each supported VO:

```
/VO=<vo>/GROUP=/<vo>/ROLE=lcgadmin sgmuser
/VO=<vo>/GROUP=/<vo> .user_prefix
```

This file defines local account mappings for VOMS enabled proxy certificates.

<INSTALL_ROOT>/edg/etc/lcmaps/groupmapfile is generated with the following contents for each supported VO:

```
/VO=<vo>/GROUP=/<vo>/ROLE=lcgadmin vo_group
/VO=<vo>/GROUP=/<vo> vo_group
```

This file defines local group mappings for VOMS enabled proxy certificates.

After the configuration is finished, *edg-mkgridmap* is run with the new configuration to generate the */etc/grid-security/grid-mapfile*. Cron job for regenerating *grid-mapfile* is installed to run four times a day.

A cron job for expiring gridmapdir pool accounts is installed to run once a day on all nodes except nodes running *dpm*. This is a temporary fix to avoid users losing access to their files after the mapping expires and they are mapped to a different local user. By default, pool accounts expire if they are not used for more than 2 days, except on RB where they are expired after 10 days.

12.1. SPECIFICATION OF FUNCTION: CONFIG_MKGRIDMAP

The function *'config_mkgridmap'* needs the following variables to be set in the configuration file:

CRON_DIR : Yaim writes all cron jobs to this directory. Change it if you want to turn off Yaim's management of cron.

GRIDMAP_AUTH : List of ldap servers in edg-mkgridmap.conf which authenticate users.

GROUPS_CONF : Path to the groups.conf file which contains information on mapping VOMS groups and roles to local groups. An example of this configuration file is given in /opt/lcg/yaim/examples/groups.conf.

INSTALL_ROOT : Installation root - change if using the re-locatable distribution.

USERS_CONF : Path to the file containing a list of Linux users (pool accounts) to be created. This file should be created by the Site Administrator, which contains a plain list of the users and IDs. An example of this configuration file is given in /opt/lcg/yaim/examples/users.conf.

VOS : List of supported VOs. For each item listed in the VOS variable you need to create a set of new variables as follows:

VO_<VO-NAME>_SGM : ldap directory with VO software managers list. WARNING: VO-NAME must be in capital cases.

VO_<VO-NAME>_USERS : ldap directory with VO users list. WARNING: VO-NAME must be in capital cases.

VO_<VO-NAME>_VOMS_POOL_PATH : If necessary, append this to the VOMS_SERVER URL for the pool account list .

VO_<VO-NAME>_VOMS_SERVERS : A list of VOMS servers for the VO.

The original code of the function can be found in:

```
/opt/lcg/yaim/functions/config_mkgridmap
```

The code is reproduced also in 27.10..

13. SET-UP JAVA LOCATION

Author(s): Vidic,Valentin
Email : support-lcg-manual-install@cern.ch

This chapter describes the configuration steps done by the *yaim* function '*config_java*'.

Since Java is not included in the LCG distribution, Java location needs to be configured with yaim.

If <JAVA_LOCATION> is not defined in *site-info.def*, it is determined from installed Java RPMs (if available).

In relocatable distribution, JAVA_HOME environment variable is defined in *<INSTALL_ROOT>/etc/profile.d/grid_en* and *<INSTALL_ROOT>/etc/profile.d/grid_env.csh*.

Otherwise, JAVA_HOME is defined in */etc/java/java.conf* and */etc/java.conf* and Java binaries added to PATH in *<INSTALL_ROOT>/edg/etc/profile.d/j2.sh* and *<INSTALL_ROOT>/edg/etc/profile.d/j2.csh*.

13.1. SPECIFICATION OF FUNCTION: CONFIG_JAVA

The function *'config_java'* needs the following variables to be set in the configuration file:

INSTALL_ROOT : Installation root - change if using the re-locatable distribution.

JAVA_LOCATION : Path to Java VM installation. It can be used in order to run a different version of java installed locally.

The original code of the function can be found in:

```
/opt/lcg/yaim/functions/config_java
```

The code is reproduced also in 27.11..

14. SET-UP R-GMA CLIENT

Author(s): Vidic,Valentin
Email : support-lcg-manual-install@cern.ch

This chapter describes the configuration steps done by the *yaim* function '*config_rgma_client*'.

R-GMA client configuration is generated in *<INSTALL_ROOT>/glite/etc/rgma/rgma.conf* by running:

```
<INSTALL_ROOT>/glite/share/rgma/scripts/rgma-setup.py --secure=no --server=<MON_HOST> --registry=<REG_HOST> --schem
```

<INSTALL_ROOT>/edg/etc/profile.d/edg-rgma-env.sh and *<INSTALL_ROOT>/edg/etc/profile.d/edg-rgma-env.csh* with the following functionality:

- RGME_HOME is set to *<INSTALL_ROOT>/glite*
- APEL_HOME is set to *<INSTALL_ROOT>/glite*
- *<INSTALL_ROOT>/glite/lib/python* is added to PYTHONPATH
- *<INSTALL_ROOT>/glite/lib* is added to LD_LIBRARY_PATH.

These files are sourced into the users environment from *<INSTALL_ROOT>/etc/profile.d/z_edg_profile.sh* and *<INSTALL_ROOT>/etc/profile.d/z_edg_profile.csh*.

14.1. SPECIFICATION OF FUNCTION: CONFIG_RGMA_CLIENT

The function *'config_rgma_client'* needs the following variables to be set in the configuration file:

INSTALL_ROOT : Installation root - change if using the re-locatable distribution.

MON_HOST : MON Box Hostname.

REG_HOST : RGMA Registry hostname.

The original code of the function can be found in:

```
/opt/lcg/yaim/functions/config_rgma_client
```

The code is also reproduced in 27.12..

15. SET-UP GENERIC INFORMATION PROVIDER

Author(s): Vidic,Valentin
Email : support-lcg-manual-install@cern.ch

This chapter describes the configuration steps done by the *yaim* function '*config_gip*'.

Generic Information Provider (GIP) is configured through *<INSTALL_ROOT>/lcg/var/gip/lcg-info-generic.conf*. The start of this file is common for all types of nodes:

```
ldif_file=<INSTALL_ROOT>/lcg/var/gip/lcg-info-static.ldif
generic_script=<INSTALL_ROOT>/lcg/libexec/lcg-info-generic
wrapper_script=<INSTALL_ROOT>/lcg/libexec/lcg-info-wrapper
temp_path=<INSTALL_ROOT>/lcg/var/gip/tmp
template=<INSTALL_ROOT>/lcg/etc/GlueSite.template
template=<INSTALL_ROOT>/lcg/etc/GlueCE.template
template=<INSTALL_ROOT>/lcg/etc/GlueCESEBind.template
template=<INSTALL_ROOT>/lcg/etc/GlueSE.template
template=<INSTALL_ROOT>/lcg/etc/GlueService.template

# Common for all
GlueInformationServiceURL: ldap://<hostname>:2135/mds-vo-name=local,o=grid
```

<hostname> is determined by running *hostname -f*.

For CE the following is added:

```
dn: GlueSiteUniqueID=<SITE_NAME>,mds-vo-name=local,o=grid
GlueSiteName: <SITE_NAME>
GlueSiteDescription: LCG Site
GlueSiteUserSupportContact: mailto: <SITE_EMAIL>
GlueSiteSysAdminContact: mailto: <SITE_EMAIL>
GlueSiteSecurityContact: mailto: <SITE_EMAIL>
GlueSiteLocation: <SITE_LOC>
GlueSiteLatitude: <SITE_LAT>
GlueSiteLongitude: <SITE_LONG>
GlueSiteWeb: <SITE_WEB>
GlueSiteOtherInfo: <SITE_TIER>
GlueSiteOtherInfo: <SITE_SUPPORT_SITE>
GlueForeignKey: GlueSiteUniqueID=<SITE_NAME>
GlueForeignKey: GlueClusterUniqueID=<CE_HOST>
GlueForeignKey: GlueSEUniqueID=<SE_HOST>

dynamic_script=<INSTALL_ROOT>/lcg/libexec/lcg-info-dynamic-ce
dynamic_script=<INSTALL_ROOT>/lcg/libexec/lcg-info-dynamic-software <INSTALL_ROOT>/lcg/var/gip/lcg-info-generic.con

# CE Information Provider
GlueCEHostingCluster: <CE_HOST>
GlueCEInfoGatekeeperPort: 2119
GlueCEInfoHostName: <CE_HOST>
GlueCEInfoLRMSType: <CE_BATCH_SYS>
GlueCEInfoLRMSVersion: not defined
GlueCEInfoTotalCPUs: 0
```

```
GlueCEPolicyMaxCPUTime: 0
GlueCEPolicyMaxRunningJobs: 0
GlueCEPolicyMaxTotalJobs: 0
GlueCEPolicyMaxWallClockTime: 0
GlueCEPolicyPriority: 1
GlueCEStateEstimatedResponseTime: 0
GlueCEStateFreeCPUs: 0
GlueCEStateRunningJobs: 0
GlueCEStateStatus: Production
GlueCEStateTotalJobs: 0
GlueCEStateWaitingJobs: 0
GlueCEStateWorstResponseTime: 0
GlueHostApplicationSoftwareRunTimeEnvironment: <ce_runtimeenv>
GlueHostArchitectureSMPSize: <CE_SMPSIZE>
GlueHostBenchmarkSF00: <CE_SF00>
GlueHostBenchmarkSI00: <CE_SI00>
GlueHostMainMemoryRAMSize: <CE_MINPHYSMEM>
GlueHostMainMemoryVirtualSize: <CE_MINVIRTMEM>
GlueHostNetworkAdapterInboundIP: <CE_INBOUNDIP>
GlueHostNetworkAdapterOutboundIP: <CE_OUTBOUNDIP>
GlueHostOperatingSystemName: <CE_OS>
GlueHostOperatingSystemRelease: <CE_OS_RELEASE>
GlueHostOperatingSystemVersion: 3
GlueHostProcessorClockSpeed: <CE_CPU_SPEED>
GlueHostProcessorModel: <CE_CPU_MODEL>
GlueHostProcessorVendor: <CE_CPU_VENDOR>
GlueSubClusterPhysicalCPUs: 0
GlueSubClusterLogicalCPUs: 0
GlueSubClusterTmpDir: /tmp
GlueSubClusterWNTmpDir: /tmp
GlueCEInfoJobManager: <JOB_MANAGER>
GlueCEStateFreeJobSlots: 0
GlueCEPolicyAssignedJobSlots: 0
GlueCESEBindMountInfo: none
GlueCESEBindWeight: 0

dn: GlueClusterUniqueID=<CE_HOST>, mds-vo-name=local,o=grid
GlueClusterName: <CE_HOST}
GlueForeignKey: GlueSiteUniqueID=<SITE_NAME>
GlueClusterService: <CE_HOST>:2119/jobmanager-<JOB_MANAGER>-<queue>
GlueForeignKey: GlueCEUniqueID=<CE_HOST>:2119/jobmanager-<JOB_MANAGER>-<queue>

dn: GlueSubClusterUniqueID=<CE_HOST>, GlueClusterUniqueID=<CE_HOST>, mds-vo-name=local,o=grid
GlueChunkKey: GlueClusterUniqueID=<CE_HOST>
GlueSubClusterName: <CE_HOST>

dn: GlueCEUniqueID=<CE_HOST>:2119/jobmanager-<JOB_MANAGER>-<queue>, mds-vo-name=local,o=grid
GlueCEName: <queue>
GlueForeignKey: GlueClusterUniqueID=<CE_HOST>
GlueCEInfoContactString: <CE_HOST>:2119/jobmanager-<JOB_MANAGER>-<queue>
GlueCEAccessControlBaseRule: VO:<vo>

dn: GlueVOViewLocalID=<vo>,GlueCEUniqueID=<CE_HOST>:2119/jobmanager-<JOB_MANAGER>-<queue>,mds-vo-name=local,o=grid
GlueCEAccessControlBaseRule: VO:<vo>
```

```
GlueCEInfoDefaultSE: <VO_<vo>_DEFAULT_SE>
GlueCEInfoApplicationDir: <VO_<vo>_SW_DIR>
GlueCEInfoDataDir: <VO_<vo>_STORAGE_DIR>
GlueChunkKey: GlueCEUniqueID=<CE_HOST>:2119/jobmanager-<JOB_MANAGER>-<queue>

dn: GlueCESEBindGroupCEUniqueID=<CE_HOST>:2119/jobmanager-<JOB_MANAGER>-<queue>, mds-vo-name=local,o=grid
GlueCESEBindGroupSEUniqueID: <se_list>

dn: GlueCESEBindSEUniqueID=<se>, GlueCESEBindGroupCEUniqueID=<CE_HOST>:2119/jobmanager-<JOB_MANAGER>-<queue>, mds-v
GlueCESEBindCEAccesspoint: <accesspoint>
GlueCESEBindCEUniqueID: <CE_HOST>:2119/jobmanager-<JOB_MANAGER>-<queue>
```

where <*accesspoint*> is:

- <*DPMDATA*> for DPM SE
- */storage* for dCache
- <*CLASSIC_STORAGE_DIR*> for SE classic.

Some lines can be generated multiple times for different <*vo*>s, <*queue*>s, <*se*>s etc.

For each of the supported VOs, a directory is created in <*INSTALL_ROOT*>*/edg/var/info/*<*vo*>. These are used by SGMs to publish information on experiment software installed on the cluster.

For the nodes running GridICE server (usually SE) the following is added:

```
dn: GlueServiceUniqueID=<GRIDICE_SERVER_HOST>:2136,Mds-vo-name=local,o=grid
GlueServiceName: <SITE_NAME>-gridice
GlueServiceType: gridice
GlueServiceVersion: 1.1.0
GlueServiceEndpoint: ldap://<GRIDICE_SERVER_HOST>:2136/mds-vo-name=local,o=grid
GlueServiceStatus: OK
GlueServiceStatusInfo: No Problems
GlueServiceStartTime: 2002-10-09T19:00:00Z
GlueServiceOwner: LCG
GlueForeignKey: GlueSiteUniqueID=<SITE_NAME>
GlueServiceAccessControlRule:<vo>
```

For PX nodes the following is added:

```
dn: GlueServiceUniqueID=<PX_HOST>:7512,Mds-vo-name=local,o=grid
GlueServiceName: <SITE_NAME>-myproxy
GlueServiceType: myproxy
GlueServiceVersion: 1.1.0
GlueServiceEndpoint: <PX_HOST>:7512
GlueServiceStatus: OK
GlueServiceStatusInfo: No Problems
GlueServiceStartTime: 2002-10-09T19:00:00Z
GlueServiceOwner: LCG
GlueForeignKey: GlueSiteUniqueID=<SITE_NAME>
GlueServiceAccessControlRule: <grid_trusted_broker>
```

For nodes running RB the following is added:

```
dn: GlueServiceUniqueID=<RB_HOST>:7772,Mds-vo-name=local,o=grid
GlueServiceName: <SITE_NAME>-rb
GlueServiceType: ResourceBroker
GlueServiceVersion: 1.2.0
GlueServiceEndpoint: <RB_HOST>:7772
GlueServiceStatus: OK
GlueServiceStatusInfo: No Problems
GlueServiceStartTime: 2002-10-09T19:00:00Z
GlueServiceOwner: LCG
GlueForeignKey: GlueSiteUniqueID=<SITE_NAME>
GlueServiceAccessControlRule: <vo>

dn: GlueServiceDataKey=HeldJobs,GlueServiceUniqueID=gram://<RB_HOST>:7772,Mds-vo-name=local,o=grid
GlueServiceDataKey: HeldJobs
GlueServiceDataValue: 0
GlueChunkKey: GlueServiceUniqueID=gram://<RB_HOST>:7772

dn: GlueServiceDataKey=IdleJobs,GlueServiceUniqueID=gram://<RB_HOST>:7772,Mds-vo-name=local,o=grid
GlueServiceDataKey: IdleJobs
GlueServiceDataValue: 0
GlueChunkKey: GlueServiceUniqueID=gram://<RB_HOST>:7772

dn: GlueServiceDataKey=JobController,GlueServiceUniqueID=gram://<RB_HOST>:7772,Mds-vo-name=local,o=grid
GlueServiceDataKey: JobController
GlueServiceDataValue: 0
GlueChunkKey: GlueServiceUniqueID=gram://<RB_HOST>:7772

dn: GlueServiceDataKey=Jobs,GlueServiceUniqueID=gram://<RB_HOST>:7772,Mds-vo-name=local,o=grid
GlueServiceDataKey: Jobs
GlueServiceDataValue: 0
GlueChunkKey: GlueServiceUniqueID=gram://<RB_HOST>:7772

dn: GlueServiceDataKey=LogMonitor,GlueServiceUniqueID=gram://<RB_HOST>:7772,Mds-vo-name=local,o=grid
GlueServiceDataKey: LogMonitor
GlueServiceDataValue: 0
GlueChunkKey: GlueServiceUniqueID=gram://<RB_HOST>:7772

dn: GlueServiceDataKey=RunningJobs,GlueServiceUniqueID=gram://<RB_HOST>:7772,Mds-vo-name=local,o=grid
GlueServiceDataKey: RunningJobs
GlueServiceDataValue: 14
GlueChunkKey: GlueServiceUniqueID=gram://<RB_HOST>:7772

dn: GlueServiceDataKey=WorkloadManager,GlueServiceUniqueID=gram://<RB_HOST>:7772,Mds-vo-name=local,o=grid
GlueServiceDataKey: WorkloadManager
GlueServiceDataValue: 0
GlueChunkKey: GlueServiceUniqueID=gram://<RB_HOST>:7772
```

For central LFC the following is added:

```
dn: GlueServiceUniqueID=http://<LFC_HOST>:8085/,mds-vo-name=local,o=grid
GlueServiceName: <SITE_NAME>-lfc-dli
GlueServiceType: data-location-interface
GlueServiceVersion: 1.0.0
GlueServiceEndpoint: http://<LFC_HOST>:8085/
GlueServiceURI: http://<LFC_HOST}:8085/
```

```
GlueServiceAccessPointURL: http://<LFC_HOST>:8085/
GlueServiceStatus: running
GlueForeignKey: GlueSiteUniqueID=<SITE_NAME>
GlueServiceOwner: <vo>
GlueServiceAccessControlRule: <vo>

dn: GlueServiceUniqueID=<LFC_HOST>,mds-vo-name=local,o=grid
GlueServiceName: <SITE_NAME>-lfc
GlueServiceType: lcg-file-catalog
GlueServiceVersion: 1.0.0
GlueServiceEndpoint: <LFC_HOST>
GlueServiceURI: <LFC_HOST>
GlueServiceAccessPointURL: <LFC_HOST>
GlueServiceStatus: running
GlueForeignKey: GlueSiteUniqueID=<SITE_NAME>
GlueServiceOwner: <vo>
GlueServiceAccessControlRule: <vo>
```

For local LFC the following is added:

```
dn: GlueServiceUniqueID=<LFC_HOST>,mds-vo-name=local,o=grid
GlueServiceName: <SITE_NAME>-lfc
GlueServiceType: lcg-local-file-catalog
GlueServiceVersion: 1.0.0
GlueServiceEndpoint: <LFC_HOST>
GlueServiceURI: <LFC_HOST>
GlueServiceAccessPointURL: <LFC_HOST>
GlueServiceStatus: running
GlueForeignKey: GlueSiteUniqueID=<SITE_NAME>
GlueServiceOwner: <vo>
GlueServiceAccessControlRule: <vo>
```

For dcache and dpm nodes the following is added:

```
dn: GlueServiceUniqueID=httpg://<SE_HOST>:8443/srm/managerv1,Mds-Vo-name=local,o=grid
GlueServiceAccessPointURL: httpg://<SE_HOST>:8443/srm/managerv1
GlueServiceEndpoint: httpg://<SE_HOST>:8443/srm/managerv1
GlueServiceType: srm_v1
GlueServiceURI: httpg://<SE_HOST>:8443/srm/managerv1
GlueServicePrimaryOwnerName: LCG
GlueServicePrimaryOwnerContact: mailto:<SITE_EMAIL>
GlueForeignKey: GlueSiteUniqueID=<SITE_NAME>
GlueServiceVersion: 1.0.0
GlueServiceAccessControlRule: <vo>
GlueServiceInformationServiceURL: MDS2GRIS:ldap://<BDII_HOST>:2170/mds-voname=local,mds-vo-name=<SITE_NAME>,mds-vo-
GlueServiceStatus: running
```

For all types of SE the following is added:

```
dynamic_script=<INSTALL_ROOT>/lcg/libexec/lcg-info-dynamic-se

GlueSEType: <se_type>
GlueSEPort: 2811
GlueSESizeTotal: 0
GlueSESizeFree: 0
```

```
GlueSEArchitecture: <se_type>
GlueSAType: permanent
GlueSAPolicyFileLifeTime: permanent
GlueSAPolicyMaxFileSize: 10000
GlueSAPolicyMinFileSize: 1
GlueSAPolicyMaxData: 100
GlueSAPolicyMaxNumFiles: 10
GlueSAPolicyMaxPinDuration: 10
GlueSAPolicyQuota: 0
GlueSAStateAvailableSpace: 1
GlueSAStateUsedSpace: 1

dn: GlueSEUniqueID=<SE_HOST>,mds-vo-name=local,o=grid
GlueSEName: <SITE_NAME>:<se_type>
GlueForeignKey: GlueSiteUniqueID=<SITE_NAME>

dn: GlueSEAccessProtocolLocalID=gsiftp, GlueSEUniqueID=<SE_HOST>,Mds-Vo-name=local,o=grid
GlueSEAccessProtocolType: gsiftp
GlueSEAccessProtocolPort: 2811
GlueSEAccessProtocolVersion: 1.0.0
GlueSEAccessProtocolSupportedSecurity: GSI
GlueChunkKey: GlueSEUniqueID=<SE_HOST>

dn: GlueSEAccessProtocolLocalID=rfio, GlueSEUniqueID=<SE_HOST>,Mds-Vo-name=local,o=grid
GlueSEAccessProtocolType: rfio
GlueSEAccessProtocolPort: 5001
GlueSEAccessProtocolVersion: 1.0.0
GlueSEAccessProtocolSupportedSecurity: RFIO
GlueChunkKey: GlueSEUniqueID=<SE_HOST>
```

where <*se_type*> is *srm_v1* for DPM and dCache and *disk* otherwise.

For SE_dpm the following is added:

```
dn: GlueSALocalID=<vo>,GlueSEUniqueID=<SE_HOST>,Mds-Vo-name=local,o=grid
GlueSARoot: <vo>:/dpm/<domain>/home/<vo>
GlueSAPath: <vo>:/dpm/<domain>/home/<vo>
GlueSAAccessControlBaseRule: <vo>
GlueChunkKey: GlueSEUniqueID=<SE_HOST>
```

For SE_dcache the following is added:

```
dn: GlueSALocalID=<vo>,GlueSEUniqueID=<SE_HOST>,Mds-Vo-name=local,o=grid
GlueSARoot: <vo>:/pnfs/<domain>/home/<vo>
GlueSAPath: <vo>:/pnfs/<domain>/home/<vo>
GlueSAAccessControlBaseRule: <vo>
GlueChunkKey: GlueSEUniqueID=<SE_HOST>
```

For other types of SE the following is used:

```
dn: GlueSALocalID=<vo>,GlueSEUniqueID=<SE_HOST>,Mds-Vo-name=local,o=grid
GlueSARoot: <vo>:<vo>
GlueSAPath: <VO_<vo>_STORAGE_DIR>
GlueSAAccessControlBaseRule: <vo>
GlueChunkKey: GlueSEUniqueID=<SE_HOST>
```

For VOBOX the following is added:

```
dn: GlueServiceUniqueID=gsissh://<VOBOX_HOST>:<VOBOX_PORT>,Mds-vo-name=local,o=grid
GlueServiceAccessPointURL: gsissh://<VOBOX_HOST>:<VOBOX_PORT>
GlueServiceName: <SITE_NAME>-vobox
GlueServiceType: VOBOX
GlueServiceEndpoint: gsissh://<VOBOX_HOST>:<VOBOX_PORT>
GlueServicePrimaryOwnerName: LCG
GlueServicePrimaryOwnerContact: <SITE_EMAIL>
GlueForeignKey: GlueSiteUniqueID=<SITE_NAME>
GlueServiceVersion: 1.0.0
GlueServiceInformationServiceURL: ldap://<VOBOX_HOST>:2135/mds-vo-name=local,o=grid
GlueServiceStatus: running
GlueServiceAccessControlRule: <vo>
```

Configuration script is run:

```
<INSTALL_ROOT>/lcg/sbin/lcg-info-generic-config <INSTALL_ROOT>/lcg/var/gip/lcg-info-generic.conf
```

Configuration script generates a ldif file (*<INSTALL_ROOT>/lcg/var/gip/lcg-info-static.ldif*) by merging templates from *<INSTALL_ROOT>/lcg/etc/* and data from *<INSTALL_ROOT>/lcg/var/gip/lcg-info-generic.conf*. Wrapper script is also created in *<INSTALL_ROOT>/lcg/libexec/lcg-info-wrapper*.

<INSTALL_ROOT>/globus/libexec/edg.info is created:

```
#!/bin/bash
#
# info-globus-ldif.sh
#
#Configures information providers for MDS
#
cat << EOF

dn: Mds-Vo-name=local,o=grid
objectclass: GlobusTop
objectclass: GlobusActiveObject
objectclass: GlobusActiveSearch
type: exec
path: <INSTALL_ROOT>/lcg/libexec
base: lcg-info-wrapper
args:
cachetime: 60
timelimit: 20
sizelimit: 250

EOF
```

<INSTALL_ROOT>/globus/libexec/edg.info is created:

```
#!/bin/bash

cat <<EOF
<INSTALL_ROOT>/globus/etc/openldap/schema/core.schema
<INSTALL_ROOT>/glue/schema/ldap/Glue-CORE.schema
<INSTALL_ROOT>/glue/schema/ldap/Glue-CE.schema
```

```
<INSTALL_ROOT>/glue/schema/ldap/Glue-CESEBind.schema
<INSTALL_ROOT>/glue/schema/ldap/Glue-SE.schema
EOF
```

These two scripts are used to generate *slapd* configuration for Globus MDS.

<INSTALL_ROOT>/lcg/libexec/lcg-info-dynamic-ce is generated to call the information provider appropriate for the LRMS. For Torque the file has these contents:

```
#!/bin/sh
<INSTALL_ROOT>/lcg/libexec/lcg-info-dynamic-pbs <INSTALL_ROOT>/lcg/var/gip/lcg-info-generic.conf <TORQUE_SERVER>
```

R-GMA GIN periodically queries MDS and inserts the data into R-GMA. GIN is configured on all nodes except UI and WN by copying host certificate to *<INSTALL_ROOT>/glite/var/rgma/.certs* and updating the configuration file appropriately (*<INSTALL_ROOT>/glite/etc/rgma/ClientAuthentication.props*). Finally, GIN configuration script (*<INSTALL_ROOT>/glite/bin/rgma-gin-config*) is run to configure the mapping between Glue schema in MDS and Glue tables in R-GMA. *rgma-gin* service is restarted and configured to start on boot.

15.1. SPECIFICATION OF FUNCTION: CONFIG_GIP

The function *'config_gip'* needs the following variables to be set in the configuration file:

BDII_HOST : BDII Hostname.

CE_BATCH_SYS : Implementation of site batch system. Available values are “torque”, “lsf”, “pbs”, “condor” etc.

CE_CPU_MODEL : Model of the CPU used by the WN (WN specification). This parameter is a string whose domain is not defined yet in the GLUE Schema. The value used for Pentium III is "PIII".

CE_CPU_SPEED : Clock frequency in Mhz (WN specification).

CE_CPU_VENDOR : Vendor of the CPU. used by the WN (WN specification). This parameter is a string whose domain is not defined yet in the GLUE Schema. The value used for Intel is “intel”.

CE_HOST : Computing Element Hostname.

CE_INBOUNDIP : TRUE if inbound connectivity is enabled at your site, FALSE otherwise (WN specification).

CE_MINPHYSMEM : RAM size in kblocks (WN specification).

CE_MINVIRTMEM : Virtual Memory size in kblocks (WN specification).

CE_OS : Operating System name (WN specification).

CE_OS_RELEASE : Operating System release (WN specification).

CE_OUTBOUNDIP : TRUE if outbound connectivity is enabled at your site, FALSE otherwise (WN specification).

CE_RUNTIMEENV : List of software tags supported by the site. The list can include VO-specific software tags. In order to assure backward compatibility it should include the entry 'LCG-2', the current middleware version and the list of previous middleware tags.

CE_SF00 : Performance index of your fabric in SpecFloat 2000 (WN specification). For some examples of Spec values see http://www.specbench.org/osg/cpu2000/results/cint2000.html.

CE_SI00 : Performance index of your fabric in SpecInt 2000 (WN specification). For some examples of Spec values see http://www.specbench.org/osg/cpu2000/results/cint2000.html.

CE_SMPSIZE : Number of cpus in an SMP box (WN specification).

CLASSIC_HOST : The name of your SE_classic host.

CLASSIC_STORAGE_DIR : The root storage directory on CLASSIC_HOST.

DCACHE_ADMIN : Host name of the server node which manages the pool of nodes.

DPMDATA : Directory where the data is stored (absolute path, e.g./storage).

DPM_HOST : Host name of the DPM host, used also as a default DPM for the lcg-stdout-mon .

GRIDICE_SERVER_HOST : GridIce server host name (usually run on the MON node).

GRID_TRUSTED_BROKERS : List of the DNs of the Resource Brokers host certificates which are trusted by the Proxy node (ex: /O=Grid/O=CERN/OU=cern.ch/CN=host/testbed013.cern.ch).

INSTALL_ROOT : Installation root - change if using the re-locatable distribution.

JOB_MANAGER : The name of the job manager used by the gatekeeper.

LFC_CENTRAL : A list of VOs for which the LFC should be configured as a central catalogue.

LFC_HOST : Set this if you are building an LFC_HOST, not if you're just using clients.

LFC_LOCAL : Normally the LFC will support all VOs in the VOS variable. If you want to limit this list, add the ones you need to LFC_LOCAL. For each item listed in the VOS variable you need to create a set of new variables as follows:

VO_<VO-NAME>_QUEUES : The queues that the VO can use on the CE.

VO_<VO-NAME>_SE : Default SE used by the VO. WARNING: VO-NAME must be in capital cases.

VO_<VO-NAME>_STORAGE_DIR : Mount point on the Storage Element for the VO. WARNING: VO-NAME must be in capital cases.

VO_<VO-NAME>_SW_DIR : Area on the WN for the installation of the experiment software. If on the WNs a predefined shared area has been mounted where VO managers can pre-install software, then these variable should point to this area. If instead there is not a shared area and each job must install the software, then this variables should contain a dot (.).Anyway the mounting of shared areas, as well as the local installation of VO software is not managed by

yaim and should be handled locally by Site Administrators. WARNING: VO-NAME must be in capital cases.

PX_HOST : PX hostname.

QUEUES : The name of the queues for the CE. These are by default set as the VO names.

RB_HOST : Resource Broker Hostname.

SE_LIST : A list of hostnames of the SEs available at your site.

SITE_EMAIL : The e-mail address as published by the information system.

SITE_LAT : Site latitude.

SITE_LOC : "City, Country".

SITE_LONG : Site longitude.

SITE_NAME : Your GIIS.

SITE_SUPPORT_SITE : Support entry point ; Unique Id for the site in the GOC DB and information system.

SITE_TIER : Site tier.

SITE_WEB : Site site.

TORQUE_SERVER : Set this if your torque server is on a different host from the CE. It is ingored for other batch systems.

VOBOX_HOST : VOBOX hostname.

VOBOX_PORT : The port the VOBOX gsisshd listens on.

VOS : List of supported VOs.

VO_SW_DIR : Directory for installation of experiment software.

The original code of the function can be found in:

```
/opt/lcg/yaim/functions/config_gip
```

The code is also reproduced in 27.13..

16. SET-UP GLOBUS DAEMONS

Author(s): Vidic,Valentin
Email : support-lcg-manual-install@cern.ch

This chapter describes the configuration steps done by the *yaim* function '*config_globus*'.

The Globus configuration file */etc/globus.conf* is parsed by Globus daemon startup scripts to locate the Globus root directory and other global/daemon specific properties. The contents of the configuration file depend on the type of the node. The following table contains information on daemon to node mapping:

node/daemon	MDS	GridFTP	Gatekeeper
CE	yes	yes	yes
VOBOX	yes	yes	yes
SE_*	yes	yes	no
SE_dpm	yes	no	no
PX	yes	no	no
RB	yes	no	no
LFC	yes	no	no
GridICE	yes	no	no

Note that SE_dpm does not run standard GridFTP server, but a specialized DPM version.

The configuration file is divided into sections:

common Defines Globus installation directory, host certificates, location of gridmap file etc.

mds Defines information providers.

gridftp Defines the location of the GridFTP log file.

gatekeeper Defines jobmanagers and their parameters.

Logrotate scripts *globus-gatekeeper* and *gridftp* are installed in */etc/logrotate.d/*.

Globus initialization script (*<INSTALL_DIR>/globus/sbin/globus-initialization.sh*) is run next.

Finally, the appropriate daemons (*globus-mds*, *globus-gatekeeper*, *globus-gridftp*, *lcg-mon-gridftp*) are started (and configured to start on boot).

16.1. SPECIFICATION OF FUNCTION: CONFIG_GLOBUS

The function *'config_globus'* needs the following variables to be set in the configuration file:

CE_HOST : Computing Element Hostname.

GRIDICE_SERVER_HOST : GridIce server host name (usually run on the MON node).

INSTALL_ROOT : Installation root - change if using the re-locatable distribution.

JOB_MANAGER : The name of the job manager used by the gatekeeper.

PX_HOST : PX hostname.

RB_HOST : Resource Broker Hostname.

SITE_NAME : Your GIIS.

The original code of the function can be found in:

```
/opt/lcg/yaim/functions/config_globus
```

The code is reproduced also in 27.14..

17. SET-UP GRIDICE AGENT

Author(s): Retico,Antonio
Email : support-lcg-manual-install@cern.ch

This chapter describes the configuration steps done by the *yaim* function '*config_fmon_client*'.

The LCG nodes can produce data for the GridICE monitoring system.
The data are then sent to a collector server node which will then be queried by the LCG central GridICE monitoring service.

If you are running agents on the nodes (data producers), you should also run a GridICE collector server to collect information from your agents.

In the default LCG-2 configuration the MON node runs the GridICE collector node.

Before going forward with configuration, please assure the following RPMs to be installed (they should have been distributed with the node RPMs).

edg-fabricMonitoring
edt_sensor

In order to enable GridICE agent on a LCG node:

- Create and configure the file */opt/edg/var/etc/edg-fmon-agent.conf* as follows:

```
# Sensor file for edg-fmonagent
MSA




Transport



UDP
Server  <GRIDICE_SERVER_HOST>
Port 12409
FilterMetrics KeepOnly
11001
11011
11021
11101
11202
11013
11022
11031
11201
```

```
10100
10101
10102
10103
10104
10105



Sensors

edtproc
CommandLine /opt/edt/monitoring/bin/GLUEsensorLinuxProc
MetricClasses
edt.uptime
edt.cpu
edt.memory
edt.disk
edt.network
edt.ctxint
edt.swap
edt.processes
edt.sockets
edt.cpuinfo
edt.os
edt.alive
edt.regfiles

sensor1
CommandLine $(EDG_LOCATION)/libexec/edg-fmon-sensor-systemCheck
MetricClasses
executeScript


Metrics
11001
MetricClass edt.uptime
11011
MetricClass edt.cpu
11021
MetricClass edt.memory
11101
MetricClass edt.disk
11202
MetricClass edt.network
Parameters
interface eth0
11013
MetricClass edt.ctxint
11022
MetricClass edt.swap
11031
MetricClass edt.processes
```

```
11201
MetricClass edt.sockets
10100
MetricClass edt.cpuinfo
10101
MetricClass edt.os
10102
MetricClass edt.alive
10103
MetricClass edt.regfiles
10104
MetricClass executeScript
Parameters
command /opt/edt/monitoring/bin/CheckDaemon.pl --cfg /opt/edt/monitoring/etc/gridice-role.cfg
10105
MetricClass executeScript
Parameters
command /opt/edt/monitoring/bin/PoolDir.pl



Samples
verylowfreq
Timing 3600 0
Metrics
10100
10101
lowfreq
Timing 1800 0
Metrics
11001
proc0
Timing 30 0
Metrics
10102
proc1
Timing 60 0
Metrics
11011
11021
11101
11202
11013
11022
11031
11201
proc2
Timing 300 0
Metrics
10103
10105
proc3
Timing 120 0
Metrics
```

```
10104
```

WARNING: be very careful not to use <SPACE> characters to indent lines in this configuration file. Use <TAB> (or nothing) instead. The edg-fmon-agent does not allow spaces at the beginning of a row in the configuration file.

The parameter **<GRIDICE_SERVER_HOST>** is the complete hostname of the node that runs the GridICE collector server and publishes the data on the information system. The collector node will have to run a plain GRIS for this.

The information is sent to the collector node via UDP (port 12409).

- start the GridICE agent

```
> chkconfig edg-fmon-agent on
> service edg-fmon-agent stop
> service edg-fmon-agent start
```

17.1. SPECIFICATION OF FUNCTION: CONFIG_FMON_CLIENT

The function *'config_fmon_client'* needs the following variables to be set in the configuration file:

BATCH_LOG_DIR : Your batch system log directory.

CE_BATCH_SYS : Implementation of site batch system. Available values are “torque”, “lsf”, “pbs”, “condor” etc.

CE_HOST : Computing Element Hostname.

CRON_DIR : Yaim writes all cron jobs to this directory. Change it if you want to turn off Yaim’s management of cron.

GRIDICE_SERVER_HOST : GridIce server host name (usually run on the MON node).

INSTALL_ROOT : Installation root - change if using the re-locatable distribution.

MON_HOST : MON Box Hostname.

MY_DOMAIN : site’s domain name.

The original code of the function can be found in:

```
/opt/lcg/yaim/functions/config_fmon_client
```

The code is also reproduced in 27.15..

18. SET-UP LCG ENVIRONMENT VARIABLES

Author(s): Retico,Antonio
Email : support-lcg-manual-install@cern.ch

This chapter describes the configuration steps done by the *yaim* function '*config_lcgenv*'.

The LCG middleware needs some environment variables to be set up at boot time. The variable should be available both in 'bash-like' shells and in 'csh-like' shells.

This can be obtained in different ways:

The simplest way, if you have 'root' permissions, is to put a shell script for each of the supported shell 'families' in the directory */etc/profile.d*. The script will be automatically sourced at start up.

If instead you are not a superuser and you are doing the installation in a private directory (e.g. you are installing a Re-locatable Distribution of a Worker Node or a User Interface in the *<INSTALL_ROOT>* directory), you could create the scripts in the directory *<INSTALL_ROOT>/etc/profile.d*, in order to have the variables automatically set up by LCG tools.

The list of the environment variables to be set up follows:

LCG_GFAL_INFOSYS: Hostname of the BDII node.

MYPROXY_SERVER: Hostname of the Proxy server.

PATH: Add to the PATH variable the path */opt/d-cache-client/bin*

LD_LIBRARY_PATH: Add to the LD_LIBRARY_PATH variable the path */opt/d-cache-client/dcap*

SRM_PATH: Installation directory of the srm client. The default value for this variable is */opt/d-cache-client/srm*

VO_<VO-NAME>_SW_DIR: For each virtual organization <VO-NAME> An environment variable VO_<VO-NAME>_SW_DIR is needed. This variable points to the installation directory of the VO software.

VO_<VO-NAME>_DEFAULT_SE: For each virtual organization <VO-NAME> An environment variable VO_<VO-NAME>_DEFAULT_SE is needed. This variable points to the Default Storage Element for that VO.

The examples given hereafter refer to the simple configuration method described above. In the following description we will refer to the two possible locations as to the <LCG_ENV_LOC>.
So, according to the cases above described, either

<LCG_ENV_LOC>=/etc/profile.d

or

<LCG_ENV_LOC>=<INSTALL_ROOT>/etc/profile.d

Examples:

- Example of file *<LCG_ENV_LOC>/lcgenv.sh*:

```
#!/bin/sh
export LCG_GFAL_INFOSYS=lxb1769.cern.ch:2170
export MYPROXY_SERVER=lxb1774.cern.ch
export PATH="${PATH}:/opt/d-cache-client/bin"
export LD_LIBRARY_PATH=${LD_LIBRARY_PATH}:/opt/d-cache-client/dcap
export SRM_PATH=/opt/d-cache-client/srm
export VO_ATLAS_SW_DIR=lxb1780.cern.ch
export VO_ATLAS_DEFAULT_SE=lxb1780.cern.ch
export VO_ALICE_SW_DIR=lxb1780.cern.ch
export VO_ALICE_DEFAULT_SE=lxb1780.cern.ch
export VO_LHCB_SW_DIR=lxb1780.cern.ch
export VO_LHCB_DEFAULT_SE=lxb1780.cern.ch
export VO_CMS_SW_DIR=lxb1780.cern.ch
export VO_CMS_DEFAULT_SE=lxb1780.cern.ch
export VO_DTEAM_SW_DIR=lxb1780.cern.ch
export VO_DTEAM_DEFAULT_SE=lxb1780.cern.ch
```

- Example of file *<LCG_ENV_LOC>/lcgenv.csh*:

```
#!/bin/csh
setenv LCG_GFAL_INFOSYS lxb1769.cern.ch:2170
setenv MYPROXY_SERVER lxb1774.cern.ch
setenv PATH "${PATH}:/opt/d-cache-client/bin"
setenv LD_LIBRARY_PATH ${LD_LIBRARY_PATH}:/opt/d-cache-client/dcap
setenv SRM_PATH /opt/d-cache-client/srm
setenv VO_ATLAS_SW_DIR lxb1780.cern.ch
setenv VO_ATLAS_DEFAULT_SE lxb1780.cern.ch
setenv VO_ALICE_SW_DIR lxb1780.cern.ch
setenv VO_ALICE_DEFAULT_SE lxb1780.cern.ch
setenv VO_LHCB_SW_DIR lxb1780.cern.ch
setenv VO_LHCB_DEFAULT_SE lxb1780.cern.ch
setenv VO_CMS_SW_DIR lxb1780.cern.ch
setenv VO_CMS_DEFAULT_SE lxb1780.cern.ch
setenv VO_DTEAM_SW_DIR lxb1780.cern.ch
setenv VO_DTEAM_DEFAULT_SE lxb1780.cern.ch
```

WARNING: The two scripts must be executable by all users.

```
> chmod a+x ${LCG_ENV_LOC}/lcgenv.csh
> chmod a+x ${LCG_ENV_LOC}/lcgenv.sh
```

18.1. SPECIFICATION OF FUNCTION: CONFIG_LCGENV

The function *'config_lcgenv'* needs the following variables to be set in the configuration file:

BDII_HOST : BDII Hostname.

CE_HOST : Computing Element Hostname.

DPM_HOST : Host name of the DPM host, used also as a default DPM for the lcg-stdout-mon .

EDG_WL_SCRATCH : Set this if you want jobs to use a particular scratch area.

EDG_WL_SCRATCH : Set this if you want jobs to use a particular scratch area.

GLOBUS_TCP_PORT_RANGE : Port range for Globus IO.

GSSKLOG : yes or no, indicating whether the site provides an AFS authentication server which maps gsi credentials into Kerberos tokens .

GSSKLOG_SERVER : If GSSKLOG is yes, the name of the AFS authentication server host.

INSTALL_ROOT : Installation root - change if using the re-locatable distribution.

PX_HOST : PX hostname.

SE_LIST : A list of hostnames of the SEs available at your site.

SITE_NAME : Your GIIS.

VOBOX_HOST : VOBOX hostname.

VOS : List of supported VOs. For each item listed in the VOS variable you need to create a set of new variables as follows:

VO_<VO-NAME>_SE : Default SE used by the VO. WARNING: VO-NAME must be in capital cases.

VO_<VO-NAME>_SW_DIR : Area on the WN for the installation of the experiment software. If on the WNs a predefined shared area has been mounted where VO managers can pre-install software, then these variable should point to this area. If instead there is not a shared area and each job must install the software, then this variables should contain a dot (.).Anyway the mounting of shared areas, as well as the local installation of VO software is not managed by *yaim* and should be handled locally by Site Administrators. WARNING: VO-NAME must be in capital cases.

The function does exit with return code 1 if they are not set.

The original code of the function can be found in:

```
/opt/lcg/yaim/functions/config_lcgenv
```

The code is reproduced also in 27.16..

19. SET-UP BDII

Author(s): Vidic,Valentin
Email : support-lcg-manual-install@cern.ch

This chapter describes the configuration steps done by the *yaim* function '*config_bdii*'.

This functions is used to configure two types of BDIIs: site BDII and top BDDI.

At each site, a site BDII collects information about all resources present at a site (i.e. data from all GRISes of the site).

A top BDII collects all information coming from site BDIIs and stores them in a permanent database.

BDII is configured in *<INSTALL_ROOT>/bdii/etc/bdii.conf*. For top BDII the configuration is:

```
BDII_PORT_READ=2170
BDII_PORTS_WRITE="2171 2172 2173"
BDII_USER=edguser
BDII_BIND=mds-vo-name=local,o=grid
BDII_PASSWD=<random_password>
BDII_SEARCH_FILTER='*'
BDII_SEARCH_TIMEOUT=30
BDII_BREATHE_TIME=60
BDII_AUTO_UPDATE=yes
BDII_AUTO_MODIFY=no
BDII_DIR=<INSTALL_ROOT>/bdii/
BDII_UPDATE_URL=<BDII_HTTP_URL>
BDII_UPDATE_LDIF=http://
SLAPD=/usr/sbin/slapd
SLAPADD=/usr/sbin/slapadd
```

BDII is setup with the base DN of *mds-vo-name=local,o=grid* and automatic generation of *<INSTALL_ROOT>/bdii/e*
update.conf by downloading from <BDII_HTTP_URL>. BDII password is generated using *mkpasswd* or from *RANDOM* shell variable.

For local BDII the configuration is:

```
BDII_PORT_READ=2170
BDII_PORTS_WRITE="2171 2172 2173"
BDII_USER=edguser
BDII_BIND=mds-vo-name=<SITE_NAME>,o=grid
BDII_PASSWD=<random_password>
BDII_SEARCH_FILTER='*'
BDII_SEARCH_TIMEOUT=30
BDII_BREATHE_TIME=60
BDII_AUTO_UPDATE=no
BDII_AUTO_MODIFY=no
BDII_DIR=<INSTALL_ROOT>/bdii/
BDII_UPDATE_URL=<BDII_HTTP_URL>
BDII_UPDATE_LDIF=http://
```

```
SLAPD=/usr/sbin/slapd
SLAPADD=/usr/sbin/slapadd
```

Base DN is set to *mds-vo-name=<SITE_NAME>,o=grid*. *<INSTALL_ROOT>/bdii/etc/bdii-update.conf* is created from the values of variables <BDII_REGIONS> and <BDII_<region>_URL>.

LDAP schemas are configured in *<INSTALL_ROOT>/bdii/etc/schemas*:

```
/etc/openldap/schema/core.schema
/opt/glue/schema/ldap/Glue-CORE.schema
/opt/glue/schema/ldap/Glue-CE.schema
/opt/glue/schema/ldap/Glue-CESEBind.schema
/opt/glue/schema/ldap/Glue-SE.schema
/opt/lcg/schema/ldap/SiteInfo.schema
```

BDII service is restarted and configured to start on boot.

19.1. SPECIFICATION OF FUNCTION: CONFIG_BDII

The function *'config_bdii'* needs the following variables to be set in the configuration file:

BDII_FCR : Set the URL of the Freedom of Choice for Rescources URL.

BDII_HOST : BDII Hostname.

BDII_HTTP_URL : URL pointing to the BDII configuration file.

BDII_REGIONS : List of node types publishing information on the bdii. For each item listed in the BDII_REGIONS variable you need to create a set of new variables as follows:

BDII_<REGION>_URL : URL of the information producer (e.g.: BDII_CE_URL="URL of the CE information producer", BDII_SE_URL="URL of the SE information producer".

INSTALL_ROOT : Installation root - change if using the re-locatable distribution.

SITE_NAME : Your GIIS.

The original code of the function can be found in:

```
/opt/lcg/yaim/functions/config_bdii
```

The code is also reproduced in 27.17..

20. SET-UP WORKLOAD MANAGER ENVIRONMENT

Author(s): Vidic,Valentin
Email : support-lcg-manual-install@cern.ch

This chapter describes the configuration steps done by the *yaim* function '*config_workload_manager_env*'.

<INSTALL_ROOT>/edg/etc/profile.d/edg-wl.csh and *<INSTALL_ROOT>/edg/etc/profile.d/edg-wl.sh* are copied to *<INSTALL_ROOT>/edg/var/etc/profile.d/*.

They are sourced into the user environment upon login (*/etc/profile.d/z_edg-profile.(c)sh*).

20.1. SPECIFICATION OF FUNCTION: CONFIG_WORKLOAD_MANAGER_ENV

The function *'config_workload_manager_env'* needs the following variables to be set in the configuration file:

INSTALL_ROOT : Installation root - change if using the re-locatable distribution.

The original code of the function can be found in:

```
/opt/lcg/yaim/functions/config_workload_manager_env
```

The code is also reproduced in 27.18..

21. SET-UP WORKLOAD MANAGER LOGGING DAEMONS

Author(s): Vidic,Valentin
Email : support-lcg-manual-install@cern.ch

This chapter describes the configuration steps done by the *yaim* function '*config_wm_locallogger*'.

/etc/rc.d/init.d/edg-wl-locallogger is restarted and configured to start on boot. Cron job is installed to periodically renew proxy certificates locallogger daemons use.

21.1. SPECIFICATION OF FUNCTION: CONFIG_WM_LOCALLOGGER

The function *'config_wm_locallogger'* needs the following variables to be set in the configuration file:

INSTALL_ROOT : Installation root - change if using the re-locatable distribution.

The original code of the function can be found in:

```
/opt/lcg/yaim/functions/config_wm_locallogger
```

The code is also reproduced in 27.19..

22. SET-UP APEL LOG PARSER

Author(s): Vidic,Valentin
Email : support-lcg-manual-install@cern.ch

This chapter describes the configuration steps done by the *yaim* function '*config_apel_pbs*'.

<INSTALL_ROOT>/glite/etc/glite-apel-pbs/parser-config.xml is copied to *<INSTALL_ROOT>/glite/etc/glite-apel-pbs/parser-config-yaim.xml*. The new file is then updated with the values of <MON_HOST>, <APEL_DB_PASSWORD>, <SITE_NAME> and <CE_HOST>.

Finally, cron job is installed to parse PBS, Gatekeeper and messages log files once a day looking for records of finished jobs. Found records are saved in a MySQL database running on <MON_HOST>.

22.1. SPECIFICATION OF FUNCTION: CONFIG_APEL_PBS

The function *'config_apel_pbs'* needs the following variables to be set in the configuration file:

APEL_DB_PASSWORD : database password for apel.

CE_HOST : Computing Element Hostname.

CRON_DIR : Yaim writes all cron jobs to this directory. Change it if you want to turn off Yaim's management of cron.

INSTALL_ROOT : Installation root - change if using the re-locatable distribution.

MON_HOST : MON Box Hostname.

SITE_NAME : Your GIIS.

TORQUE_SERVER : Set this if your torque server is on a different host from the CE. It is ingored for other batch systems.

The original code of the function can be found in:

```
/opt/lcg/yaim/functions/config_apel_pbs
```

The code is also reproduced in 27.20..

23. SET-UP LCMAPS

Author(s): Vidic,Valentin
Email : support-lcg-manual-install@cern.ch

This chapter describes the configuration steps done by the *yaim* function '*config_lcmaps*'.

Local Credential Mapping Service (LCMAPS) determines the mapping of grid user credentials (proxy certificates) to local (Unix) accounts and groups. LCMAPS is configured by creating *<INSTALL_ROOT>/edg/etc/lcmaps/lcmaps.db*. Modules and two policies are defined:

```
# where to look for modules
path = ${INSTALL_ROOT}/edg/lib/lcmaps/modules

# module definitions
posixenf = "lcmaps_posix_enf.mod -maxuid 1 -maxpgid 1 -maxsgid 32 "
localaccount = "lcmaps_localaccount.mod  -gridmapfile /etc/grid-security/grid-mapfile"
poolaccount = "lcmaps_poolaccount.mod  -gridmapfile /etc/grid-security/grid-mapfile -gridmapdir /etc/grid-security/
vomsextract = "lcmaps_voms.mod  -vomsdir /etc/grid-security/vomsdir/ -certdir /etc/grid-security/certificates/"
vomslocalgroup = "lcmaps_voms_localgroup.mod  -groupmapfile ${INSTALL_ROOT}/edg/etc/lcmaps/groupmapfile -mapmin 0"
vomspoolaccount = "lcmaps_voms_poolaccount.mod  -gridmapfile ${INSTALL_ROOT}/edg/etc/lcmaps/gridmapfile -gridmapdir
vomslocalaccount = "lcmaps_voms_localaccount.mod  -gridmapfile ${INSTALL_ROOT}/edg/etc/lcmaps/gridmapfile -use_voms

# policies
voms:
vomsextract -> vomslocalgroup
vomslocalgroup -> vomspoolaccount
vomspoolaccount -> posixenf | vomslocalaccount
vomslocalaccount -> posixenf

standard:
localaccount -> posixenf | poolaccount
poolaccount -> posixenf
```

If the user's proxy certificate contains VOMS attributes, they are used to map the user to local account and group. Otherwise, proxy certificate subject is used to map the user according to */etc/grid-security/grid-mapfile*.

23.1. SPECIFICATION OF FUNCTION: CONFIG_LCMAPS

The function *'config_lcmaps'* needs the following variables to be set in the configuration file:

INSTALL_ROOT : Installation root - change if using the re-locatable distribution.

The original code of the function can be found in:

```
/opt/lcg/yaim/functions/config_lcmaps
```

The code is also reproduced in 27.21..

24. SET-UP LCAS

Author(s): Vidic,Valentin
Email : support-lcg-manual-install@cern.ch

This chapter describes the configuration steps done by the *yaim* function '*config_lcas*'.

Local Centre Authorization Service (LCAS) handles authorization requests to the local resource. LCAS is configured by creating the file *<INSTALL_ROOT>/edg/etc/lcas/lcas.db*. By default the following authorization plugins are enabled:

```
pluginname=lcas_userban.mod,pluginargs=ban_users.db
pluginname=lcas_timeslots.mod,pluginargs=timeslots.db
pluginname=lcas_plugin_example.mod,pluginargs=arguments
```

lcas_userban.mod module can be used to ban specific users from the resource. It is configured in *<INSTALL_ROOT>/edg/etc/lcas/ ban_users.db* with an empty (no users banned) configuration.

lcas_timeslots.mod module can be used disable access to the resource at certain times of day/week/month. It is configured in *<INSTALL_ROOT>/edg/etc/lcas/timeslots.db* with a default (always open) configuration.

lcas_plugin_example.mod is an dummy plugin that can be used for creating site specific authorization plugins.

24.1. SPECIFICATION OF FUNCTION: CONFIG_LCAS

The function *'config_lcas'* needs the following variables to be set in the configuration file:

INSTALL_ROOT : Installation root - change if using the re-locatable distribution.

The original code of the function can be found in:

```
/opt/lcg/yaim/functions/config_lcas
```

The code is also reproduced in 27.22..

25. SET-UP SSH HOSTBASED AUTHENTICATION

Author(s): Vidic,Valentin
Email : support-lcg-manual-install@cern.ch

This chapter describes the configuration steps done by the *yaim* function '*config_torque_submitter_ssh*'.

/var/spool/pbs/server_name is initialized with the hostname where the Torque server is running (usually CE, otherwise set in <TORQUE_SERVER>). This file is used by Torque utils for contacting the server. This is neccessary because *pbsnodes* will be called latter in the script.

<INSTALL_ROOT>/edg/etc/edg-pbs-shostsequiv.conf is created with the following contents:

```
# Example configuration file for the edg-pbs-shostsequiv script
# File where the list of nodes will be written
SHOSTSEQUIV = /etc/ssh/shosts.equiv
# List of nodes to be included in the SHOSTSEQUIV file even if not reported
# by the pbsnodes command
NODES       =
# Location of the pbsnodes command
PBSBIN      = /usr/bin
```

edg-pbs-shostsequiv is run once to initialize */etc/ssh/shosts.equiv* and configured to run from cron four times a day. *edg-pbs-shostsequiv* uses *pbsnodes* and this config file to acquire the list of nodes. For every node not already present in */etc/ssh/shosts.equiv*, its hostname is appended.

<INSTALL_ROOT>/edg/etc/edg-pbs-knownhosts.conf is created with the following contents:

```
NODES = <CE_HOST> <SE_HOST>
PBSBIN = /usr/bin
KEYTYPES = rsa1,rsa,dsa
KNOWNHOSTS = /etc/ssh/ssh_known_hosts
```

edg-pbs-knownhosts is run once to initialize */etc/ssh/ssh_known_hosts* and configured to run from cron four times a day. *edg-pbs-knownhosts* uses *pbsnodes* and this config file to acquire the list of nodes. For every node not already present in */etc/ssh/ssh_known_hosts*, host keys are discovered using *ssh-keyscan* and appended to */etc/ssh/ssh_known_hosts*.

Hostbased authentication is enabled in */etc/ssh/sshd_config* by adding these lines:

```
HostbasedAuthentication yes
IgnoreUserKnownHosts yes
IgnoreRhosts yes
```

SSH daemon is reloaded with the new configuration.

Client connections are now authenticated by checking if the client hostname is listed in */etc/hosts.equiv* or */etc/ssh/shosts.equiv* and if the correct client host key is listed in */etc/ssh/ssh_known_hosts*.

25.1. SPECIFICATION OF FUNCTION: CONFIG_TORQUE_SUBMITTER_SSH

The function *'config_torque_submitter_ssh'* needs the following variables to be set in the configuration file:

CE_HOST : Computing Element Hostname.

INSTALL_ROOT : Installation root - change if using the re-locatable distribution.

QUEUES : The name of the queues for the CE. These are by default set as the VO names.

SE_LIST : A list of hostnames of the SEs available at your site.

TORQUE_SERVER : Set this if your torque server is on a different host from the CE. It is ingored for other batch systems.

WN_LIST : Path to the list of Worker Nodes. The list of Worker Nodes is a file to be created by the Site Administrator, which contains a plain list of the batch nodes. An example of this configuration file is given in /opt/lcg/yaim/examples/wn-list.conf.

The original code of the function can be found in:

```
/opt/lcg/yaim/functions/config_torque_submitter_ssh
```

The code is also reproduced in 27.23..

26. SET-UP TORQUE SERVER

Author(s): Vidic,Valentin
Email : support-lcg-manual-install@cern.ch

This chapter describes the configuration steps done by the *yaim* function '*config_torque_server*'.

Torque and MAUI service ports are defined by adding the following lines to */etc/services*:

```
pbs             15001/tcp
pbs_mom         15002/tcp
pbs_resmon      15003/tcp
pbs_resmon      15003/udp
maui            15004/tcp
```

/var/spool/pbs/server_name is initialized with the name of the host running Torque server. If not defined differently with <TORQUE_SERVER>, this is assumed to be CE.

Nodes are defined by creating */var/spool/pbs/server_priv/nodes* from the list of hostnames in <WN_LIST>. For each of the nodes the following line is added:

```
<node> np=<CE_SMPSIZE> lcgpro
```

The rest of the configuration is done through *qmgr* so Torque server is started. First some general server parameters are set:

```
set server scheduling = True
set server acl_host_enable = False
set server managers = root@<hostname>
set server operators = root@<hostname>
set server default_queue = dteam
set server log_events = 511
set server mail_from = adm
set server query_other_jobs = True
set server scheduler_iteration = 600
set server default_node = lcgpro
set server node_pack = False
```

Queues listed in <QUEUES> are created (if they don't already exist). For each of the queues the following parameters are set:

```
set queue <queue> queue_type = Execution
set queue <queue> resources_max.cput = 48:00:00
set queue <queue> resources_max.walltime = 72:00:00
set queue <queue> enabled = True
set queue <queue> started = True
set queue <queue> acl_group_enable = True
```

Queue ACL is set to allow access to VOs that list the queue in <VO_<vo>_QUEUES>:

```
set queue <queue> acl_groups += <vo>
```

Torque server is then restarted in order to save the new configuration.

As Torque server logs (*/var/spool/pbs/server_logs*) can get quite big, a cron job is installed to compress them once a day.

If Torque server is not running on CE, hostname of CE needs to be added to */etc/hosts.equiv* in order for the job submission to work.

MAUI scheduler is initialized with the following default configuration (*/var/spool/maui/maui.cfg*):

```
# MAUI configuration example

SERVERHOST              <hostname>
ADMIN1                  root
ADMINHOST               <hostname>
RMCFG[base]             TYPE=PBS
SERVERPORT              40559
SERVERMODE              NORMAL

# Set PBS server polling interval. If you have short
# queues or/and jobs it is worth to set a short interval. (10 seconds)

RMPOLLINTERVAL        00:00:10

# a max. 10 MByte log file in a logical location

LOGFILE               /var/log/maui.log
LOGFILEMAXSIZE        10000000
LOGLEVEL              1

# Set the delay to 1 minute before Maui tries to run a job again,
# in case it failed to run the first time.
# The default value is 1 hour.

DEFERTIME       00:01:00
```

Finally, MAUI is restarted and configured to start on boot.

26.1. SPECIFICATION OF FUNCTION: CONFIG_TORQUE_SERVER

The function *'config_torque_server'* needs the following variables to be set in the configuration file:

CE_HOST : Computing Element Hostname.

CE_SMPSIZE : Number of cpus in an SMP box (WN specification).

INSTALL_ROOT : Installation root - change if using the re-locatable distribution.

QUEUES : The name of the queues for the CE. These are by default set as the VO names.

TORQUE_SERVER : Set this if your torque server is on a different host from the CE. It is ingored for other batch systems.

VOS : List of supported VOs. For each item listed in the VOS variable you need to create a set of new variables as follows:

VO_<VO-NAME>_QUEUES : The queues that the VO can use on the CE.

WN_LIST : Path to the list of Worker Nodes. The list of Worker Nodes is a file to be created by the Site Administrator, which contains a plain list of the batch nodes. An example of this configuration file is given in /opt/lcg/yaim/examples/wn-list.conf.

The original code of the function can be found in:

```
/opt/lcg/yaim/functions/config_torque_server
```

The code is also reproduced in 27.24..

27. SOURCE CODE

27.1. CONFIG_LDCONF

```
config_ldconf () {

    INSTALL_ROOT=${INSTALL_ROOT:-/opt}

cp -p /etc/ld.so.conf /etc/ld.so.conf.orig

    LIBDIRS="${INSTALL_ROOT}/globus/lib \
     ${INSTALL_ROOT}/edg/lib \
             ${INSTALL_ROOT}/edg/externals/lib/ \
     /usr/local/lib \
             ${INSTALL_ROOT}/lcg/lib \
             /usr/kerberos/lib \
             /usr/X11R6/lib \
             /usr/lib/qt-3.1/lib \
             ${INSTALL_ROOT}/gcc-3.2.2/lib \
             ${INSTALL_ROOT}/glite/lib \
             ${INSTALL_ROOT}/glite/externals/lib"


    if [ -f /etc/ld.so.conf.add ]; then
rm -f /etc/ld.so.conf.add
    fi

    for libdir in ${LIBDIRS}; do
if  ( ! grep -q $libdir /etc/ld.so.conf && [ -d $libdir ] ); then
    echo $libdir >> /etc/ld.so.conf.add
fi
    done

    if [ -f /etc/ld.so.conf.add ]; then
sort -u /etc/ld.so.conf.add >> /etc/ld.so.conf
rm -f  /etc/ld.so.conf.add
    fi

    /sbin/ldconfig

    return 0
}
```

27.2. CONFIG_SYSCONFIG_EDG

```
config_sysconfig_edg(){

INSTALL_ROOT=${INSTALL_ROOT:-/opt}

cat <<EOF > /etc/sysconfig/edg
EDG_LOCATION=$INSTALL_ROOT/edg
```

```
EDG_LOCATION_VAR=$INSTALL_ROOT/edg/var
EDG_TMP=/tmp
X509_USER_CERT=/etc/grid-security/hostcert.pem
X509_USER_KEY=/etc/grid-security/hostkey.pem
GRIDMAP=/etc/grid-security/grid-mapfile
GRIDMAPDIR=/etc/grid-security/gridmapdir/
EDG_WL_BKSERVERD_ADDOPTS=--rgmaexport
EDG_WL_RGMA_FILE=/var/edgwl/logging/status.log
EOF

return 0
}
```

27.3. CONFIG_SYSCONFIG_GLOBUS

```
config_sysconfig_globus() {

INSTALL_ROOT=${INSTALL_ROOT:-/opt}

# If GLOBUS_TCP_PORT_RANGE is unset, give it a good default
# Leave it alone if it is set but empty
GLOBUS_TCP_PORT_RANGE=${GLOBUS_TCP_PORT_RANGE-"20000 25000"}

cat <<EOF > /etc/sysconfig/globus
GLOBUS_LOCATION=$INSTALL_ROOT/globus
GLOBUS_CONFIG=/etc/globus.conf
export LANG=C
EOF

# Set GLOBUS_TCP_PORT_RANGE, but not for nodes which are only WNs
if [ "$GLOBUS_TCP_PORT_RANGE" ] && ( ! echo $NODE_TYPE_LIST | egrep -q '^ *WN_?[[:alpha:]]* *$' ); then
    echo "GLOBUS_TCP_PORT_RANGE=\"$GLOBUS_TCP_PORT_RANGE\"" >> /etc/sysconfig/globus
fi

(
    # HACK to avoid complaints from services that do not need it,
    # but get started via a login shell before the file is created...

    f=$INSTALL_ROOT/globus/libexec/globus-script-initializer
    echo '' > $f
    chmod 755 $f
)

return 0
}
```

27.4. CONFIG_SYSCONFIG_LCG

```
config_sysconfig_lcg(){

INSTALL_ROOT=${INSTALL_ROOT:-/opt}
```

```
cat <<EOF > /etc/sysconfig/lcg
LCG_LOCATION=$INSTALL_ROOT/lcg
LCG_LOCATION_VAR=$INSTALL_ROOT/lcg/var
LCG_TMP=/tmp
export SITE_NAME=$SITE_NAME
EOF

return 0
}
```

27.5. CONFIG_CRL

```
config_crl(){

INSTALL_ROOT=${INSTALL_ROOT:-/opt}

let minute="$RANDOM%60"

let h1="$RANDOM%24"
let h2="($h1+6)%24"
let h3="($h1+12)%24"
let h4="($h1+18)%24"

if !( echo "${NODE_TYPE_LIST}" | grep TAR > /dev/null ); then

    if [ ! -f /etc/cron.d/edg-fetch-crl ]; then
echo "Now updating the CRLs - this may take a few minutes..."
$INSTALL_ROOT/edg/etc/cron/edg-fetch-crl-cron >> /var/log/edg-fetch-crl-cron.log 2>&1
    fi

cron_job edg-fetch-crl root "$minute $h1,$h2,$h3,$h4 * * * $INSTALL_ROOT/edg/etc/cron/edg-fetch-crl-cron >> /var/lc

    cat <<EOF > /etc/logrotate.d/edg-fetch
/var/log/edg-fetch-crl-cron.log  {
    compress
    monthly
    rotate 12
    missingok
    ifempty
    create
}
EOF

else

    cron_job edg-fetch-crl `whoami` "$minute $h1,$h2,$h3,$h4 * * * $INSTALL_ROOT/edg/etc/cron/edg-fetch-crl-cron >>
    if [ ! -d $INSTALL_ROOT/edg/var/log ]; then
mkdir -p $INSTALL_ROOT/edg/var/log
    fi
    echo "Now updating the CRLs - this may take a few minutes..."
    $INSTALL_ROOT/edg/etc/cron/edg-fetch-crl-cron >> $INSTALL_ROOT/edg/var/log/edg-fetch-crl-cron.log 2>&1
```

```
fi

return 0
}
```

27.6. CONFIG_RFIO

```
config_rfio() {

INSTALL_ROOT=${INSTALL_ROOT:-/opt}

# This function turns rfio on where necessary and
# just as important, turns it off where it isn't necessary

if ( echo "${NODE_TYPE_LIST}" | grep -q SE_classic ); then

    if [ "x`grep rfio /etc/services | grep tcp`" = "x"  ]; then
echo "rfio     5001/tcp" >> /etc/services
    fi

    if [ "x`grep rfio /etc/services | grep udp`" = "x"  ]; then
echo "rfio     5001/udp" >> /etc/services
    fi

    /sbin/service rfiod restart

elif ( echo "${NODE_TYPE_LIST}" | grep -q SE_dpm ); then

    return 0

elif (  rpm -qa | grep -q CASTOR-client ); then

    /sbin/service rfiod stop
    /sbin/chkconfig --level 2345 rfiod off

fi

return 0

}
```

27.7. CONFIG_HOST_CERTS

```
config_host_certs(){

if [ -f /etc/grid-security/hostkey.pem ]; then
    chmod 400 /etc/grid-security/hostkey.pem
elif  [ -f /etc/grid-security/hostcert.pem ]; then
    chmod 644 /etc/grid-security/hostcert.pem
else
```

```
    echo "Please copy the hostkey.pem and hostcert.pem to /etc/grid-security"
    return 1
fi
return 0
}
```

27.8. CONFIG_USERS

```
config_users(){
#
# Creates the Pool Users.
#
# Takes the users, groups and ids from a configuration file (USERS_CONF).
# File format:
#
# UserId:User:GroupId:Group
#

INSTALL_ROOT=${INSTALL_ROOT:-/opt}

requires USERS_CONF VOS

if [ ! -e $USERS_CONF ]; then
    echo "$USERS_CONF not found."
    return 1
fi

check_users_conf_format

# Add each group required by $VOS
awk -F: '{print $3, $4, $5}' ${USERS_CONF} | sort -u | while read gid groupname virtorg; do
    if ( [ "$virtorg" ] && echo $VOS | grep -w "$virtorg" > /dev/null ); then
groupadd -g $gid $groupname 2> /dev/null
    fi
done

grid_accounts=
newline='
'

# Add all the users for each VO in ${VOS}
for x in `cat $USERS_CONF`; do
    # ensure that this VO is in the $VOS list
    virtorg=`echo $x | cut -d":" -f5`
    if ( [ "$virtorg" ] && echo $VOS | grep -w "$virtorg" > /dev/null ); then
user=`echo $x | cut -d":" -f2`
id=`echo $x | cut -d":" -f1`
group=`echo $x | cut -d":" -f3`
if ( ! id $user > /dev/null 2>&1 ); then
    useradd -c "mapped user for group ID $group" -u $id -g $group $user
fi
```

```
# grid users shall not be able to submit at or cron jobs
for deny in /etc/at.deny /etc/cron.deny; do
    tmp=$deny.$$
    touch $deny
    (grep -v "^$user\$" $deny; echo "$user") > $tmp && mv $tmp $deny
done

grid_accounts="$grid_accounts$newline$user"
    fi
done

(
    cga=$INSTALL_ROOT/lcg/etc/cleanup-grid-accounts.conf
    cga_tmp=$cga.$$

    [ -r $cga ] || exit

    (
sed '/YAIM/,$d' $cga
echo "# next lines added by YAIM on `date`"
echo "ACCOUNTS='$grid_accounts$newline'"
    ) > $cga_tmp

    mv $cga_tmp $cga
)

let minute="$RANDOM%60"
let h="$RANDOM%6"
f=/var/log/cleanup-grid-accounts.log

if ( echo "${NODE_TYPE_LIST}" | grep '\<CE' > /dev/null ); then

    cron_job cleanup-grid-accounts root "$minute $h * * * \
$INSTALL_ROOT/lcg/sbin/cleanup-grid-accounts.sh -v -F >> $f 2>&1"

    cat <<EOF > /etc/logrotate.d/cleanup-grid-accounts
$f {
    compress
    daily
    rotate 30
    missingok
}
EOF

elif ( echo "${NODE_TYPE_LIST}" | grep '\<WN' > /dev/null ); then

    cron_job cleanup-grid-accounts root "$minute $h * * * \
$INSTALL_ROOT/lcg/sbin/cleanup-grid-accounts.sh -v >> $f 2>&1"

    cat <<EOF > /etc/logrotate.d/cleanup-grid-accounts
$f {
    compress
    daily
    rotate 30
```

```
    missingok
}
EOF

fi

return 0
}
```

27.9. CONFIG_EDGUSERS

```
config_edgusers(){

INSTALL_ROOT=${INSTALL_ROOT:-/opt}

check_users_conf_format

if ( ! id edguser > /dev/null 2>&1 ); then
    useradd -r -c "EDG User" edguser
    mkdir -p /home/edguser
    chown edguser:edguser /home/edguser
fi

if ( ! id edginfo > /dev/null 2>&1 ); then
    useradd -r -c "EDG Info user" edginfo
    mkdir -p /home/edginfo
    chown edginfo:edginfo /home/edginfo
fi

if ( ! id rgma > /dev/null 2>&1 ); then
    useradd -r -c "RGMA user" -m -d ${INSTALL_ROOT}/glite/etc/rgma rgma
fi

# Make sure edguser is a member of each group

awk -F: '{print $3, $4, $5}' ${USERS_CONF} | sort -u | while read gid groupname virtorg; do
    if ( [ "$virtorg" ] && echo $VOS | grep -w "$virtorg" > /dev/null ); then
# On some nodes the users are not created, so the group will not exist
# Isn't there a better way to check for group existance??
if ( grep "^${groupname}:" /etc/group > /dev/null ); then
    gpasswd -a edguser $groupname  > /dev/null
fi
    fi
done

return 0
}
```

27.10. CONFIG_MKGRIDMAP

```
config_mkgridmap(){
```

```
requires USERS_CONF GROUPS_CONF VOS

INSTALL_ROOT=${INSTALL_ROOT:-/opt}

if [ ! -e $USERS_CONF ]; then
    echo "$USERS_CONF not found."
    return 1
fi

if [ ! -e $GROUPS_CONF ]; then
    echo "$GROUPS_CONF not found."
    return 1
fi

check_users_conf_format

gmc=$INSTALL_ROOT/edg/etc/edg-mkgridmap.conf
gmd=/etc/grid-security/gridmapdir

mkdir -p $gmd
chown root:edguser $gmd
chmod 775 $gmd

for user in `awk -F: '$6==""{print $2}' $USERS_CONF`; do
    f=$gmd/$user
    [ -f $f ] || touch $f
done

if ( echo "${NODE_TYPE_LIST}" | egrep -q 'dpm|LFC' ); then
    gmc_dm=$INSTALL_ROOT/lcg/etc/lcgdm-mkgridmap.conf
else
    gmc_dm=/dev/null
fi

cat << EOF > $gmc
##############################################################################
#
# edg-mkgridmap.conf generated by YAIM on `date`
#
##############################################################################

EOF

cat << EOF > $gmc_dm
##############################################################################
#
# lcgdm-mkgridmap.conf generated by YAIM on `date`
#
##############################################################################

EOF

lcmaps=${INSTALL_ROOT}/edg/etc/lcmaps
```

```
lcmaps_gridmapfile=$lcmaps/gridmapfile
lcmaps_groupmapfile=$lcmaps/groupmapfile

mkdir -p $lcmaps
rm -f $lcmaps_gridmapfile $lcmaps_groupmapfile

for VO in `echo $VOS | tr '[:lower:]' '[:upper:]'`; do

    # Set some variables

    VO_lower=`echo $VO | tr '[:upper:]' '[:lower:]'`

    vo_user_prefix=`users_getvoprefix $VO`
    [ -z "$vo_user_prefix" ] && vo_user_prefix=$VO_lower

    vo_group=`users_getvogroup $VO`

    sgmuser=`users_getsgmuser $VO`
    prduser=`users_getprduser $VO`

    eval voms_pool='$'VO_${VO}_VOMS_POOL_PATH
    test -z "$voms_pool" || voms_pool=/${voms_pool#/}

    eval voms_servers='$'VO_${VO}_VOMS_SERVERS

    vo_match=/VO=$VO_lower/GROUP=/$VO_lower
    role_match=$vo_match/ROLE=

    echo "# $VO" >> $gmc

    ### VOMS sgm
    if [ "$sgmuser" -a "$voms_servers" ]; then

#
# "/VO=dteam/GROUP=/dteam/ROLE=lcgadmin":::sgm:
#

role=`sed -n 's|^"'"$role_match"'\(.*\)":.*:sgm:* *$|\1|p' $GROUPS_CONF`

echo "# Map VO members (Role) $sgmuser" >> $gmc

split_quoted_variable $voms_servers | while read server; do
    echo "group ${server%/}/Role=$role $sgmuser"  >> $gmc
    echo "group ${server%/}/Role=$role $VO_lower" >> $gmc_dm
done
echo >> $gmc
    fi

    ### VOMS prd
    if [ "$prduser" -a "$voms_servers" ]; then

#
# "/VO=dteam/GROUP=/dteam/ROLE=production":::prd:
#
```

```
role=`sed -n 's|^"'"$role_match"'\(.*\)":.*:prd:* *$|\1|p' $GROUPS_CONF`

echo "# Map VO members (Role) $prduser" >> $gmc

split_quoted_variable $voms_servers | while read server; do
    echo "group ${server%/}/Role=$role $prduser"  >> $gmc
    echo "group ${server%/}/Role=$role $VO_lower" >> $gmc_dm
done
echo >> $gmc
    fi

    ### VOMS pool
    if [ "$voms_servers" ]; then
echo "# Map VO members  (root Group) $VO_lower" >> $gmc

split_quoted_variable $voms_servers | while read server; do
    echo "group ${server%/}${voms_pool} .$vo_user_prefix" >> $gmc
    echo "group ${server%/}${voms_pool} $VO_lower"        >> $gmc_dm
done
echo >> $gmc
    fi

    echo "# LDAP lines for ${VO}" >> $gmc

    ### LDAP sgm
    if [ "$sgmuser" ]; then
eval ldap_sgm='$'VO_${VO}_SGM
test -z "$ldap_sgm" || {
    echo "group $ldap_sgm $sgmuser"  >> $gmc
    echo "group $ldap_sgm $VO_lower" >> $gmc_dm
}
    fi

    ### LDAP pool
    eval ldap_users='$'VO_${VO}_USERS
    test -z "$ldap_users" || {
echo "group $ldap_users .$vo_user_prefix" >> $gmc
echo "group $ldap_users $VO_lower"        >> $gmc_dm
    }

    echo >> $gmc
    echo >> $gmc

    ### VOMS gridmapfile and groupmapfile

    #
    # "/VO=cms/GROUP=/cms/ROLE=lcgadmin":::sgm:
    # "/VO=cms/GROUP=/cms/ROLE=production":::prd:
    # "/VO=cms/GROUP=/cms/GROUP=HeavyIons":cms01:1340::
    # "/VO=cms/GROUP=/cms/GROUP=Higgs":cms02:1341::
    # "/VO=cms/GROUP=/cms/GROUP=StandardModel":cms03:1342::
    # "/VO=cms/GROUP=/cms/GROUP=Susy":cms04:1343::
    # "/VO=cms/GROUP=/cms"::::
```

```
    #

    sed -n '/^"\/VO='"$VO_lower"'\//p' $GROUPS_CONF | while read line; do

fqan=` echo "$line" | sed 's/":.*/"/' `
line=` echo "$line" | sed 's/.*":://'  `
group=`echo "$line" | sed 's/:.*//'   `
line=` echo "$line" | sed 's/[^:]*://'`
gid=`  echo "$line" | sed 's/:.*//'   `
line=` echo "$line" | sed 's/[^:]*://'`
flag=` echo "$line" | sed 's/:.*//'   `

if [ "$flag" = sgm ]; then

    u=$sgmuser
    g=$vo_group

elif [ "$flag" = prd ]; then

    u=$prduser
    g=$vo_group

elif [ "$group" ]; then

    groupadd ${gid:+"-g"} ${gid:+"$gid"} "$group" 2>&1 | grep -v exists

    u=.$vo_user_prefix
    g=$group

else

    u=.$vo_user_prefix
    g=$vo_group
fi

echo "$fqan $u" >> $lcmaps_gridmapfile
echo "$fqan $g" >> $lcmaps_groupmapfile

    done

done # End of VO loop

cat << EOF >> $gmc
###############################################################################
# List of auth URIs
# eg 'auth ldap://marianne.in2p3.fr/ou=People,o=testbed,dc=eu-datagrid,dc=org'
# If these are defined then users must be authorised in one of the following
# auth servers.


# A list of authorised users.
EOF

GRIDMAP_AUTH=${GRIDMAP_AUTH:-\
```

```
ldap://lcg-registrar.cern.ch/ou=users,o=registrar,dc=lcg,dc=org}

for i in $GRIDMAP_AUTH; do
    echo "auth $i" >> $gmc
    echo "auth $i" >> $gmc_dm
    echo >> $gmc
done

f=$INSTALL_ROOT/edg/etc/grid-mapfile-local

[ -f $f ] || touch $f

cat << EOF >> $gmc
###############################################################################
# DEFAULT_LCLUSER: default_lcluser lcluser
# default_lcuser .


###############################################################################
# ALLOW and DENY: deny|allow pattern_to_match
# allow *INFN*


###############################################################################
# Local grid-mapfile to import and overide all the above information.
# eg, gmf_local $f

gmf_local $f
EOF

if [ ${gmc_dm:-/dev/null} != /dev/null ]; then
    f=${INSTALL_ROOT}/lcg/etc/lcgdm-mapfile-local

    [ -f $f ] || touch $f
fi

cat << EOF >> $gmc_dm
gmf_local $f
EOF

#
# bootstrap the grid-mapfile
#

cmd="$INSTALL_ROOT/edg/sbin/edg-mkgridmap \
--output=/etc/grid-security/grid-mapfile --safe"

echo "Now creating the grid-mapfile - this may take a few minutes..."

$cmd 2>> $YAIM_LOG

let minute="$RANDOM%60"

let h1="$RANDOM%6"
```

```
let h2="$h1+6"
let h3="$h2+6"
let h4="$h3+6"

cron_job edg-mkgridmap root "$minute $h1,$h2,$h3,$h4 * * * $cmd"

if ( echo "${NODE_TYPE_LIST}" | egrep -q 'dpm|LFC' ); then

    cmd="$INSTALL_ROOT/edg/libexec/edg-mkgridmap/edg-mkgridmap.pl \
--conf=$gmc_dm --output=$INSTALL_ROOT/lcg/etc/lcgdm-mapfile --safe"

    echo "Now creating the lcgdm-mapfile - this may take a few minutes..."

    $cmd 2>> $YAIM_LOG

    let minute="$RANDOM%60"

    let h1="$RANDOM%6"
    let h2="$h1+6"
    let h3="$h2+6"
    let h4="$h3+6"

    cron_job lcgdm-mkgridmap root "$minute $h1,$h2,$h3,$h4 * * * $cmd"

fi

if ( echo "${NODE_TYPE_LIST}" | grep -q '\<'RB ); then

    cron_job lcg-expiregridmapdir root  "5 * * * * \
${INSTALL_ROOT}/edg/sbin/lcg-expiregridmapdir.pl -e 240 -v >> \
/var/log/lcg-expiregridmapdir.log 2>&1"

elif ( echo "${NODE_TYPE_LIST}" | egrep -q 'dpm|LFC' ); then

    # No expiry
    rm -f ${CRON_DIR}/lcg-expiregridmapdir

else

    cron_job lcg-expiregridmapdir root  "5 * * * * \
${INSTALL_ROOT}/edg/sbin/lcg-expiregridmapdir.pl -v >> \
/var/log/lcg-expiregridmapdir.log 2>&1"

fi

return 0
}
```

27.11. CONFIG_JAVA

```
function config_java () {
```

```
INSTALL_ROOT=${INSTALL_ROOT:-/opt}

# If JAVA_LOCATION is not set by the admin, take a guess
if [ -z "$JAVA_LOCATION" ]; then
    java=`rpm -qa | grep j2sdk-` || java=`rpm -qa | grep j2re`
    if [ "$java" ]; then
JAVA_LOCATION=`rpm -ql $java | egrep '/bin/java$' | sort | head -1 | sed 's#/bin/java##'`
    fi
fi

if [ ! "$JAVA_LOCATION" -o ! -d "$JAVA_LOCATION" ]; then
   echo "Please check your value for JAVA_LOCATION"
   return 1
fi

if ( echo "${NODE_TYPE_LIST}" | grep TAR > /dev/null ); then

# We're configuring a relocatable distro

    if [ ! -d ${INSTALL_ROOT}/edg/etc/profile.d ]; then
mkdir -p ${INSTALL_ROOT}/edg/etc/profile.d/
    fi

    cat  > $INSTALL_ROOT/edg/etc/profile.d/j2.sh <<EOF

JAVA_HOME=$JAVA_LOCATION
export JAVA_HOME
EOF

    cat  > $INSTALL_ROOT/edg/etc/profile.d/j2.csh <<EOF

setenv JAVA_HOME $JAVA_LOCATION
EOF

    chmod a+rx $INSTALL_ROOT/edg/etc/profile.d/j2.sh
    chmod a+rx $INSTALL_ROOT/edg/etc/profile.d/j2.csh

    return 0

fi # end of relocatable stuff

# We're root and it's not a relocatable

if [ ! -d /etc/java ]; then
    mkdir /etc/java
fi

echo "export JAVA_HOME=$JAVA_LOCATION" > /etc/java/java.conf
echo "export JAVA_HOME=$JAVA_LOCATION" > /etc/java.conf
chmod +x /etc/java/java.conf

#This hack is here due to SL and the java profile rpms, Laurence Field

if [ ! -d ${INSTALL_ROOT}/edg/etc/profile.d ]; then
```

```
    mkdir -p ${INSTALL_ROOT}/edg/etc/profile.d/
fi

cat << EOF > $INSTALL_ROOT/edg/etc/profile.d/j2.sh
if [ -z "\$PATH" ]; then
   export PATH=${JAVA_LOCATION}/bin
else
   export PATH=${JAVA_LOCATION}/bin:\${PATH}
fi
EOF

chmod a+rx $INSTALL_ROOT/edg/etc/profile.d/j2.sh

cat << EOF > $INSTALL_ROOT/edg/etc/profile.d/j2.csh
if ( \$?PATH ) then
    setenv PATH ${JAVA_LOCATION}/bin:\${PATH}
else
    setenv PATH ${JAVA_LOCATION}/bin
endif
EOF

chmod a+rx $INSTALL_ROOT/edg/etc/profile.d/j2.csh

return 0

}
```

27.12. CONFIG_RGMA_CLIENT

```
config_rgma_client(){

requires MON_HOST REG_HOST

INSTALL_ROOT=${INSTALL_ROOT:-/opt}

# NB java stuff now in config_java, which must be run before

export RGMA_HOME=${INSTALL_ROOT}/glite

# in order to use python from userdeps.tgz we need to source the env
if ( echo "${NODE_TYPE_LIST}" | grep TAR > /dev/null ); then
    . $INSTALL_ROOT/etc/profile.d/grid_env.sh
fi

${RGMA_HOME}/share/rgma/scripts/rgma-setup.py --secure=yes --server=${MON_HOST} --registry=${REG_HOST} --schema=${R

cat << EOF > ${INSTALL_ROOT}/edg/etc/profile.d/edg-rgma-env.sh
export RGMA_HOME=${INSTALL_ROOT}/glite
export APEL_HOME=${INSTALL_ROOT}/glite

echo \$PYTHONPATH | grep -q ${INSTALL_ROOT}/glite/lib/python && isthere=1 || isthere=0
if [ \$isthere = 0 ]; then
```

```
    if [ -z \$PYTHONPATH ]; then
        export PYTHONPATH=${INSTALL_ROOT}/glite/lib/python
    else
        export PYTHONPATH=\$PYTHONPATH:${INSTALL_ROOT}/glite/lib/python
    fi
fi

echo \$LD_LIBRARY_PATH | grep -q ${INSTALL_ROOT}/glite/lib && isthere=1 || isthere=0
if [ \$isthere = 0 ]; then
    if [ -z \$LD_LIBRARY_PATH ]; then
        export LD_LIBRARY_PATH=${INSTALL_ROOT}/glite/lib
    else
        export LD_LIBRARY_PATH=\$LD_LIBRARY_PATH:${INSTALL_ROOT}/glite/lib
    fi
fi
EOF

chmod a+rx ${INSTALL_ROOT}/edg/etc/profile.d/edg-rgma-env.sh

cat << EOF > ${INSTALL_ROOT}/edg/etc/profile.d/edg-rgma-env.csh
setenv RGMA_HOME ${INSTALL_ROOT}/glite
setenv APEL_HOME ${INSTALL_ROOT}/glite

echo \$PYTHONPATH | grep -q ${INSTALL_ROOT}/glite/lib/python && set isthere=1 || set isthere=0
if ( \$isthere == 0 ) then
    if ( -z \$PYTHONPATH ) then
        setenv PYTHONPATH ${INSTALL_ROOT}/glite/lib/python
    else
        setenv PYTHONPATH \$PYTHONPATH\:${INSTALL_ROOT}/glite/lib/python
    endif
endif

echo \$LD_LIBRARY_PATH | grep -q ${INSTALL_ROOT}/glite/lib && set isthere=1 || set isthere=0
if ( \$isthere == 0 ) then
    if ( -z \$LD_LIBRARY_PATH ) then
        setenv LD_LIBRARY_PATH ${INSTALL_ROOT}/glite/lib
    else
        setenv LD_LIBRARY_PATH \$LD_LIBRARY_PATH\:${INSTALL_ROOT}/glite/lib
    endif
endif
EOF

chmod a+rx  ${INSTALL_ROOT}/edg/etc/profile.d/edg-rgma-env.csh

return 0
}
```

27.13. CONFIG_GIP

```
config_gip () {

INSTALL_ROOT=${INSTALL_ROOT:-/opt}
```

```
requires CE_HOST RB_HOST PX_HOST

#check_users_conf_format

#set some vars for storage elements
if ( echo "${NODE_TYPE_LIST}" | grep '\<SE' > /dev/null ); then
    requires VOS SITE_EMAIL SITE_NAME BDII_HOST VOS SITE_NAME
    if ( echo "${NODE_TYPE_LIST}" | grep SE_dpm > /dev/null ); then
requires DPM_HOST
se_host=$DPM_HOST
se_type="srm_v1"
control_protocol=srm_v1
control_endpoint=httpg://${se_host}
    elif ( echo "${NODE_TYPE_LIST}" | grep SE_dcache > /dev/null ); then
requires DCACHE_ADMIN
se_host=$DCACHE_ADMIN
se_type="srm_v1"
control_protocol=srm_v1
control_endpoint=httpg://${se_host}
    else
requires CLASSIC_STORAGE_DIR CLASSIC_HOST VO__STORAGE_DIR
se_host=$CLASSIC_HOST
se_type="disk"
control_protocol=classic
control_endpoint=classic
    fi
fi

if ( echo "${NODE_TYPE_LIST}" | grep '\<CE' > /dev/null ); then

    # GlueSite

    requires SITE_EMAIL SITE_NAME SITE_LOC SITE_LAT SITE_LONG SITE_WEB \
SITE_TIER SITE_SUPPORT_SITE SE_LIST

    outfile=$INSTALL_ROOT/lcg/var/gip/lcg-info-static-site.conf

    # set default SEs if they're currently undefined
    default_se=`set x $SE_LIST; echo "$2"`
    if [ "$default_se" ]; then
for VO in `echo $VOS | tr '[:lower:]' '[:upper:]'`; do
    if [ "x`eval echo '$'VO_${VO}_DEFAULT_SE`" = "x" ]; then
eval VO_${VO}_DEFAULT_SE=$default_se
    fi
done
    fi

    cat << EOF > $outfile
dn: GlueSiteUniqueID=$SITE_NAME
GlueSiteUniqueID: $SITE_NAME
GlueSiteName: $SITE_NAME
GlueSiteDescription: LCG Site
GlueSiteUserSupportContact: mailto: $SITE_EMAIL
```

```
GlueSiteSysAdminContact: mailto: $SITE_EMAIL
GlueSiteSecurityContact: mailto: $SITE_EMAIL
GlueSiteLocation: $SITE_LOC
GlueSiteLatitude: $SITE_LAT
GlueSiteLongitude: $SITE_LONG
GlueSiteWeb: $SITE_WEB
GlueSiteSponsor: none
GlueSiteOtherInfo: $SITE_TIER
GlueSiteOtherInfo: $SITE_SUPPORT_SITE
GlueForeignKey: GlueSiteUniqueID=${SITE_NAME}
EOF

    $INSTALL_ROOT/lcg/sbin/lcg-info-static-create -c $outfile -t \
    $INSTALL_ROOT/lcg/etc/GlueSite.template > \
    $INSTALL_ROOT/lcg/var/gip/ldif/static-file-Site.ldif

    # GlueCluster

    requires JOB_MANAGER CE_BATCH_SYS VOS QUEUES CE_BATCH_SYS CE_CPU_MODEL \
CE_CPU_VENDOR CE_CPU_SPEED CE_OS CE_OS_RELEASE CE_MINPHYSMEM \
CE_MINVIRTMEM CE_SMPSIZE CE_SI00 CE_SF00 CE_OUTBOUNDIP CE_INBOUNDIP \
CE_RUNTIMEENV

    outfile=$INSTALL_ROOT/lcg/var/gip/lcg-info-static-cluster.conf

    for VO in $VOS; do
        dir=${INSTALL_ROOT}/edg/var/info/$VO
        mkdir -p $dir
f=$dir/$VO.list
[ -f $f ] || touch $f
        # work out the sgm user for this VO
        sgmuser=`users_getsgmuser $VO`
sgmgroup=`id -g $sgmuser`
chown -R ${sgmuser}:${sgmgroup} $dir
chmod -R go-w $dir
    done

    cat <<EOF > $outfile

dn: GlueClusterUniqueID=${CE_HOST}
GlueClusterName: ${CE_HOST}
GlueForeignKey: GlueSiteUniqueID=${SITE_NAME}
GlueInformationServiceURL: ldap://`hostname -f`:2135/mds-vo-name=local,o=grid
EOF

    for QUEUE in $QUEUES; do
        echo "GlueClusterService: ${CE_HOST}:2119/jobmanager-$JOB_MANAGER-$QUEUE" >> $outfile
    done

    for QUEUE in $QUEUES; do
        echo "GlueForeignKey:" \
    "GlueCEUniqueID=${CE_HOST}:2119/jobmanager-$JOB_MANAGER-$QUEUE" >> $outfile
    done
```

```
    cat << EOF >> $outfile

dn: GlueSubClusterUniqueID=${CE_HOST}, GlueClusterUniqueID=${CE_HOST}
GlueChunkKey: GlueClusterUniqueID=${CE_HOST}
GlueHostArchitectureSMPSize: $CE_SMPSIZE
GlueHostBenchmarkSF00: $CE_SF00
GlueHostBenchmarkSI00: $CE_SI00
GlueHostMainMemoryRAMSize: $CE_MINPHYSMEM
GlueHostMainMemoryVirtualSize: $CE_MINVIRTMEM
GlueHostNetworkAdapterInboundIP: $CE_INBOUNDIP
GlueHostNetworkAdapterOutboundIP: $CE_OUTBOUNDIP
GlueHostOperatingSystemName: $CE_OS
GlueHostOperatingSystemRelease: $CE_OS_RELEASE
GlueHostOperatingSystemVersion: 3
GlueHostProcessorClockSpeed: $CE_CPU_SPEED
GlueHostProcessorModel: $CE_CPU_MODEL
GlueHostProcessorVendor: $CE_CPU_VENDOR
GlueSubClusterName: ${CE_HOST}
GlueSubClusterPhysicalCPUs: 0
GlueSubClusterLogicalCPUs: 0
GlueSubClusterTmpDir: /tmp
GlueSubClusterWNTmpDir: /tmp
GlueInformationServiceURL: ldap://`hostname -f`:2135/mds-vo-name=local,o=grid
EOF

    for x in $CE_RUNTIMEENV; do
        echo "GlueHostApplicationSoftwareRunTimeEnvironment: $x" >>  $outfile
    done

    $INSTALL_ROOT/lcg/sbin/lcg-info-static-create -c $outfile -t \
    $INSTALL_ROOT/lcg/etc/GlueCluster.template > \
    $INSTALL_ROOT/lcg/var/gip/ldif/static-file-Cluster.ldif

    # GlueCE

    outfile=$INSTALL_ROOT/lcg/var/gip/lcg-info-static-ce.conf

    cat /dev/null > $outfile

    for QUEUE in $QUEUES; do
        cat <<EOF >> $outfile


dn: GlueCEUniqueID=${CE_HOST}:2119/jobmanager-$JOB_MANAGER-$QUEUE
GlueCEHostingCluster: ${CE_HOST}
GlueCEName: $QUEUE
GlueCEInfoGatekeeperPort: 2119
GlueCEInfoHostName: ${CE_HOST}
GlueCEInfoLRMSType: $CE_BATCH_SYS
GlueCEInfoLRMSVersion: not defined
GlueCEInfoTotalCPUs: 0
GlueCEInfoJobManager: ${JOB_MANAGER}
GlueCEInfoContactString: ${CE_HOST}:2119/jobmanager-${JOB_MANAGER}-${QUEUE}
GlueCEInfoApplicationDir: ${VO_SW_DIR}
```

```
GlueCEInfoDataDir: ${CE_DATADIR:-unset}
GlueCEInfoDefaultSE: $default_se
GlueCEStateEstimatedResponseTime: 0
GlueCEStateFreeCPUs: 0
GlueCEStateRunningJobs: 0
GlueCEStateStatus: Production
GlueCEStateTotalJobs: 0
GlueCEStateWaitingJobs: 0
GlueCEStateWorstResponseTime: 0
GlueCEStateFreeJobSlots: 0
GlueCEPolicyMaxCPUTime: 0
GlueCEPolicyMaxRunningJobs: 0
GlueCEPolicyMaxTotalJobs: 0
GlueCEPolicyMaxWallClockTime: 0
GlueCEPolicyPriority: 1
GlueCEPolicyAssignedJobSlots: 0
GlueForeignKey: GlueClusterUniqueID=${CE_HOST}
GlueInformationServiceURL: ldap://`hostname -f`:2135/mds-vo-name=local,o=grid
EOF

        for VO in `echo $VOS | tr '[:lower:]' '[:upper:]'`; do
            for VO_QUEUE in `eval echo '$'VO_${VO}_QUEUES`; do
                if [ "${QUEUE}" = "${VO_QUEUE}" ]; then
                    echo "GlueCEAccessControlBaseRule:" \
"VO:`echo $VO | tr '[:upper:]' '[:lower:]'`" >> $outfile
                fi
            done
        done

for VO in `echo $VOS | tr '[:lower:]' '[:upper:]'`; do
            for VO_QUEUE in `eval echo '$'VO_${VO}_QUEUES`; do
                if [ "${QUEUE}" = "${VO_QUEUE}" ]; then
    cat << EOF >> $outfile

dn: GlueVOViewLocalID=`echo $VO | tr '[:upper:]' '[:lower:]'`,\
GlueCEUniqueID=${CE_HOST}:2119/jobmanager-${JOB_MANAGER}-${QUEUE}
GlueCEAccessControlBaseRule: VO:`echo $VO | tr '[:upper:]' '[:lower:]'`
GlueCEStateRunningJobs: 0
GlueCEStateWaitingJobs: 0
GlueCEStateTotalJobs: 0
GlueCEStateFreeJobSlots: 0
GlueCEStateEstimatedResponseTime: 0
GlueCEStateWorstResponseTime: 0
GlueCEInfoDefaultSE:  `eval echo '$'VO_${VO}_DEFAULT_SE`
GlueCEInfoApplicationDir:  `eval echo '$'VO_${VO}_SW_DIR`
GlueCEInfoDataDir: ${CE_DATADIR:-unset}
GlueChunkKey: GlueCEUniqueID=${CE_HOST}:2119/jobmanager-${JOB_MANAGER}-${QUEUE}
EOF
                fi
    done
done
    done

    $INSTALL_ROOT/lcg/sbin/lcg-info-static-create -c $outfile -t \
```

```
    $INSTALL_ROOT/lcg/etc/GlueCE.template > \
    $INSTALL_ROOT/lcg/var/gip/ldif/static-file-CE.ldif


    # GlueCESEBind

    outfile=$INSTALL_ROOT/lcg/var/gip/lcg-info-static-cesebind.conf
    echo "" > $outfile

    for QUEUE in $QUEUES
      do
      echo "dn: GlueCESEBindGroupCEUniqueID=${CE_HOST}:2119/jobmanager-$JOB_MANAGER-$QUEUE" \
>> $outfile
      for se in $SE_LIST
        do
        echo "GlueCESEBindGroupSEUniqueID: $se" >> $outfile
        done
    done

    for se in $SE_LIST; do

case "$se" in
"$DPM_HOST") accesspoint=$DPMDATA;;
"$DCACHE_ADMIN") accesspoint="/pnfs/`hostname -d`/data";;
*) accesspoint=$CLASSIC_STORAGE_DIR ;;
esac

        for QUEUE in $QUEUES; do

            cat <<EOF >> $outfile

dn: GlueCESEBindSEUniqueID=$se,\
GlueCESEBindGroupCEUniqueID=${CE_HOST}:2119/jobmanager-$JOB_MANAGER-$QUEUE
GlueCESEBindCEAccesspoint: $accesspoint
GlueCESEBindCEUniqueID: ${CE_HOST}:2119/jobmanager-$JOB_MANAGER-$QUEUE
GlueCESEBindMountInfo: $accesspoint
GlueCESEBindWeight: 0


EOF
        done
    done

    $INSTALL_ROOT/lcg/sbin/lcg-info-static-create -c $outfile -t \
    $INSTALL_ROOT/lcg/etc/GlueCESEBind.template > \
    $INSTALL_ROOT/lcg/var/gip/ldif/static-file-CESEBind.ldif

    # Set some vars based on the LRMS

    case "$CE_BATCH_SYS" in
    condor|CONDOR) plugin="${INSTALL_ROOT}/lcg/libexec/lcg-info-dynamic-condor /opt/condor/bin/ $INSTALL_ROOT/lcg/e
    lsf|LSF)       plugin="${INSTALL_ROOT}/lcg/libexec/lcg-info-dynamic-lsf /usr/local/lsf/bin/ $INSTALL_ROOT/lcg/e
    pbs|PBS)       plugin="${INSTALL_ROOT}/lcg/libexec/lcg-info-dynamic-pbs /opt/lcg/var/gip/ldif/static-file-CE.ld
   vo_max_jobs_cmd="";;
```

```
    *)              plugin="${INSTALL_ROOT}/lcg/libexec/lcg-info-dynamic-pbs /opt/lcg/var/gip/ldif/static-file-CE.ld
   vo_max_jobs_cmd="$INSTALL_ROOT/lcg/libexec/vomaxjobs-maui";;
    esac

    # Configure the dynamic plugin appropriate for the batch sys

    cat << EOF  > ${INSTALL_ROOT}/lcg/var/gip/plugin/lcg-info-dynamic-ce
#!/bin/sh
$plugin
EOF

    chmod +x ${INSTALL_ROOT}/lcg/var/gip/plugin/lcg-info-dynamic-ce

    # Configure the ERT plugin

    cat << EOF > ${INSTALL_ROOT}/lcg/var/gip/plugin/lcg-info-dynamic-scheduler-wrapper
#!/bin/sh
${INSTALL_ROOT}/lcg/libexec/lcg-info-dynamic-scheduler -c ${INSTALL_ROOT}/lcg/etc/lcg-info-dynamic-scheduler.conf
EOF

    chmod +x ${INSTALL_ROOT}/lcg/var/gip/plugin/lcg-info-dynamic-scheduler-wrapper

    if ( echo $CE_BATCH_SYS | egrep -qi 'pbs|torque' ); then

cat <<EOF > $INSTALL_ROOT/lcg/etc/lcg-info-dynamic-scheduler.conf
[Main]
static_ldif_file: $INSTALL_ROOT/lcg/var/gip/ldif/static-file-CE.ldif
vomap :
EOF

for vo in $VOS; do
    vo_group=`users_getvogroup $vo`
    if [ $vo_group ]; then
echo "   $vo_group:$vo" >> $INSTALL_ROOT/lcg/etc/lcg-info-dynamic-scheduler.conf
    fi
done

cat <<EOF >> $INSTALL_ROOT/lcg/etc/lcg-info-dynamic-scheduler.conf
module_search_path : ../lrms:../ett
[LRMS]
lrms_backend_cmd: $INSTALL_ROOT/lcg/libexec/lrmsinfo-pbs
[Scheduler]
vo_max_jobs_cmd: $vo_max_jobs_cmd
cycle_time : 0
EOF
    fi

    # Configure the provider for installed software

    if [ -f $INSTALL_ROOT/lcg/libexec/lcg-info-provider-software ]; then
cat <<EOF > $INSTALL_ROOT/lcg/var/gip/provider/lcg-info-provider-software-wrapper
#!/bin/sh
$INSTALL_ROOT/lcg/libexec/lcg-info-provider-software -p $INSTALL_ROOT/edg/var/info  -c $CE_HOST
EOF
```

```
chmod +x $INSTALL_ROOT/lcg/var/gip/provider/lcg-info-provider-software-wrapper
    fi

fi #endif for CE_HOST

if [ "$GRIDICE_SERVER_HOST" = "`hostname -f`" ]; then

    requires VOS SITE_NAME SITE_EMAIL

outfile=$INSTALL_ROOT/lcg/var/gip/lcg-info-static-gridice.conf

    cat <<EOF > $outfile

dn: GlueServiceUniqueID=${GRIDICE_SERVER_HOST}:2136
GlueServiceName: ${SITE_NAME}-gridice
GlueServiceType: gridice
GlueServiceVersion: 1.1.0
GlueServiceEndpoint: ldap://${GRIDICE_SERVER_HOST}:2136/mds-vo-name=local,o=grid
GlueServiceURI: unset
GlueServiceAccessPointURL: not_used
GlueServiceStatus: OK
GlueServiceStatusInfo: No Problems
GlueServiceWSDL: unset
GlueServiceSemantics: unset
GlueServiceStartTime: 1970-01-01T00:00:00Z
GlueForeignKey: GlueSiteUniqueID=${SITE_NAME}
EOF

    for VO in $VOS; do
        echo "GlueServiceAccessControlRule: $VO" >> $outfile
echo "GlueServiceOwner: $VO" >> $outfile
    done


FMON='--fmon=yes'

    $INSTALL_ROOT/lcg/sbin/lcg-info-static-create -c $outfile -t \
$INSTALL_ROOT/lcg/etc/GlueService.template > \
$INSTALL_ROOT/lcg/var/gip/ldif/static-file-GRIDICE.ldif

fi #endif for GRIDICE_SERVER_HOST

if ( echo "${NODE_TYPE_LIST}" | grep -w PX > /dev/null ); then

    requires GRID_TRUSTED_BROKERS SITE_EMAIL SITE_NAME

outfile=$INSTALL_ROOT/lcg/var/gip/lcg-info-static-px.conf

    cat << EOF > $outfile

dn: GlueServiceUniqueID=${PX_HOST}:7512
GlueServiceName: ${SITE_NAME}-myproxy
GlueServiceType: myproxy
GlueServiceVersion: 1.1.0
```

```
GlueServiceEndpoint: ${PX_HOST}:7512
GlueServiceURI: unset
GlueServiceAccessPointURL: myproxy://${PX_HOST}
GlueServiceStatus: OK
GlueServiceStatusInfo: No Problems
GlueServiceWSDL: unset
GlueServiceSemantics: unset
GlueServiceStartTime: 1970-01-01T00:00:00Z
GlueServiceOwner: LCG
GlueForeignKey: GlueSiteUniqueID=${SITE_NAME}
EOF

    split_quoted_variable $GRID_TRUSTED_BROKERS | while read x; do
        echo "GlueServiceAccessControlRule: $x" >> $outfile
    done

    $INSTALL_ROOT/lcg/sbin/lcg-info-static-create -c $outfile -t \
$INSTALL_ROOT/lcg/etc/GlueService.template > \
$INSTALL_ROOT/lcg/var/gip/ldif/static-file-PX.ldif

fi #endif for PX_HOST

if ( echo "${NODE_TYPE_LIST}" | grep -w RB > /dev/null ); then

    requires VOS SITE_EMAIL SITE_NAME

outfile=$INSTALL_ROOT/lcg/var/gip/lcg-info-static-rb.conf


    cat <<EOF > $outfile

dn: GlueServiceUniqueID=${RB_HOST}:7772
GlueServiceName: ${SITE_NAME}-rb
GlueServiceType: ResourceBroker
GlueServiceVersion: 1.2.0
GlueServiceEndpoint: ${RB_HOST}:7772
GlueServiceURI: unset
GlueServiceAccessPointURL: not_used
GlueServiceStatus: OK
GlueServiceStatusInfo: No Problems
GlueServiceWSDL: unset
GlueServiceSemantics: unset
GlueServiceStartTime: 1970-01-01T00:00:00Z
GlueForeignKey: GlueSiteUniqueID=${SITE_NAME}
EOF

    for VO in $VOS; do
        echo "GlueServiceAccessControlRule: $VO" >> $outfile
echo "GlueServiceOwner: $VO" >> $outfile
    done

    cat <<EOF >> $outfile
dn: GlueServiceDataKey=HeldJobs,GlueServiceUniqueID=gram://${RB_HOST}:7772
GlueServiceDataKey: HeldJobs
```

```
GlueServiceDataValue: 0
GlueChunkKey: GlueServiceUniqueID=gram://${RB_HOST}:7772

dn: GlueServiceDataKey=IdleJobs,GlueServiceUniqueID=gram://${RB_HOST}:7772
GlueServiceDataKey: IdleJobs
GlueServiceDataValue: 0
GlueChunkKey: GlueServiceUniqueID=gram://${RB_HOST}:7772

dn: GlueServiceDataKey=JobController,GlueServiceUniqueID=gram://${RB_HOST}:7772
GlueServiceDataKey: JobController
GlueServiceDataValue: 0
GlueChunkKey: GlueServiceUniqueID=gram://${RB_HOST}:7772

dn: GlueServiceDataKey=Jobs,GlueServiceUniqueID=gram://${RB_HOST}:7772
GlueServiceDataKey: Jobs
GlueServiceDataValue: 0
GlueChunkKey: GlueServiceUniqueID=gram://${RB_HOST}:7772

dn: GlueServiceDataKey=LogMonitor,GlueServiceUniqueID=gram://${RB_HOST}:7772
GlueServiceDataKey: LogMonitor
GlueServiceDataValue: 0
GlueChunkKey: GlueServiceUniqueID=gram://${RB_HOST}:7772

dn: GlueServiceDataKey=RunningJobs,GlueServiceUniqueID=gram://${RB_HOST}:7772
GlueServiceDataKey: RunningJobs
GlueServiceDataValue: 14
GlueChunkKey: GlueServiceUniqueID=gram://${RB_HOST}:7772

dn: GlueServiceDataKey=WorkloadManager,GlueServiceUniqueID=gram://${RB_HOST}:7772
GlueServiceDataKey: WorkloadManager
GlueServiceDataValue: 0
GlueChunkKey: GlueServiceUniqueID=gram://${RB_HOST}:7772

EOF

    $INSTALL_ROOT/lcg/sbin/lcg-info-static-create -c $outfile -t \
$INSTALL_ROOT/lcg/etc/GlueService.template > \
$INSTALL_ROOT/lcg/var/gip/ldif/static-file-RB.ldif

fi #endif for RB_HOST

if ( echo "${NODE_TYPE_LIST}" | grep '\<LFC' > /dev/null ); then

outfile=$INSTALL_ROOT/lcg/var/gip/lcg-info-static-lfc.conf
cat /dev/null > $outfile

    requires VOS SITE_EMAIL SITE_NAME BDII_HOST LFC_HOST

    if [ "$LFC_LOCAL" ]; then
lfc_local=$LFC_LOCAL
    else
# populate lfc_local with the VOS which are not set to be central
unset lfc_local
for i in $VOS; do
```

```
    if ( ! echo $LFC_CENTRAL | grep -qw $i ); then
lfc_local="$lfc_local $i"
    fi
done
    fi

    if [ "$LFC_CENTRAL" ]; then

cat <<EOF >> $outfile
dn: GlueServiceUniqueID=http://${LFC_HOST}:8085/
GlueServiceName: ${SITE_NAME}-lfc-dli
GlueServiceType: data-location-interface
GlueServiceVersion: 1.0.0
GlueServiceEndpoint: http://${LFC_HOST}:8085/
GlueServiceURI: http://${LFC_HOST}:8085/
GlueServiceAccessPointURL: http://${LFC_HOST}:8085/
GlueServiceStatus: OK
GlueServiceStatusInfo: No Problems
GlueServiceWSDL: unset
GlueServiceSemantics: unset
GlueServiceStartTime: 1970-01-01T00:00:00Z
GlueForeignKey: GlueSiteUniqueID=${SITE_NAME}
EOF

for VO in $LFC_CENTRAL; do
    echo "GlueServiceOwner: $VO" >> $outfile
    echo "GlueServiceAccessControlRule: $VO" >> $outfile
done

echo >> $outfile

cat <<EOF >> $outfile
dn: GlueServiceUniqueID=${LFC_HOST}
GlueServiceName: ${SITE_NAME}-lfc
GlueServiceType: lcg-file-catalog
GlueServiceVersion: 1.0.0
GlueServiceEndpoint: ${LFC_HOST}
GlueServiceURI: ${LFC_HOST}
GlueServiceAccessPointURL: ${LFC_HOST}
GlueServiceStatus: OK
GlueServiceStatusInfo: No Problems
GlueServiceWSDL: unset
GlueServiceSemantics: unset
GlueServiceStartTime: 1970-01-01T00:00:00Z
GlueForeignKey: GlueSiteUniqueID=${SITE_NAME}
EOF

for VO in $LFC_CENTRAL; do
    echo "GlueServiceOwner: $VO" >> $outfile
    echo "GlueServiceAccessControlRule: $VO" >> $outfile
done

        echo >> $outfile
```

```
    fi

    if [ "$lfc_local" ]; then

        cat <<EOF >> $outfile
dn: GlueServiceUniqueID=http://${LFC_HOST}:8085/,o=local
GlueServiceName: ${SITE_NAME}-lfc-dli
GlueServiceType: local-data-location-interface
GlueServiceVersion: 1.0.0
GlueServiceEndpoint: http://${LFC_HOST}:8085/
GlueServiceURI: http://${LFC_HOST}:8085/
GlueServiceAccessPointURL: http://${LFC_HOST}:8085/
GlueServiceStatus: OK
GlueServiceStatusInfo: No Problems
GlueServiceWSDL: unset
GlueServiceSemantics: unset
GlueServiceStartTime: 1970-01-01T00:00:00Z
GlueForeignKey: GlueSiteUniqueID=${SITE_NAME}
EOF

        for VO in $lfc_local; do
            echo "GlueServiceOwner: $VO" >> $outfile
            echo "GlueServiceAccessControlRule: $VO" >> $outfile
        done

        echo >> $outfile

cat <<EOF >> $outfile
dn: GlueServiceUniqueID=${LFC_HOST},o=local
GlueServiceName: ${SITE_NAME}-lfc
GlueServiceType: lcg-local-file-catalog
GlueServiceVersion: 1.0.0
GlueServiceEndpoint: ${LFC_HOST}
GlueServiceURI: ${LFC_HOST}
GlueServiceAccessPointURL: ${LFC_HOST}
GlueServiceStatus: OK
GlueServiceStatusInfo: No Problems
GlueServiceWSDL: unset
GlueServiceSemantics: unset
GlueServiceStartTime: 1970-01-01T00:00:00Z
GlueForeignKey: GlueSiteUniqueID=${SITE_NAME}
EOF

for VO in $lfc_local; do
    echo "GlueServiceOwner: $VO" >> $outfile
    echo "GlueServiceAccessControlRule: $VO" >> $outfile
done

    fi

    $INSTALL_ROOT/lcg/sbin/lcg-info-static-create -c $outfile -t \
$INSTALL_ROOT/lcg/etc/GlueService.template > \
$INSTALL_ROOT/lcg/var/gip/ldif/static-file-LFC.ldif
```

```
fi # end of LFC

if ( echo "${NODE_TYPE_LIST}" | egrep -q 'dcache|dpm_(mysql|oracle)' ); then

    outfile=$INSTALL_ROOT/lcg/var/gip/lcg-info-static-dse.conf

    cat <<EOF > $outfile

dn: GlueServiceUniqueID=httpg://${se_host}:8443/srm/managerv1
GlueServiceName: ${SITE_NAME}-srm
GlueServiceType: srm_v1
GlueServiceVersion: 1.0.0
GlueServiceEndpoint: httpg://${se_host}:8443/srm/managerv1
GlueServiceURI: httpg://${se_host}:8443/srm/managerv1
GlueServiceAccessPointURL: httpg://${se_host}:8443/srm/managerv1
GlueServiceStatus: OK
GlueServiceStatusInfo: No Problems
GlueServiceWSDL: unset
GlueServiceSemantics: unset
GlueServiceStartTime: 1970-01-01T00:00:00Z
GlueServiceOwner: LCG
GlueForeignKey: GlueSiteUniqueID=${SITE_NAME}
EOF

    for VO in $VOS; do
echo "GlueServiceAccessControlRule: $VO" >> $outfile
    done

    cat <<EOF >> $outfile
GlueServiceInformationServiceURL: \
MDS2GRIS:ldap://${BDII_HOST}:2170/mds-vo-name=${SITE_NAME},o=grid
GlueServiceStatus: OK
EOF

    $INSTALL_ROOT/lcg/sbin/lcg-info-static-create -c $outfile -t \
$INSTALL_ROOT/lcg/etc/GlueService.template > \
$INSTALL_ROOT/lcg/var/gip/ldif/static-file-dSE.ldif

fi # end of dcache,dpm

if ( echo "${NODE_TYPE_LIST}" | egrep -q 'SE_dpm_(mysql|oracle)' ); then

    # Install dynamic script pointing to gip plugin
    cat << EOF  > ${INSTALL_ROOT}/lcg/var/gip/plugin/lcg-info-dynamic-se
#! /bin/sh
${INSTALL_ROOT}/lcg/libexec/lcg-info-dynamic-dpm ${INSTALL_ROOT}/lcg/var/gip/ldif/static-file-SE.ldif
EOF

    chmod +x ${INSTALL_ROOT}/lcg/var/gip/plugin/lcg-info-dynamic-se

fi # end of dpm


if ( echo "${NODE_TYPE_LIST}" | grep '\<SE' > /dev/null ); then
```

```
outfile=$INSTALL_ROOT/lcg/var/gip/lcg-info-static-se.conf


    # dynamic_script points to the script generated by config_info_dynamic_se<se_type>
#    echo ""> $outfile
#    echo "dynamic_script=${INSTALL_ROOT}/lcg/libexec5A/lcg-info-dynamic-se" >> $outfile
#    echo >> $outfile        # Empty line to separate it form published info

    cat <<EOF > $outfile
dn: GlueSEUniqueID=${se_host}
GlueSEName: $SITE_NAME:${se_type}
GlueSEPort: 2811
GlueSESizeTotal: 0
GlueSESizeFree: 0
GlueSEArchitecture: multidisk
GlueInformationServiceURL: ldap://`hostname -f`:2135/mds-vo-name=local,o=grid
GlueForeignKey: GlueSiteUniqueID=${SITE_NAME}

dn: GlueSEAccessProtocolLocalID=gsiftp, GlueSEUniqueID=${se_host}
GlueSEAccessProtocolType: gsiftp
GlueSEAccessProtocolEndpoint: gsiftp://${se_host}
GlueSEAccessProtocolCapability: file transfer
GlueSEAccessProtocolVersion: 1.0.0
GlueSEAccessProtocolPort: 2811
GlueSEAccessProtocolSupportedSecurity: GSI
GlueChunkKey: GlueSEUniqueID=${se_host}

dn: GlueSEAccessProtocolLocalID=rfio, GlueSEUniqueID=${se_host}
GlueSEAccessProtocolType: rfio
GlueSEAccessProtocolEndpoint:
GlueSEAccessProtocolCapability:
GlueSEAccessProtocolVersion: 1.0.0
GlueSEAccessProtocolPort: 5001
GlueSEAccessProtocolSupportedSecurity: RFIO
GlueChunkKey: GlueSEUniqueID=${se_host}

dn: GlueSEControlProtocolLocalID=$control_protocol, GlueSEUniqueID=${se_host}
GlueSEControlProtocolType: $control_protocol
GlueSEControlProtocolEndpoint: $control_endpoint
GlueSEControlProtocolCapability:
GlueSEControlProtocolVersion: 1.0.0
GlueChunkKey: GlueSEUniqueID=${se_host}
EOF

for VO in $VOS; do

    if ( echo "${NODE_TYPE_LIST}" | grep SE_dpm > /dev/null ); then
storage_path="/dpm/`hostname -d`/home/${VO}"
storage_root="${VO}:${storage_path}"
    elif ( echo "${NODE_TYPE_LIST}" | grep SE_dcache > /dev/null ); then
storage_path="/pnfs/`hostname -d`/data/${VO}"
storage_root="${VO}:${storage_path}"
    else
```

```
storage_path=$( eval echo '$'VO_`echo ${VO} | tr '[:lower:]' '[:upper:]'`_STORAGE_DIR )
storage_root="${VO}:${storage_path#${CLASSIC_STORAGE_DIR}}"
    fi

    cat <<EOF >> $outfile

dn: GlueSALocalID=$VO,GlueSEUniqueID=${se_host}
GlueSARoot: $storage_root
GlueSAPath: $storage_path
GlueSAType: permanent
GlueSAPolicyMaxFileSize: 10000
GlueSAPolicyMinFileSize: 1
GlueSAPolicyMaxData: 100
GlueSAPolicyMaxNumFiles: 10
GlueSAPolicyMaxPinDuration: 10
GlueSAPolicyQuota: 0
GlueSAPolicyFileLifeTime: permanent
GlueSAStateAvailableSpace: 1
GlueSAStateUsedSpace: 1
GlueSAAccessControlBaseRule: $VO
GlueChunkKey: GlueSEUniqueID=${se_host}
EOF

done

    $INSTALL_ROOT/lcg/sbin/lcg-info-static-create -c $outfile -t \
$INSTALL_ROOT/lcg/etc/GlueSE.template > \
$INSTALL_ROOT/lcg/var/gip/ldif/static-file-SE.ldif

fi #endif for SE_HOST

if ( echo "${NODE_TYPE_LIST}" | grep -w VOBOX > /dev/null ); then

outfile=$INSTALL_ROOT/lcg/var/gip/lcg-info-static-vobox.conf


    for x in VOS SITE_EMAIL SITE_NAME VOBOX_PORT; do
        if [ "x`eval echo '$'$x`" = "x" ]; then
            echo "\$$x not set"
            return 1
        fi
    done

    for VO in $VOS; do
        dir=${INSTALL_ROOT}/edg/var/info/$VO
        mkdir -p $dir
f=$dir/$VO.list
[ -f $f ] || touch $f
        # work out the sgm user for this VO
        sgmuser=`users_getsgmuser $VO`
sgmgroup=`id -g $sgmuser`
chown -R ${sgmuser}:${sgmgroup} $dir
chmod -R go-w $dir
    done
```

```
    cat <<EOF > $outfile
dn: GlueServiceUniqueID=gsissh://${VOBOX_HOST}:${VOBOX_PORT}
GlueServiceName: ${SITE_NAME}-vobox
GlueServiceType: VOBOX
GlueServiceVersion: 1.0.0
GlueServiceEndpoint: gsissh://${VOBOX_HOST}:${VOBOX_PORT}
GlueServiceURI: unset
GlueServiceAccessPointURL: gsissh://${VOBOX_HOST}:${VOBOX_PORT}
GlueServiceStatus: OK
GlueServiceStatusInfo: No Problems
GlueServiceWSDL: unset
GlueServiceSemantics: unset
GlueServiceStartTime: 1970-01-01T00:00:00Z
GlueServiceOwner: LCG
GlueForeignKey: GlueSiteUniqueID=${SITE_NAME}
EOF

    for VO in $VOS; do
        echo "GlueServiceAccessControlRule: $VO" >> $outfile
    done

    echo >> $outfile

    $INSTALL_ROOT/lcg/sbin/lcg-info-static-create -c $outfile -t \
$INSTALL_ROOT/lcg/etc/GlueService.template > \
$INSTALL_ROOT/lcg/var/gip/ldif/static-file-VOBOX.ldif

fi #endif for VOBOX_HOST


cat << EOT > $INSTALL_ROOT/globus/libexec/edg.info
#!/bin/bash
#
# info-globus-ldif.sh
#
#Configures information providers for MDS
#
cat << EOF

dn: Mds-Vo-name=local,o=grid
objectclass: GlobusTop
objectclass: GlobusActiveObject
objectclass: GlobusActiveSearch
type: exec
path: $INSTALL_ROOT/lcg/libexec/
base: lcg-info-wrapper
args:
cachetime: 60
timelimit: 20
sizelimit: 250

EOF
```

```
EOT

chmod a+x $INSTALL_ROOT/globus/libexec/edg.info

if [ ! -d "$INSTALL_ROOT/lcg/libexec" ]; then
    mkdir -p $INSTALL_ROOT/lcg/libexec
fi

cat << EOF > $INSTALL_ROOT/lcg/libexec/lcg-info-wrapper
#!/bin/sh

export LANG=C
$INSTALL_ROOT/lcg/bin/lcg-info-generic $INSTALL_ROOT/lcg/etc/lcg-info-generic.conf

EOF

chmod a+x $INSTALL_ROOT/lcg/libexec/lcg-info-wrapper

cat << EOT > $INSTALL_ROOT/globus/libexec/edg.schemalist
#!/bin/bash

cat <<EOF
${INSTALL_ROOT}/globus/etc/openldap/schema/core.schema
${INSTALL_ROOT}/glue/schema/ldap/Glue-CORE.schema
${INSTALL_ROOT}/glue/schema/ldap/Glue-CE.schema
${INSTALL_ROOT}/glue/schema/ldap/Glue-CESEBind.schema
${INSTALL_ROOT}/glue/schema/ldap/Glue-SE.schema
EOF

EOT

chmod a+x $INSTALL_ROOT/globus/libexec/edg.schemalist

# Configure gin
if ( ! echo "${NODE_TYPE_LIST}" | egrep -q '^UI$|^WN[A-Za-z_]*$'  ); then
    if [ ! -d ${INSTALL_ROOT}/glite/var/rgma/.certs ]; then
mkdir -p ${INSTALL_ROOT}/glite/var/rgma/.certs
    fi

    cp -pf /etc/grid-security/hostcert.pem /etc/grid-security/hostkey.pem \
${INSTALL_ROOT}/glite/var/rgma/.certs
    chown rgma:rgma ${INSTALL_ROOT}/glite/var/rgma/.certs/host*

    (
egrep -v 'sslCertFile|sslKey' \
    ${INSTALL_ROOT}/glite/etc/rgma/ClientAuthentication.props
echo "sslCertFile=${INSTALL_ROOT}/glite/var/rgma/.certs/hostcert.pem"
echo "sslKey=${INSTALL_ROOT}/glite/var/rgma/.certs/hostkey.pem"
    ) > /tmp/props.$$
    mv -f /tmp/props.$$ ${INSTALL_ROOT}/glite/etc/rgma/ClientAuthentication.props

    #Turn on Gin for the GIP and maybe FMON
    export RGMA_HOME=${INSTALL_ROOT}/glite
    ${RGMA_HOME}/bin/rgma-gin-config --gip=yes ${FMON}
```

```
    /sbin/chkconfig rgma-gin on
    /etc/rc.d/init.d/rgma-gin restart 2>${YAIM_LOG}
fi


return 0
}
```

27.14. CONFIG_GLOBUS

```
config_globus(){
# $Id: config_globus,v 1.34 2006/01/06 13:45:51 maart Exp $


requires CE_HOST PX_HOST RB_HOST SITE_NAME

GLOBUS_MDS=no
GLOBUS_GRIDFTP=no
GLOBUS_GATEKEEPER=no

if ( echo "${NODE_TYPE_LIST}" | grep '\<'CE > /dev/null ); then
    GLOBUS_MDS=yes
    GLOBUS_GRIDFTP=yes
    GLOBUS_GATEKEEPER=yes
fi
if ( echo "${NODE_TYPE_LIST}" | grep VOBOX > /dev/null ); then
    GLOBUS_MDS=yes
    if ! ( echo "${NODE_TYPE_LIST}" | grep '\<'RB > /dev/null ); then
GLOBUS_GRIDFTP=yes
    fi
fi
if ( echo "${NODE_TYPE_LIST}" | grep '\<'SE > /dev/null ); then
    GLOBUS_MDS=yes
    GLOBUS_GRIDFTP=yes
fi
# DPM has its own ftp server
if ( echo "${NODE_TYPE_LIST}" | grep SE_dpm > /dev/null ); then
    GLOBUS_GRIDFTP=no
fi
if ( echo "${NODE_TYPE_LIST}" | grep '\<'PX > /dev/null ); then
    GLOBUS_MDS=yes
fi
if ( echo "${NODE_TYPE_LIST}" | grep '\<'RB > /dev/null ); then
    GLOBUS_MDS=yes
fi
if ( echo "${NODE_TYPE_LIST}" | grep '\<'LFC > /dev/null ); then
    GLOBUS_MDS=yes
fi
if ( echo "${NODE_TYPE_LIST}" | grep SE_dpm > /dev/null ); then
    X509_DPM1="x509_user_cert=/home/edginfo/.globus/usercert.pem"
    X509_DPM2="x509_user_key=/home/edginfo/.globus/userkey.pem"
else
```

```
    X509_DPM1=""
    X509_DPM2=""
fi
if [ "$GRIDICE_SERVER_HOST" = "`hostname -f`" ]; then
    GLOBUS_MDS=yes
fi

INSTALL_ROOT=${INSTALL_ROOT:-/opt}

cat <<EOF > /etc/globus.conf
########################################################################
#
# Globus configuraton.
#
########################################################################
[common]
GLOBUS_LOCATION=${INSTALL_ROOT}/globus
globus_flavor_name=gcc32dbg
x509_user_cert=/etc/grid-security/hostcert.pem
x509_user_key=/etc/grid-security/hostkey.pem
gridmap=/etc/grid-security/grid-mapfile
gridmapdir=/etc/grid-security/gridmapdir/

EOF

if [ "$GLOBUS_MDS" = "yes" ]; then
cat <<EOF >> /etc/globus.conf

[mds]
globus_flavor_name=gcc32dbgpthr
user=edginfo
$X509_DPM1
$X509_DPM2

[mds/gris/provider/edg]

EOF

cat <<EOF >> /etc/globus.conf
[mds/gris/registration/site]
regname=$SITE_NAME
reghn=$CE_HOST

EOF
else
echo "[mds]"  >> /etc/globus.conf

fi

if [ "$GLOBUS_GRIDFTP" = "yes" ]; then

    cat <<EOF >> /etc/globus.conf
[gridftp]
log=/var/log/globus-gridftp.log
```

```
EOF

    cat <<EOF > /etc/logrotate.d/gridftp
/var/log/globus-gridftp.log /var/log/gridftp-lcas_lcmaps.log {
missingok
daily
compress
rotate 31
create 0644 root root
sharedscripts
}
EOF

else
    echo "[gridftp]"  >> /etc/globus.conf
fi


if [ "$GLOBUS_GATEKEEPER" = "yes" ]; then

if [ "x`grep globus-gatekeeper /etc/services`" = "x" ]; then
    echo "globus-gatekeeper   2119/tcp" >> /etc/services
fi

cat <<EOF > /etc/logrotate.d/globus-gatekeeper
/var/log/globus-gatekeeper.log {
nocompress
copy
rotate 1
prerotate
killall -s USR1 -e /opt/edg/sbin/edg-gatekeeper
endscript
postrotate
find /var/log/globus-gatekeeper.log.20???????????.*[0-9] -mtime +7 -exec gzip {} \;
endscript
}
EOF

cat <<EOF >> /etc/globus.conf
[gatekeeper]

default_jobmanager=fork
job_manager_path=\$GLOBUS_LOCATION/libexec
globus_gatekeeper=${INSTALL_ROOT}/edg/sbin/edg-gatekeeper
extra_options=\"-lcas_db_file lcas.db -lcas_etc_dir ${INSTALL_ROOT}/edg/etc/lcas/ -lcasmod_dir \
${INSTALL_ROOT}/edg/lib/lcas/ -lcmaps_db_file lcmaps.db -lcmaps_etc_dir ${INSTALL_ROOT}/edg/etc/lcmaps -lcmapsmod_d
logfile=/var/log/globus-gatekeeper.log
jobmanagers="fork ${JOB_MANAGER}"

[gatekeeper/fork]
type=fork
job_manager=globus-job-manager

[gatekeeper/${JOB_MANAGER}]
```

```
type=${JOB_MANAGER}

EOF
else
cat <<EOF >> /etc/globus.conf
[gatekeeper]
default_jobmanager=fork
job_manager_path=${GLOBUS_LOCATION}/libexec

jobmanagers="fork "

[gatekeeper/fork]
type=fork
job_manager=globus-job-manager
EOF
fi

$INSTALL_ROOT/globus/sbin/globus-initialization.sh 2>> $YAIM_LOG

if [ "$GLOBUS_MDS" = "yes" ]; then
    /sbin/chkconfig globus-mds on
    /sbin/service globus-mds stop
    /sbin/service globus-mds start
fi
if [ "$GLOBUS_GATEKEEPER" = "yes" ]; then
    /sbin/chkconfig globus-gatekeeper on
    /sbin/service globus-gatekeeper stop
    /sbin/service globus-gatekeeper start
fi
if [ "$GLOBUS_GRIDFTP" = "yes" ]; then
    /sbin/chkconfig globus-gridftp on
    /sbin/service globus-gridftp stop
    /sbin/service globus-gridftp start
    /sbin/chkconfig lcg-mon-gridftp on
    /etc/rc.d/init.d/lcg-mon-gridftp restart
fi

return 0
}
```

27.15. CONFIG_FMON_CLIENT

```
config_fmon_client(){

# Modified by Cristina Aiftimiei (aiftim <at> pd.infn.it):
# Modified by Enrico Ferro (enrico.ferro <at> pd.infn.it)
#  host kernel version no more published
#  user DN hidden by default
#  job monitoring resource refresh for jobs in on Q/R status disabled by default
#  support new job monitoring probe
#  support new LRMSInfo probe
```

```
INSTALL_ROOT=${INSTALL_ROOT:-/opt}

requires GRIDICE_SERVER_HOST

mkdir -p ${INSTALL_ROOT}/edg/var/etc
> ${INSTALL_ROOT}/gridice/monitoring/etc/gridice-role.cfg

# Job-Monitoring parameters
JM_TMP_DIR=/var/spool/gridice/JM
LAST_HOURS_EXEC_JOBS=2
mkdir -p ${JM_TMP_DIR}/new
mkdir -p ${JM_TMP_DIR}/ended
mkdir -p ${JM_TMP_DIR}/subject
mkdir -p ${JM_TMP_DIR}/processed

#  Monitoring of processes/daemon with gridice
if ( echo "${NODE_TYPE_LIST}" | grep CE > /dev/null ); then
cat <<EOF >>${INSTALL_ROOT}/gridice/monitoring/etc/gridice-role.cfg
[ce-access-node]
gsiftp ^[\s\w\/\.-]*ftpd
edg-gatekeeper ^[\s\w\/\.-]*edg-gatekeeper
globus-mds ^[\s\w\/\.-]*${INSTALL_ROOT}/globus/libexec/slapd
fmon-agent ^[\s\w\/\.-]*fmon-agent
lcg-bdii-fwd    ^[\s\w\/\.-]*bdii-fwd
lcg-bdii-update ^[\w\/\.-]*perl\s[\s\w\/\.-]*bdii-update
lcg-bdii-slapd  ^[\w\/\.-]*slapd\s[\s\w\/\.\-]*bdii
EOF

if [ "$CE_BATCH_SYS" = "torque" ] || [ "$CE_BATCH_SYS" = "pbs" ] || [ "$CE_BATCH_SYS" = "lcgpbs" ]; then
   cat <<EOF >>${INSTALL_ROOT}/gridice/monitoring/etc/gridice-role.cfg
pbs-server ^[\s\w\/\.-]*pbs_server
maui ^[\s\w\/\.-]*maui
EOF
fi
if [ "$CE_BATCH_SYS" = "lsf" ]; then
   cat <<EOF >>${INSTALL_ROOT}/gridice/monitoring/etc/gridice-role.cfg
lsf-lim ^[\s\w\/\.-]*lim
lsf-pim ^[\s\w\/\.-]*pim
lsf-res ^[\s\w\/\.-]*res
lsf-sbatchd ^[\s\w\/\.-]*sbatchd
EOF
MASTER="`lsclusters |grep -v MASTER |awk '{print \$3}'`"
if [ "$CE_HOST" = "$MASTER" -o "$CE_HOST" = "$MASTER.$MY_DOMAIN" ]; then
       cat <<EOF >>${INSTALL_ROOT}/gridice/monitoring/etc/gridice-role.cfg
lsf-mbatchd ^[\s\w\/\.-]*mbatchd
EOF
fi

fi
cat <<EOF >>${INSTALL_ROOT}/gridice/monitoring/etc/gridice-role.cfg
[ce-access-node end]
EOF
fi
```

```
if ( echo "${NODE_TYPE_LIST}" | grep SE > /dev/null ); then
cat <<EOF >>${INSTALL_ROOT}/gridice/monitoring/etc/gridice-role.cfg
[se-access-node]
gsiftp ^[\s\w\/\.-]*ftpd
globus-mds ^[\s\w\/\.-]*${INSTALL_ROOT}/globus/libexec/slapd.*2135.*
fmon-agent ^[\s\w\/\.-]*fmon-agent
[se-access-node end]
EOF
fi

if ( echo "${NODE_TYPE_LIST}" | grep SE_dpm_mysql > /dev/null ); then
cat <<EOF >>${INSTALL_ROOT}/gridice/monitoring/etc/gridice-role.cfg
[dpm-master-node]
globus-mds ^[\s\w\/\.-]*/opt/globus/libexec/slapd.*2135.*
fmon-agent ^[\s\w\/\.-]*fmon-agent
dpm-master ^[\s\w\/\.-]*dpm
dpm-names ^[\s\w\/\.-]*dpnsdaemon
MySQL ^[\s\w\/\.-]*mysqld
srm-v1-interface ^[\s\w\/\.-]*srmv1
srm-v2-interface ^[\s\w\/\.-]*srmv2
gsiftp ^[\w,\/,-]*ftpd
rfio ^[\w,\/,-]*rfiod
[dpm-master-node end]
EOF
fi

if ( echo "${NODE_TYPE_LIST}" | grep SE_dpm_disk > /dev/null ); then
cat <<EOF >>${INSTALL_ROOT}/gridice/monitoring/etc/gridice-role.cfg
[dpm-pool-node]
gsiftp ^[\w,\/,-]*ftpd
rfio ^[\w,\/,-]*rfiod
[dpm-pool-node end]
EOF
fi

if [ "X$GRIDICE_SERVER_HOST" = "X`hostname -f`" ]; then
cat <<EOF >>${INSTALL_ROOT}/gridice/monitoring/etc/gridice-role.cfg
[gridice-collector]
gridice-mds ^[\s\w\/\.-]*${INSTALL_ROOT}/globus/libexec/slapd.*2136.*
fmon-server ^[\s\w\/\.-]*fmon-server
[gridice-collector end]
EOF
fi

if [ "X$MON_HOST" = "X`hostname -f`" ]; then
cat <<EOF >>${INSTALL_ROOT}/gridice/monitoring/etc/gridice-role.cfg
[rgma-monbox]
ntpd ^[\s\w\/\.-]*ntpd
tomcat [\s\w\/\.-]tomcat
fmon-agent ^[\s\w\/\.-]*fmon-agent
[rgma-monbox end]
EOF
fi
```

```
if ( echo "${NODE_TYPE_LIST}" | grep RB > /dev/null ); then
cat <<EOF >>${INSTALL_ROOT}/gridice/monitoring/etc/gridice-role.cfg
[broker]
ftp-server              ^[\s\w\/\.-]*ftpd
job-controller          ^[\s\w\/\.-]*edg-wl-job_controller
condor-master           ^[\s\w\/\.-]*condor_master
logging-and-bookkeeping ^[\s\w\/\.-]*edg-wl-bkserverd
condorg-scheduler       ^[\s\w\/\.-]*condor_schedd
log-monitor             ^[\s\w\/\.-]*edg-wl-log_monitor
local-logger            ^[\s\w\/\.-]*edg-wl-logd
local-logger-interlog   ^[\s\w\/\.-]*edg-wl-interlogd
network-server          ^[\s\w\/\.-]*edg-wl-ns_daemon
proxy-renewal           ^[\s\w\/\.-]*edg-wl-renewd
workload-manager        ^[\s\w\/\.-]*edg-wl-workload_manager
fmon-agent              ^[\s\w\/\.-]*fmon-agent
[broker end]
EOF
fi

if ( echo "${NODE_TYPE_LIST}" | grep BDII > /dev/null ); then
cat <<EOF >>${INSTALL_ROOT}/gridice/monitoring/etc/gridice-role.cfg
[bdii]
lcg-bdii-fwd    ^[\s\w\/\.-]*bdii-fwd
lcg-bdii-update ^[\w\/\.-]*perl\s[\s\w\/\.-]*bdii-update
lcg-bdii-slapd  ^[\w\/\.-]*slapd\s[\s\w\/\.\-]*bdii
fmon-agent      ^[\s\w\/\.-]*fmon-agent
[bdii end]
EOF
fi


# Configuration File for JobMonitoring
# If not definied before, use these defaults
GRIDICE_HIDE_USER_DN=${GRIDICE_HIDE_USER_DN:-yes}
GRIDICE_REFRESH_INFO_JOBS=${GRIDICE_REFRESH_INFO_JOBS:-no}

cat <<EOF >${INSTALL_ROOT}/gridice/monitoring/etc/JM.conf
##
## /opt/gridice/monitoring/etc/JM.conf
##

LRMS_TYPE=${CE_BATCH_SYS}

# --jm-dir=<$JM_TMP_PATH> (default /var/spool/gridice/JM) -- inside this directory
#                 will be created "new/" "ended/" "subject/" "processed/";
#                 when messlog_mon.pl is restarted it has to delete all
#                 "processed/.jmgridice*" files
JM_TMP_DIR=${JM_TMP_DIR}

# "--lrms-path=<LRMS_SPOOL_DIR>" (path for logs of batch-system)
LRMS_SPOOL_DIR=${BATCH_LOG_DIR}

# "--hide-subject=<yes|no>" (default: yes)
HIDE_USER_DN=${GRIDICE_HIDE_USER_DN}
```

```
# "--interval=<interval for ended jobs>", in hours (default: 2)
LAST_HOURS_EXEC_JOBS=${LAST_HOURS_EXEC_JOBS}

# <yes|no> (set the parameter "--no-update" if "no", otherwise no parameter is passed)
REFRESH_INFO_FOR_RUNNING_JOBS=${GRIDICE_REFRESH_INFO_JOBS}
EOF

# End configuration File for JobMonitoring

cat <<EOF >${INSTALL_ROOT}/edg/var/etc/edg-fmon-agent.conf
# template Sensor file for edg-fmonagent
# ** DO NOT EDIT **
# Generated from template: /usr/lib/lcfg/conf/fmonagent/sensors.cfg

MSA



Transport



UDP
Server ${GRIDICE_SERVER_HOST}
Port 12409
FilterMetrics KeepOnly
11001
11011
11021
11101
11202
11022
11031
11201
10100
10102
10103
10104
EOF
if ( echo "${NODE_TYPE_LIST}" | grep CE > /dev/null ); then
cat <<EOF >>${INSTALL_ROOT}/edg/var/etc/edg-fmon-agent.conf
TCP
Server ${GRIDICE_SERVER_HOST}
Port 12409
FilterMetrics KeepOnly
10106
10107
EOF
fi
cat <<EOF >>${INSTALL_ROOT}/edg/var/etc/edg-fmon-agent.conf

Sensors
```

```
edtproc
CommandLine ${INSTALL_ROOT}/gridice/monitoring/bin/GLUEsensorLinuxProc
MetricClasses
edt.uptime
edt.cpu
edt.memory
edt.disk
edt.network
edt.ctxint
edt.swap
edt.processes
edt.sockets
edt.cpuinfo
edt.os
edt.alive
edt.regfiles

sensor1
CommandLine ${INSTALL_ROOT}/edg/libexec/edg-fmon-sensor-systemCheck
MetricClasses
executeScript


Metrics
11001
MetricClass edt.uptime
11011
MetricClass edt.cpu
11021
MetricClass edt.memory
11101
MetricClass edt.disk
11202
MetricClass edt.network
Parameters
interface       eth0
11013
MetricClass edt.ctxint
11022
MetricClass edt.swap
11031
MetricClass edt.processes
11201
MetricClass edt.sockets
10100
MetricClass edt.cpuinfo
10102
MetricClass edt.alive
10103
MetricClass edt.regfiles
10104
MetricClass executeScript
Parameters
```

```
command ${INSTALL_ROOT}/gridice/monitoring/bin/CheckDaemon.pl --cfg ${INSTALL_ROOT}/gridice/monitoring/etc/gridice-
EOF
if ( echo "${NODE_TYPE_LIST}" | grep CE > /dev/null ); then
if [ "X$GRIDICE_REFRESH_INFO_JOBS" = "Xno" ]; then
  OPT_REFRESH=" --no-update"
fi
cat <<EOF >>${INSTALL_ROOT}/edg/var/etc/edg-fmon-agent.conf
10106
MetricClass executeScript
Parameters
command  ${INSTALL_ROOT}/gridice/monitoring/bin/CheckJobs.pl --lrms=${CE_BATCH_SYS} --lrms-path=${BATCH_LOG_DIR} --
EOF
cat <<EOF >>${INSTALL_ROOT}/edg/var/etc/edg-fmon-agent.conf
10107
MetricClass executeScript
Parameters
command ${INSTALL_ROOT}/gridice/monitoring/bin/LRMSinfo.pl --lrms=${CE_BATCH_SYS}
EOF
fi
cat <<EOF >>${INSTALL_ROOT}/edg/var/etc/edg-fmon-agent.conf

Samples
verylowfreq
Timing 3600 0
Metrics
10100
lowfreq
Timing 1800 0
Metrics
11001
EOF
if ( echo "${NODE_TYPE_LIST}" | grep CE > /dev/null ) && [ "X$GRIDICE_JM" = "Xyes" ]; then
cat <<EOF >>${INSTALL_ROOT}/edg/var/etc/edg-fmon-agent.conf
midfreq
Timing 1200 0
Metrics
10106
EOF
fi
cat <<EOF >>${INSTALL_ROOT}/edg/var/etc/edg-fmon-agent.conf
proc0
Timing 30 0
Metrics
10102
proc1
Timing 60 0
Metrics
11011
11021
11101
11202
11022
11031
11201
```

```
proc2
Timing 300 0
Metrics
10103
EOF
if ( echo "${NODE_TYPE_LIST}" | grep CE > /dev/null ); then
cat <<EOF >>${INSTALL_ROOT}/edg/var/etc/edg-fmon-agent.conf
10107
EOF
fi
cat <<EOF >>${INSTALL_ROOT}/edg/var/etc/edg-fmon-agent.conf
proc3
Timing 120 0
Metrics
10104
EOF

# Configure the job monitoring daemon only on CE
if ( echo "${NODE_TYPE_LIST}" | grep CE > /dev/null ); then
  /sbin/chkconfig gridice_daemons on
  /sbin/service gridice_daemons stop
  /sbin/service gridice_daemons start
fi

/sbin/chkconfig edg-fmon-agent on
/sbin/service edg-fmon-agent stop
/sbin/service edg-fmon-agent start

# The cron job required was originally installed under
# the spurious name edg-fmon-knownhosts
if [ -f ${CRON_DIR}/edg-fmon-knownhosts ]; then
    rm -f ${CRON_DIR}/edg-fmon-knownhosts
fi

if [ "X$GRIDICE_SERVER_HOST" = "X`hostname -f`" ]; then
/sbin/chkconfig edg-fmon-server on
/sbin/chkconfig gridice-mds on
/sbin/service edg-fmon-server stop
/sbin/service edg-fmon-server start
/sbin/service gridice-mds stop
/sbin/service gridice-mds start

cron_job edg-fmon-cleanspool root "41 1 * * * ${INSTALL_ROOT}/edg/sbin/edg-fmon-cleanspool &> /dev/null"

#Clean up any remaining sensitive information
find /var/fmonServer/ -name 'last.00010101' -exec rm -f '{}' \;

fi


return 0
}
```

27.16. CONFIG_LCGENV

```
config_lcgenv() {

INSTALL_ROOT=${INSTALL_ROOT:-/opt}

if !( echo "${NODE_TYPE_LIST}" | grep TAR > /dev/null ); then
LCG_ENV_LOC=/etc/profile.d
else
LCG_ENV_LOC=${INSTALL_ROOT}/etc/env.d
fi

requires BDII_HOST SITE_NAME CE_HOST

if ( ! echo "${NODE_TYPE_LIST}" | grep -q UI ); then
    requires VOS VO__SW_DIR SE_LIST
fi

default_se="${SE_LIST%% *}"

if [ "$default_se" ]; then
    for VO in `echo $VOS | tr '[:lower:]' '[:upper:]'`; do
if [ "x`eval echo '$'VO_${VO}_DEFAULT_SE`" = "x" ]; then
    eval VO_${VO}_DEFAULT_SE=$default_se
fi
    done
fi

########## sh ##########
cat << EOF > ${LCG_ENV_LOC}/lcgenv.sh
#!/bin/sh
if test "x\${LCG_ENV_SET+x}" = x; then
export LCG_GFAL_INFOSYS=$BDII_HOST:2170
EOF

if [ "$PX_HOST" ]; then
echo "export MYPROXY_SERVER=$PX_HOST" >> ${LCG_ENV_LOC}/lcgenv.sh
fi

if ( echo "${NODE_TYPE_LIST}" | egrep -q 'WN|VOBOX' ); then
    if [ "$SITE_NAME" ]; then
echo "export SITE_NAME=$SITE_NAME" >> ${LCG_ENV_LOC}/lcgenv.sh
    fi

    if [ "$CE_HOST" ]; then
echo "export SITE_GIIS_URL=$CE_HOST" >> ${LCG_ENV_LOC}/lcgenv.sh
    fi
fi

if [ -d ${INSTALL_ROOT}/d-cache/srm/bin ]; then
    echo export PATH="\${PATH}:${INSTALL_ROOT}/d-cache/srm/bin:${INSTALL_ROOT}/d-cache/dcap/bin" >> ${LCG_ENV_LOC}/
fi

if [ -d ${INSTALL_ROOT}/d-cache/dcap/lib ]; then
```

```
    echo export LD_LIBRARY_PATH=\${LD_LIBRARY_PATH}:${INSTALL_ROOT}/d-cache/dcap/lib >> ${LCG_ENV_LOC}/lcgenv.sh
fi

if [ -d ${INSTALL_ROOT}/d-cache/srm ]; then
    echo export SRM_PATH=${INSTALL_ROOT}/d-cache/srm >> ${LCG_ENV_LOC}/lcgenv.sh
fi

if [ "$EDG_WL_SCRATCH" ]; then
    echo "export EDG_WL_SCRATCH=$EDG_WL_SCRATCH"  >> ${LCG_ENV_LOC}/lcgenv.sh
fi

for VO in `echo $VOS | tr '[:lower:]' '[:upper:]'`; do

    default_se=`eval echo '$'VO_${VO}_DEFAULT_SE`
    if [ "$default_se" ]; then
echo "export VO_${VO}_DEFAULT_SE=$default_se" >> ${LCG_ENV_LOC}/lcgenv.sh
    fi

    if ( ! echo "${NODE_TYPE_LIST}" | grep -q UI ); then
sw_dir=`eval echo '$'VO_${VO}_SW_DIR`
if [ "$sw_dir" ]; then
    echo "export VO_${VO}_SW_DIR=$sw_dir" >> ${LCG_ENV_LOC}/lcgenv.sh
fi
    fi

done

if [ "$VOBOX_HOST" ]; then
   requires GSSKLOG
   if  [ "${GSSKLOG}x" == "yesx" ]; then
      requires GSSKLOG_SERVER
      echo "export GSSKLOG_SERVER=$GSSKLOG_SERVER" >> ${LCG_ENV_LOC}/lcgenv.sh
   fi
fi

if [ "${DPM_HOST}" ]; then
    echo "export DPNS_HOST=${DPM_HOST}" >> ${LCG_ENV_LOC}/lcgenv.sh
    echo "export DPM_HOST=${DPM_HOST}" >> ${LCG_ENV_LOC}/lcgenv.sh
fi

if [ "$GLOBUS_TCP_PORT_RANGE" ]; then
    echo "export MYPROXY_TCP_PORT_RANGE=\"${GLOBUS_TCP_PORT_RANGE/ /,}\"" >> ${LCG_ENV_LOC}/lcgenv.sh
fi

if ( echo $NODE_TYPE_LIST | egrep -q UI ); then
    cat << EOF >> ${LCG_ENV_LOC}/lcgenv.sh
if [ "x\$X509_USER_PROXY" = "x" ]; then
    export X509_USER_PROXY=/tmp/x509up_u\$(id -u)
fi
EOF
fi

echo fi >> ${LCG_ENV_LOC}/lcgenv.sh
########## sh ##########
```

```
########## csh ##########
cat << EOF > ${LCG_ENV_LOC}/lcgenv.csh
#!/bin/csh
if ( ! \$?LCG_ENV_SET ) then
setenv LCG_GFAL_INFOSYS $BDII_HOST:2170
EOF

if [ "$PX_HOST" ]; then
echo "setenv MYPROXY_SERVER $PX_HOST" >>  ${LCG_ENV_LOC}/lcgenv.csh
fi

if ( echo "${NODE_TYPE_LIST}" | egrep -q 'WN|VOBOX' ); then
    if [ "$SITE_NAME" ]; then
echo "setenv SITE_NAME $SITE_NAME" >>  ${LCG_ENV_LOC}/lcgenv.csh
    fi

    if [ "$CE_HOST" ]; then
echo "setenv SITE_GIIS_URL $CE_HOST" >>  ${LCG_ENV_LOC}/lcgenv.csh
    fi
fi

if [ -d ${INSTALL_ROOT}/d-cache/srm/bin ]; then
    echo setenv PATH "\${PATH}:${INSTALL_ROOT}/d-cache/srm/bin:${INSTALL_ROOT}/d-cache/dcap/bin" >> ${LCG_ENV_LOC}/
fi

if [ -d ${INSTALL_ROOT}/d-cache/dcap/lib ]; then
    echo setenv LD_LIBRARY_PATH \${LD_LIBRARY_PATH}:${INSTALL_ROOT}/d-cache/dcap/lib >> ${LCG_ENV_LOC}/lcgenv.csh
fi

if [ -d ${INSTALL_ROOT}/d-cache/srm ]; then
    echo setenv SRM_PATH ${INSTALL_ROOT}/d-cache/srm >> ${LCG_ENV_LOC}/lcgenv.csh
fi

if [ "$EDG_WL_SCRATCH" ]; then
    echo "setenv EDG_WL_SCRATCH $EDG_WL_SCRATCH"  >> ${LCG_ENV_LOC}/lcgenv.csh
fi

for VO in `echo $VOS | tr '[:lower:]' '[:upper:]'`; do

    default_se=`eval echo '$'VO_${VO}_DEFAULT_SE`
    if [ "$default_se" ]; then
echo "setenv VO_${VO}_DEFAULT_SE $default_se" >> ${LCG_ENV_LOC}/lcgenv.csh
    fi

    if ( ! echo "${NODE_TYPE_LIST}" | grep -q UI ); then
sw_dir=`eval echo '$'VO_${VO}_SW_DIR`
if [ "$sw_dir" ]; then
    echo "setenv VO_${VO}_SW_DIR $sw_dir" >> ${LCG_ENV_LOC}/lcgenv.csh
fi
    fi

done
```

```
if [ "$VOBOX_HOST" ]; then
   requires GSSKLOG
   if [ "${GSSKLOG}x" == "yesx" ]; then
   requires GSSKLOG_SERVER
      echo "setenv GSSKLOG_SERVER $GSSKLOG_SERVER" >> ${LCG_ENV_LOC}/lcgenv.csh
   fi
fi


if [ "${DPM_HOST}" ]; then
    echo "setenv DPNS_HOST ${DPM_HOST}" >> ${LCG_ENV_LOC}/lcgenv.csh
    echo "setenv DPM_HOST ${DPM_HOST}" >> ${LCG_ENV_LOC}/lcgenv.csh
fi

if [ "$GLOBUS_TCP_PORT_RANGE" ]; then
    echo "setenv MYPROXY_TCP_PORT_RANGE \"${GLOBUS_TCP_PORT_RANGE/ /,}\"" >> ${LCG_ENV_LOC}/lcgenv.csh
fi

if ( echo $NODE_TYPE_LIST | egrep -q UI ); then
    cat << EOF >> ${LCG_ENV_LOC}/lcgenv.csh
if ( ! \$?X509_USER_PROXY ) then
   setenv X509_USER_PROXY /tmp/x509up_u\`id -u\`
endif
EOF
fi

echo endif >> ${LCG_ENV_LOC}/lcgenv.csh
########## csh ##########

chmod a+xr ${LCG_ENV_LOC}/lcgenv.csh
chmod a+xr ${LCG_ENV_LOC}/lcgenv.sh

return 0
}
```

27.17. CONFIG_BDII

```
config_bdii(){
#
# Configures the BDII.
#
# If SITE_BDII=yes configures as a site BDII otherwise top level
#
# Uses CE_HOST SE HOST RB_HOST and PX_HOST.
#
# These values should be changed the common ones.
#

requires BDII_HOST

INSTALL_ROOT=${INSTALL_ROOT:-/opt}
```

```
mkdir -p $INSTALL_ROOT/bdii/etc
chown edguser $INSTALL_ROOT/bdii/etc
mkdir -p $INSTALL_ROOT/bdii/var
chown edguser $INSTALL_ROOT/bdii/var

BDII_AUTO_MODIFY=no

if ( ! echo "${NODE_TYPE_LIST}" | grep BDII > /dev/null ); then

# We're a site BDII

requires BDII_REGIONS BDII__URL

BDII_BIND=mds-vo-name=$SITE_NAME,o=grid
BDII_AUTO_UPDATE=no

rm -f $INSTALL_ROOT/bdii/etc/bdii-update.conf
for REGION in $BDII_REGIONS; do
            echo "$REGION  `eval echo '$'BDII_${REGION}_URL`" >> $INSTALL_ROOT/bdii/etc/bdii-update.conf
done

else

# We're a top level BDII

requires BDII_HTTP_URL

if [ "$BDII_FCR" ]; then
    BDII_AUTO_MODIFY=yes
fi

BDII_BIND=mds-vo-name=local,o=grid
BDII_AUTO_UPDATE=yes

fi

pass=`mkpasswd -s 0 2> /dev/null` || pass=$RANDOM

cat << EOF > $INSTALL_ROOT/bdii/etc/bdii.conf
BDII_PORT_READ=2170
BDII_PORTS_WRITE="2171 2172 2173"
BDII_USER=edguser
BDII_BIND=$BDII_BIND
BDII_PASSWD=$pass
BDII_SEARCH_FILTER='*'
BDII_SEARCH_TIMEOUT=30
BDII_BREATHE_TIME=60
BDII_AUTO_UPDATE=$BDII_AUTO_UPDATE
BDII_AUTO_MODIFY=$BDII_AUTO_MODIFY
BDII_DIR=$INSTALL_ROOT/bdii/
BDII_UPDATE_URL=$BDII_HTTP_URL
BDII_UPDATE_LDIF=${BDII_FCR:-http://}
SLAPD=/usr/sbin/slapd
```

```
SLAPADD=/usr/sbin/slapadd

EOF

cat << EOF > $INSTALL_ROOT/bdii/etc/schemas
/etc/openldap/schema/core.schema
/opt/glue/schema/ldap/Glue-CORE.schema
/opt/glue/schema/ldap/Glue-CE.schema
/opt/glue/schema/ldap/Glue-CESEBind.schema
/opt/glue/schema/ldap/Glue-SE.schema
/opt/lcg/schema/ldap/SiteInfo.schema

EOF

/sbin/chkconfig --add bdii
/sbin/service bdii stop
/sbin/service bdii start

return 0
}
```

27.18. CONFIG_WORKLOAD_MANAGER_ENV

```
config_workload_manager_env(){

INSTALL_ROOT=${INSTALL_ROOT:-/opt}

mkdir -p $INSTALL_ROOT/edg/var/etc/profile.d

cp $INSTALL_ROOT/edg/etc/profile.d/edg-wl.csh $INSTALL_ROOT/edg/etc/profile.d/edg-wl.sh $INSTALL_ROOT/edg/var/etc/p

return 0
}
```

27.19. CONFIG_WM_LOCALLOGGER

```
config_wm_locallogger(){

INSTALL_ROOT=${INSTALL_ROOT:-/opt}

/sbin/chkconfig edg-wl-locallogger on
/etc/rc.d/init.d/edg-wl-locallogger restart

cron_job edg-wl-locallogger root "56 2,8,14,20 * * * /sbin/service edg-wl-locallogger proxy"

return 0
}
```

27.20. CONFIG_APEL_PBS

```
config_apel_pbs(){

INSTALL_ROOT=${INSTALL_ROOT:-/opt}

TORQUE_SERVER=${TORQUE_SERVER:-${CE_HOST}}

requires MON_HOST SITE_NAME CE_HOST APEL_DB_PASSWORD

cat ${INSTALL_ROOT}/glite/etc/glite-apel-pbs/parser-config.xml | sed \
    -e "s/localhost/${MON_HOST}/" \
    -e "s/<DBUsername>.*/<DBUsername>accounting<\/DBUsername>/" \
    -e "s/<DBPassword>.*/<DBPassword>${APEL_DB_PASSWORD}<\/DBPassword>/" \
    -e "s/<SubmitHost>.*/<SubmitHost>${TORQUE_SERVER}<\/SubmitHost>/" \
    -e "s/<SiteName>.*/<SiteName>${SITE_NAME}<\/SiteName>/" \
    -e "s/<GIIS>.*/<GIIS>${CE_HOST}<\/GIIS>/" \
    -e "/<DBDeleteProcessor/d" \
    -e "/<ExtraFile/d" \
> ${INSTALL_ROOT}/glite/etc/glite-apel-pbs/parser-config-yaim.xml


chown root:root ${INSTALL_ROOT}/glite/etc/glite-apel-pbs/parse\r-config-yaim.xml
chmod 600 ${INSTALL_ROOT}/glite/etc/glite-apel-pbs/parse\r-config-yaim.xml

# Remove confusion with two different jobs being called edg-rgma-apel
if [ -f ${CRON_DIR}/edg-rgma-apel ]; then
    rm -f ${CRON_DIR}/edg-rgma-apel
fi

cron_job edg-apel-pbs-parser root "35 01 * * * env RGMA_HOME=${INSTALL_ROOT}/glite APEL_HOME=${INSTALL_ROOT}/glite

return 0

}
```

27.21. CONFIG_LCMAPS

```
config_lcmaps(){

INSTALL_ROOT=${INSTALL_ROOT:-/opt}

cat <<EOF >${INSTALL_ROOT}/edg/etc/lcmaps/lcmaps.db
# LCMAPS policyfile generated by YAIM - DO NOT EDIT
#
# where to look for modules
path = ${INSTALL_ROOT}/edg/lib/lcmaps/modules

# module definitions
posixenf = "lcmaps_posix_enf.mod -maxuid 1 -maxpgid 1 -maxsgid 32 "
localaccount = "lcmaps_localaccount.mod  -gridmapfile /etc/grid-security/grid-mapfile"
poolaccount = "lcmaps_poolaccount.mod  -gridmapfile /etc/grid-security/grid-mapfile -gridmapdir /etc/grid-security/
vomsextract = "lcmaps_voms.mod  -vomsdir /etc/grid-security/vomsdir/ -certdir /etc/grid-security/certificates/"
```

```
vomslocalgroup = "lcmaps_voms_localgroup.mod  -groupmapfile ${INSTALL_ROOT}/edg/etc/lcmaps/groupmapfile -mapmin 0"
vomspoolaccount = "lcmaps_voms_poolaccount.mod  -gridmapfile ${INSTALL_ROOT}/edg/etc/lcmaps/gridmapfile -gridmapdir
vomslocalaccount = "lcmaps_voms_localaccount.mod  -gridmapfile ${INSTALL_ROOT}/edg/etc/lcmaps/gridmapfile -use_voms

# policies
voms:
vomsextract -> vomslocalgroup
vomslocalgroup -> vomspoolaccount
vomspoolaccount -> posixenf | vomslocalaccount
vomslocalaccount -> posixenf

standard:
localaccount -> posixenf | poolaccount
poolaccount -> posixenf

EOF
}
```

27.22. CONFIG_LCAS

```
config_lcas (){

INSTALL_ROOT=${INSTALL_ROOT:-/opt}

if [ ! -f ${INSTALL_ROOT}/edg/etc/lcas/ban_users.db ]; then
    cp -f ${INSTALL_ROOT}/edg/etc/lcas/ban_users.db.in ${INSTALL_ROOT}/edg/etc/lcas/ban_users.db
fi

cat <<EOF >${INSTALL_ROOT}/edg/etc/lcas/timeslots.db
#
# This file contains the time slots for which the fabric
# is available for Grid jobs
# Format:
#        minute1-minute2 hour1-hour2 mday1-mday2 month1-month2 year1-year2 wday1-wday2
# max range:  [0-59]        [0-23]      [1-31]       [1-12]    [1970-...]     [0-6]
#
# wday:
#  0-6 = Sunday-Saturday
#  5-3 = Friday-Wednesday
#
# '*' means the maximum range
# <val>- means from <val> to maximum value
#
# The wall clock time should match at least one time slot for authorization
# The wall clock time matches if:
#           (hour1:minute1)      <= (hour:minute)     <= (hour2:minute2)
#       AND (year1.month1.mday1) <= (year.month.mday) <= (year2.month2.mday2)
#       AND (wday1)              <= (wday)            <= (wday2)
#
# If the fabric is open on working days from 8:30-18:00 h, from 1 July 2002 till 15 January 2003
# the following line should be added:
#      30-0           8-18        1-15         7-1       2002-2003      1-5
```

```
# If the fabric is open from 18:00-7:00 h, two time slots should be used:
#      18:00-24:00 and 0:00-7:00
#
#       0-0          18-24         *            *            *           *
#       0-0           0-7          *            *            *           *
# If the fabric is always open the following line should be uncommented:
# minute1-minute2 hour1-hour2 mday1-mday2 month1-month2 year1-year2 wday1-wday2
* * * * * *
0-0 23-24 * * * *
EOF

cat <<EOF> ${INSTALL_ROOT}/edg/etc/lcas/lcas.db
# LCAS database/plugin list
#
# Format of each line:
# pluginname="<name/path of plugin>", pluginargs="<arguments>"
#
#
pluginname=lcas_userban.mod,pluginargs=ban_users.db
pluginname=lcas_timeslots.mod,pluginargs=timeslots.db
pluginname=lcas_plugin_example.mod,pluginargs=arguments
EOF

}
```

27.23. CONFIG_TORQUE_SUBMITTER_SSH

```
config_torque_submitter_ssh (){

TORQUE_SERVER=${TORQUE_SERVER:-${CE_HOST}}

requires WN_LIST QUEUES TORQUE_SERVER

INSTALL_ROOT=${INSTALL_ROOT:-/opt}

se_host="${SE_LIST%% *}"

echo $TORQUE_SERVER > /var/spool/pbs/server_name

cat <<EOF > $INSTALL_ROOT/edg/etc/edg-pbs-shostsequiv.conf
# Example configuration file for the edg-pbs-shostsequiv script
# File where the list of nodes will be written
SHOSTSEQUIV = /etc/ssh/shosts.equiv
# List of nodes to be included in the SHOSTSEQUIV file even if not reported
# by the pbsnodes command
NODES       =
# Location of the pbsnodes command
PBSBIN      = /usr/bin
EOF

cron_job edg-pbs-shostsequiv root "05 1,7,13,19 * * * ${INSTALL_ROOT}/edg/sbin/edg-pbs-shostsequiv"
```

```
$INSTALL_ROOT/edg/sbin/edg-pbs-shostsequiv

cat <<EOF > ${INSTALL_ROOT}/edg/etc/edg-pbs-knownhosts.conf
NODES = $CE_HOST $se_host
PBSBIN = /usr/bin
KEYTYPES = rsa1,rsa,dsa
KNOWNHOSTS = /etc/ssh/ssh_known_hosts
EOF

# workaround for duplicate key entries (Savannah bug 5530) for hostname
for hostname in $CE_HOST $se_host; do
   if [ -f /etc/ssh/ssh_known_hosts ];then
      grep -v $hostname /etc/ssh/ssh_known_hosts  > /etc/ssh/ssh_known_hosts.tmp
      /usr/bin/ssh-keyscan -t rsa $hostname >> /etc/ssh/ssh_known_hosts.tmp 2>/dev/null

      if [ $? = 0 ]; then
         mv -f /etc/ssh/ssh_known_hosts.tmp  /etc/ssh/ssh_known_hosts
      fi
   fi
done

cron_job edg-pbs-knownhosts root "03 1,7,13,19 * * * ${INSTALL_ROOT}/edg/sbin/edg-pbs-knownhosts"

${INSTALL_ROOT}/edg/sbin/edg-pbs-knownhosts

result=`tail -3 /etc/ssh/sshd_config | grep HostbasedAuthentication | grep yes`
if [ "x$result" = "x" ]; then
cat <<EOF >> /etc/ssh/sshd_config
        HostbasedAuthentication yes
        IgnoreUserKnownHosts yes
        IgnoreRhosts yes
EOF
fi

/etc/rc.d/init.d/sshd reload

return 0
}
```

27.24. CONFIG_TORQUE_SERVER

```
config_torque_server(){

TORQUE_SERVER=${TORQUE_SERVER:-${CE_HOST}}

requires TORQUE_SERVER WN_LIST QUEUES

INSTALL_ROOT=${INSTALL_ROOT:-/opt}

if ( echo "${NODE_TYPE_LIST}" | grep -q CE_torque ); then
    if [ "x${TORQUE_SERVER}" != "x${CE_HOST}" ]; then
```

```
# We're a CE but not the torque server
return 0
    fi
fi

if [ "x`grep pbs /etc/services`" = "x" ]; then
cat << EOF  >> /etc/services
pbs 15001/tcp
pbs_mom         15002/tcp
pbs_resmon       15003/tcp
pbs_resmon       15003/udp
EOF
fi

echo $TORQUE_SERVER > /var/spool/pbs/server_name

rm -f /var/spool/pbs/server_priv/nodes
for node in `cat $WN_LIST `; do
    echo "$node np=$CE_SMPSIZE lcgpro" >> /var/spool/pbs/server_priv/nodes
done

if [ ! "x`/etc/rc.d/init.d/pbs_server status | grep stopped`" = "x" ]; then
     /etc/rc.d/init.d/pbs_server start
fi

# Sometimes the pbs_server doesn't respond now.

if ( ! qmgr -c 'p s' > /dev/null ); then
    sleep 10
    /etc/rc.d/init.d/pbs_server restart
fi

for que in `echo $QUEUES | sed 's/"//g'` ; do
    result=`qstat -Q | grep $que`
    if [ "x$result" = "x" ]; then
/usr/bin/qmgr <<EOF
create queue $que
EOF
    fi

done


/usr/bin/qmgr <<EOF

set server scheduling = True
set server acl_host_enable = False
set server managers = root@`hostname -f`
set server operators = root@`hostname -f`
set server default_queue = dteam
set server log_events = 511
set server mail_from = adm
set server query_other_jobs = True
set server scheduler_iteration = 600
```

```
set server default_node = lcgpro
set server node_pack = False

EOF

for QUEUE in `echo $QUEUES | sed 's/"//g'` ; do
/usr/bin/qmgr <<EOF
set queue $QUEUE queue_type = Execution
set queue $QUEUE resources_max.cput = 48:00:00
set queue $QUEUE resources_max.walltime = 72:00:00
set queue $QUEUE enabled = True
set queue $QUEUE started = True
set queue $QUEUE acl_group_enable = True
EOF

for VO in `echo $VOS | tr '[:lower:]' '[:upper:]'`; do
    if eval echo \$VO_${VO}_QUEUES | grep -wq "$QUEUE"; then
if ( ! qmgr -c "list queue $QUEUE acl_groups" | grep acl_groups | grep -qwi $VO ); then
    /usr/bin/qmgr -c "set queue $QUEUE acl_groups += `echo $VO | tr '[:upper:]' '[:lower:]'`"
fi
    fi
done

done

/etc/rc.d/init.d/pbs_server restart

# zip server logs
cron_job server_logs root "33 3 * * * root find /var/spool/pbs/server_logs -mtime +7 -exec gzip -9 {} \; 2> /dev/nu

# If the torque server is on a separate machine from the CE
# allow communication

if [ "x${TORQUE_SERVER}" != "x${CE_HOST}" ]; then
    if ( ! grep -q ${CE_HOST} /etc/hosts.equiv 2> /dev/null ); then
echo ${CE_HOST} >>  /etc/hosts.equiv
    fi
fi

# maui

if [ "x`grep maui /etc/services`" = "x"  ]; then
    echo "maui 15004/tcp" >> /etc/services
fi


cat << EOF > /var/spool/maui/maui.cfg
# MAUI configuration example

SERVERHOST              `hostname -f`
ADMIN1                  root
ADMIN3                  edginfo rgma
ADMINHOST               `hostname -f`
RMCFG[base]             TYPE=PBS
```

```
SERVERPORT              40559
SERVERMODE              NORMAL

# Set PBS server polling interval. If you have short # queues or/and jobs it is worth to set a short interval. (10

RMPOLLINTERVAL        00:00:10

# a max. 10 MByte log file in a logical location

LOGFILE               /var/log/maui.log
LOGFILEMAXSIZE        10000000
LOGLEVEL              1

# Set the delay to 1 minute before Maui tries to run a job again, # in case it failed to run the first time.
# The default value is 1 hour.

DEFERTIME       00:01:00

EOF

/sbin/chkconfig maui on
/etc/rc.d/init.d/maui stop
sleep 1
/etc/rc.d/init.d/maui start
return 0
}
```