Guide to LCAS version 1.1.16

Martijn Steenbakkers

15 September 2003

1 Introduction

The Gridification subtask of WP4 of the European Datagrid project[1] interfaces the local fabric to other middleware components by a number of services, among which the Local Centre Authorization Service (LCAS) handles authorization requests to the local computing fabric and the Local Credential Mapping Service (LCMAPS) provides all local credentials needed for jobs allowed into the fabric. This document describes LCAS, which is the first component released by the Gridification subtask.

In this release the LCAS is a shared library, which is loaded dynamically by the globus gatekeeper. The gatekeeper has been slightly modified for this purpose and will from now on be referred to as edg-gatekeeper.

In the future the LCAS will evolve into an AAA server and can be contacted by other components, e.g. by the Storage Element.

The authorization decision of the LCAS is based upon the users's certificate and the job specification in RSL (JDL) format. The certificate and RSL are passed to (plugin) authorization modules, which grant or deny the access to the fabric. Three standard authorization modules are provided by default:

- `lcas_userallow.mod`, a module that checks if the user is allowed on the fabric (currently the gridmap file is checked). More info ...
- `lcas_userban.mod`, a module that checks if the user should be banned from the fabric. More info ...
- `lcas_timeslots.mod`, a module that checks if the fabric is open at this time of the day for datagrid jobs. More info ...

All three modules get their information from simple configuration files: `allowed_users.db` [2], `ban_users.db` and `timeslots.db`, respectively.

[1] http://www.eu-datagrid.org

[2] In this release (1.1.16) the gridmap file is used instead of `allowed_users.db`

In addition a plugin is provided that decides if the user is authorized based on the VOMS (VO Membership Service) information stored in the user proxy X509 certificate:

- `lcas_voms.mod`. More info ...

This plugin is driven by a policy file, which can have different formats:

1. text file format (just a list of allowed VO-GROUP-ROLE combinations)

2. GACL[3] format (GACL is an XML ACL language).

3. XACML[4] format (generic XML authorization language). This format is not supported yet, but will be in the future.

From release 1.1 and higher, hooks are provided for external authorization plugin modules. These plugins will be delivered by other subsystems like for example the Resource Management subsystem in order to do accounting and quota checks (for users/roles) or the Storage Element (WP5) in order to check file access or to make storage reservations. An example plugin has been added to the LCAS distribution.

More information on the LCAS and other components of the Gridification subsystem can be found in:

- the WP4 architecture document D4.2: pdf version[5] or doc version[6].

- the description of the LCAS API: here[7] , PostScript file[8] and PDF file[9].

- LCMAPS: http://www.dutchgrid.nl/DataGrid/wp4/lcmaps/[10]

- the README[11], INSTALL[12], and LICENSE[13] files.

---

[3] http://www.gridpp.ac.uk/authz/gacl/

[4] http://www.oasis-open.org/committees/xacml/

[5] http://hep-proj-grid-fabric.web.cern.ch/hep-proj-grid-fabric/architecture/eu/WP4-architecture-2_1.pdf

[6] http://hep-proj-grid-fabric.web.cern.ch/hep-proj-grid-fabric/architecture/eu/WP4-architecture-2_1.doc

[7] apidoc/html/index.html

[8] apidoc/latex/refman.ps

[9] apidoc/latex/refman.pdf

[10] http://www.dutchgrid.nl/DataGrid/wp4/lcmaps

[11] README

[12] INSTALL

[13] LICENSE

Table 1: RPMs to be installed.

RPM	min. version	description + URL
`edg-lcas`	1.1.13	the LCAS shared library with the standard and example authorization plugin modules http://datagrid.in2p3.fr/distribution/autobuild/i386-rh7.3/wp4/gridification/RPMS/
`edg-lcas-voms_plugins`	1.1.13	the LCAS plugin that uses the VOMS information for the authorization http://datagrid.in2p3.fr/distribution/autobuild/i386-rh7.3/wp4/gridification/RPMS/
`edg-lcas-interface`	1.0.3	LCAS interface/API, only needed for software development (new plugins) http://datagrid.in2p3.fr/distribution/autobuild/i386-rh7.3/wp4/gridification/RPMS/
`edg-lcfg-lcas`	1.1.2	the LCFG object that configures the LCAS authorization modules and plugin database http://datagrid.in2p3.fr/distribution/autobuild/i386-rh7.3/wp4/edg-lcfg/RPMS/
`voms-api`	1.1.16	the VOMS API, used by `edg-lcas-voms_plugins` http://datagrid.in2p3.fr/distribution/autobuild/i386-rh7.3/wp6/RPMS/
`gacl`	0.9.0	the GACL runtime library, used by `edg-lcmaps-voms_plugins` http://datagrid.in2p3.fr/distribution/autobuild/i386-rh7.3/other/RPMS/
`edg_gatekeeper-gcc32dbg_pgm`	2.2.8	the modified globus gatekeeper http://datagrid.in2p3.fr/distribution/autobuild/i386-rh7.3/wp4/gridification/RPMS/
`globus-config`	0.20-1	globus configuration scripts, including the init.d gatekeeper script http://datagrid.in2p3.fr/distribution/globus/config/RPMS/
`edg-lcfg-globuscfg`	1.3.12	the LCFG component to manage the Globus configuration http://datagrid.in2p3.fr/distribution/autobuild/i386-rh7.3/wp4/edg-lcfg/RPMS/
`vdt_globus_essentials`	VDTALT1.1.8	VDT globus rpm that contains a.o. the security libraries http://www.lsc-group.phys.uwm.edu/vdt/vdt_rpms/edg/vdt-1.1.8/globus_coarse_rpm/

2 Installation

LCAS uses the globus security libraries (gss, gsi, openssl), which are provided by e.g. VDT (Virtual Data Toolkit), the GACL libraries, and the VOMS API. These libraries in addition to the libraries listed in table1 , have to be installed on the CE.

The LCAS library will be installed in `/opt/edg/lib/lcas/` and the example configuration files in `/opt/edg/etc/lcas/`. The authorization modules and example authorization plugin will be installed in the `modules` subdirectory of the directory where the LCAS library is installed.

The path of the modified gatekeeper executable will be `/opt/edg/sbin/edg-gatekeeper`.

From CVS:
The LCAS library can also be built directly from the cvs repository[14] by the following steps:

- `cvs export -r <version_tag>  fabric_mgt/gridification/lcas` – export the source from CVS using a tagged version (e.g. v1_1_2)
- `cd fabric_mgt/gridification/lcas; ./autogen.sh` – run the bootstrap script to run autotools
- `./configure --prefix=<path> --libdir=<path>/lib/lcas --includedir=<path>/include/lcas --sysconfdir=<path>/etc/lcas` – run the configure script

[14]http://datagrid.in2p3.fr/cgi-bin/cvsweb.cgi/fabric_mgt/gridification/lcas/

- `make rpm` – if you want to make the rpm
- `make; make install` – build and install the LCAS Library and the example configuration
- `make apidoc` – if you want to create the API documentation. This is useful for developers of new authorization plugins.

3 Configuration

3.1 edg-gatekeeper

The edg-gatekeeper is configurable with a few more command line options in addition to the normal globus-gatekeeper options:

-lcmaps_debug_level <debug level>:	set the debug level for LCMAPS (default: 0 (= no debugging))
-lcmaps_db_file <file>:	specifies the filename of the LCMAPS policy file (default: `lcmaps.db`).
-lcmaps_etc_dir <path>:	specifies the directory where the LCMAPS configuration files are located (default: `/opt/edg/etc/lcmaps/`).
-lcmapsmod_dir <path>:	specifies the directory where the LCMAPS library is located (default: `/opt/edg/lib/lcmaps/`).
-lcas_debug_level <debug level>:	set the debug level for LCAS (0–5, default: 0 (= no debugging))
-lcas_db_file <file>:	specifies the filename of the LCAS policy file
(default: `lcas.db`).	
-lcas_etc_dir <path>:	specifies the directory where the LCAS authorization configuration files are located (default `/opt/edg/etc/lcas/`).
-lcas_dir <path>:	same as -lcas_etc_dir ¡path¿, deprecated.
-lcasmod_dir <path>:	specifies the directory where the LCAS library is located (default `/opt/edg/lib/lcas/`).
-plainoldglobus:	provides the old globus-gatekeeper functionality, LCAS and LCMAPS are not used.
-no_lcas:	do not use LCAS.
-no_lcmaps:	do not use LCMAPS (use standard gridmap functionality of gatekeeper).

The `globus.conf` file (usually residing in the `/etc` directory) contains the configuration parameters for the globus software. The gatekeeper init.d script uses this file to to configure the edg-gatekeeper. The following lines were added/modified in `/etc/globus.conf`:

```
[common]

[...]

[gatekeeper]

[...]

globus_gatekeeper=/opt/edg/sbin/edg-gatekeeper

extra_options="-lcas_etc_dir /opt/edg/etc/lcas/ -lcasmod_dir
/opt/edg/lib/lcas/ -lcas_db_file lcas.db -lcmaps_etc_dir /opt/edg/etc/lcmaps/
-lcmapsmod_dir /opt/edg/lib/lcmaps -lcmaps_db_file lcmaps.db"
```

The `globus_gatekeeper=` line gives the path of the gatekeeper to be used and the `extra_options=` line the gatekeeper options to be added.

LCFG configuration:
The `globus.conf` file can be created using the globus LCFG object contained in package `edg-lcfg-globuscfg`. The extra lines for the configuration files have to be specified in an LCFGng resource file in the way that is shown in the Computing Element resource file ComputingElement-cfg.h[15].

3.2 LCAS

The LCAS reads its configuration, in particular the plugins that it should load, from the file `lcas.db`. The format of this file is shown here:

```
# LCAS database/plugin list
#
# Format of each line:
# pluginname="<name/path of plugin>", pluginargs="<arguments>"
#
#
pluginname="lcas_userallow.mod",pluginargs="allowed_users.db"
pluginname="lcas_userban.mod",pluginargs="ban_users.db"
pluginname="lcas_timeslots.mod",pluginargs="timeslots.db"
pluginname="lcas_plugin_example.mod",pluginargs="Some bogus arguments"
pluginname="lcas_voms.mod",pluginargs="-vomsdir /etc/grid-security -certdir /etc/grid-security/certificates -authfile /opt/edg/etc/lcas/lcas_voms.gacl -gacl_use_voms_dn always"
```

For now only lines containing an entry for the name of a plugin (i.e. `pluginname="<path>"`) are allowed. The arguments the plugin requires are specified as `pluginargs="<arguments>"`

[15] http://datagrid.in2p3.fr/cgi-bin/cvsweb.cgi/edg-release/ng_source/ComputingElement-cfg.h

on the same line.
If no absolute path is specified for the plugin module, the LCAS will search for it in the following directories: (in order of appearance)

- `./`
- `./modules/`
- `/opt/edg/etc/lcas/`
- `/opt/opt/edg/lib/lcas/modules/`
- `/opt/opt/edg/lib/lcas/`

The three standard authorization plugins each have their own configuration database:

- `allowed_users.db`: this file is not used in this release, but will eventually partly replace the gridmap file. It will contain the list of LDAP distinguished names (DN) of the users that are allowed on the fabric. This version of the LCAS, however, still relies on the gridmap file.
- `ban_users.db`: this file contains the list of DNs of the users that should be banned from the fabric. An example is shown here:

```
#
# This file contains the globus user ids that are BANNED from this fabric
#
"/O=GetRID/O=abusers/CN=Endless Job"
```

- `timeslots.db`: This file contains the 'opening hours' of the fabric. The format of the file is shown here:

```
#
# 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 uncommented:
#      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:
         *            *           *            *           *          *
```

- the voms authorization file (specified with the `-authfile` option). This file controls which VOs, groups etc. are allowed on the fabric. Three different formats are/will be supported (specified with the `-authformat` option):
  - text file format (just a list of allowed VO-GROUP-ROLE combinations). The same files can be used as are by the LCMAPS voms plugins. An example is shown here:

    ```
    "/VO=wilma/GROUP=wilma/*" .test
    "/VO=fred/GROUP=fred/*" .test
    ```

    . N.B.: The second column is ignored.
  - GACL[16] format (GACL is an XML ACL language). Voms credentials and DNs are supported. Any kind of permission (read, write, list etc.) will result in an authorization success. An example is shown here:

    ```
    <?xml version="1.0"?>
    <gacl version="0.0.1">
    <entry>
    <voms-cred>
    <voms>/O=dutchgrid/O=hosts/OU=nikhef.nl/CN=asen.nikhef.nl</voms>
    <vo>wilma</vo>
    <group>wilma</group>
    </voms-cred>
    <allow><read/><write/></allow>
    <deny><list/></deny>
    </entry>
    <entry>
    <person>
    <dn>/O=dutchgrid/O=users/O=knmi/CN=Wim Jan Som de Cerff</dn>
    </person>
    <allow><read/></allow>
    ```

[16] http://www.gridpp.ac.uk/authz/gacl/

```
<deny><list/></deny>
</entry>
</gacl>
```

. A tool exists (`edg-lcas-voms2gacl`) to convert the textformat into gacl format.

– XACML[17] format (generic XML authorization language). This format is not supported yet, but will be in the future.

LCFG configuration:
The LCAS configuration files (except for the voms plugin policy file) can also be created using the LCAS LCFG object contained in package `edg-lcfg-lcas`. The lines for the configuration files have to be specified in an LCFG resource file in the way that is shown in the Computing Element resource file ComputingElement-cfg.h[18]. One should be careful when specifying asterixes and double quotes. The VOMS authorization file can be the same file as the `vomapfile` and `groupmapfile` from LCMAPS or can be created from those by the command `edg-lcas-voms2gacl`.

4 Adding authorization plugins

In addition to the three standard authorization plugins, new plugins may be written. The plugins have to be provided as shared objects. When the LCAS receives an authorization request, it uses `dlopen` to open the plugin shared object. The interface of the plugins to the LCAS consists of the following three functions, which are called in order with a `dlsym` call by the LCAS:

- **int plugin_initialize(int argc, char **argv)**:
  Everything that is needed to initialize the plugin should be put inside this function. Arguments as read from the LCAS database (**argc**, **argv**) are passed to the plugin.

- **int plugin_confirm_authorization(lcas_request_t request, lcas_cred_id_t lcas_cred)**:
  By this call, the LCAS asks the plugin for authorization by passing the request in RSL (later JDL) and the user credential (**lcas_cred**). The user credential will contain information on the role the user wants to play. In the RSL (JDL) the user might specify the resources he wants to

[17]http://www.oasis-open.org/committees/xacml/
[18]http://datagrid.in2p3.fr/cgi-bin/cvsweb.cgi/edg-release/ng_source/ComputingElement-cfg.h

use. The authorization decision has to be made using this information. The LCAS provides no library for parsing the RSL (JDL).

- **int plugin_terminate()**:
  Whatever is needed to terminate the plugin module goes in here.

If these symbols cannot be found by LCAS at runtime, an error occurs, resulting in an authorization failure. More information on the plugin interface can be found in the **apidoc** documentation for the plugin interface. The LCAS Library also contains utilities for logging, file checking and (extremely simple) LCAS credential handling. The API to be used by the LCAS plugins can be found in the **apidoc** documentation for the API for the plugins. In order to use these utilities a line like

```
#include "lcas_modules.h"
```

has to appear in the plugin source. A line similar to

```
-I $GLOBUS_LOCATION/include/gcc32dbg -I /opt/edg/include/lcas
```

has to be added to the compilation command line in order to include the LCAS and GLOBUS include directories.

To make life easier for the plugin developer an example plugin has been written in C, which is available in the LCAS cvs repository[19]. The example plugin is built using autotools (automake, autoconf, libtool), for which the files configure.in[20] and Makefile.am[21] have to be present. The source code is shown here:

```
/*
 * Copyright (c) 2001 EU DataGrid.
 * For license conditions see http://www.eu-datagrid.org/license.html
 *
 * Copyright (c) 2001, 2002 by
 *     Martijn Steenbakkers <martijn@nikhef.nl>,
 *     David Groep <davidg@nikhef.nl>,
 *     NIKHEF Amsterdam, the Netherlands
 */

/*!
    \file   lcas_plugin_example.c
    \brief  Interface to the LCAS plugins
    \author Martijn Steenbakkers for the EU DataGrid.

    This file contains the code for an example LCAS plugin and shows the interface
    the plugin has to provide to the LCAS. The interface consists of the following
    functions:
    -# plugin_initialize()
    -# plugin_confirm_authorization()
    -# plugin_terminate()
*/
/*!
    \defgroup LcasPluginInterface The interface to the LCAS plugins

    Here the interface is shown that the plugin has to provide to the LCAS.
    The interface consists of the following functions:
```

[19]http://datagrid.in2p3.fr/cgi-bin/cvsweb.cgi/fabric_mgt/gridification/lcas/plugin_example
[20]http://datagrid.in2p3.fr/cgi-bin/cvsweb.cgi/fabric_mgt/gridification/lcas/configure.in
[21]http://datagrid.in2p3.fr/cgi-bin/cvsweb.cgi/fabric_mgt/gridification/lcas/plugin_example/Makefile.am

```
    -# plugin_initialize()
    -# plugin_confirm_authorization()
    -# plugin_terminate()
*/
/*\@{*/

/*****************************************************************************
                            Include header files
******************************************************************************/
#include "lcas_config.h"
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include "lcas_modules.h"

/******************************************************************************
                                Definitions
******************************************************************************/

/******************************************************************************
                          Module specific prototypes
******************************************************************************/

/******************************************************************************
                       Define module specific variables
******************************************************************************/

/******************************************************************************
Function:   plugin_initialize
Description:
    Initialize plugin
Parameters:
    argc, argv
    argv[0]: the name of the plugin
Returns:
    LCAS_MOD_SUCCESS : succes
    LCAS_MOD_FAIL    : failure
    LCAS_MOD_NOFILE  : db file not found (will halt LCAS initialization)
******************************************************************************/
/*!
    \fn plugin_initialize(
        int argc,
        char ** argv
        )
    \brief initialize the plugin.

    Everything that is needed to initialize the plugin should be put inside this function.
    Arguments as read from the LCAS database (argc, argv) are passed to the plugin.

    \param argc number of passed arguments.
    \param argv argument list. argv[0] contains the name of the plugin.
    \retval LCAS_MOD_SUCCESS successful initialization
    \retval LCAS_MOD_FAIL    failure in the plugin initialization
    \retval LCAS_MOD_NOFILE  private plugin database could not be found (same effect as
                             LCAS_MOD_FAIL)
*/
int plugin_initialize(
        int argc,
        char ** argv
)
{
    int i;

    lcas_log_debug(1,"lcas_plugin_example-plugin_initialize(): passed arguments:\n");
    for (i=0; i < argc; i++)
    {
        lcas_log_debug(2,"lcas_plugin_example-plugin_initialize(): arg %d is %s\n",
             i,argv[i]);
    }

    return LCAS_MOD_SUCCESS;
}

/******************************************************************************
Function:   plugin_confirm_authorization
Description:
    Ask for authorization by passing RSL(JDL) and user credential
Parameters:
    request:   RSL request
    lcas_cred: user credential
Returns:
    LCAS_MOD_SUCCESS: authorization succeeded
    LCAS_MOD_FAIL   : authorization failed
```

```
    LCAS_MOD_NOFILE : db file not found (LCAS will deny authorization)
******************************************************************************/
/*!
    \fn plugin_confirm_authorization(
        lcas_request_t request,
        lcas_cred_id_t lcas_cred
        )
    \brief Ask the plugin authorization module for authorization.

    Ask for authorization by passing the RSL (later JDL) and the user credential.
    The user credential will contain information on the role the user wants to have.
    In the RSL (JDL) the user might specify the resources he wants to use.
    The authorization decision has to be made using this information.
    The LCAS provides no library for parsing the RSL (JDL).

    \param request   LCAS (RSL) request
    \param lcas_cred LCAS credential
    \retval LCAS_MOD_SUCCESS authorization succeeded
    \retval LCAS_MOD_FAIL    authorization failed
    \retval LCAS_MOD_NOFILE  private plugin database could not be found (LCAS will deny authorization)
*/
int plugin_confirm_authorization(
        lcas_request_t request,
        lcas_cred_id_t lcas_cred
)
{
    char *  username = NULL;

    lcas_log_debug(1,"\tlcas_plugin_example-plugin_confirm_authorization():\n");

    /*
     * check credential and get the globus name
     */
    if ( (username = lcas_get_dn(lcas_cred)) == NULL)
    {
        lcas_log(0, "lcas.mod-lcas_get_fabric_authorization() error: user DN empty\n");
        goto fail_example;
    }

    if (username != NULL)
    {
        lcas_log_debug(0,
            "\tlcas_plugin_example-plugin_confirm_authorization(): OK, what the heck, I'll authorize Mr/Mrs %s\n",
                username);
    }
    else
    {
        lcas_log_debug(0,
            "\tlcas_plugin_example-plugin_confirm_authorization(): Authorization failure: invalid username (%s)\n",username);
        goto fail_example;
    }

    /* succes */
    return LCAS_MOD_SUCCESS;

 fail_example:
    return LCAS_MOD_FAIL;
}

/******************************************************************************
Function:   plugin_terminate
Description:
    Terminate plugin
Parameters:

Returns:
    LCAS_MOD_SUCCESS : succes
    LCAS_MOD_FAIL    : failure
******************************************************************************/
/*!
    \fn plugin_terminate()
    \brief Whatever is needed to terminate the plugin module goes in here.
    \retval LCAS_MOD_SUCCESS success
    \retval LCAS_MOD_FAIL    failure (will result in an authorization failure)
*/
int plugin_terminate()
{
    lcas_log_debug(1,"lcas_plugin_example-plugin_terminate(): terminating\n");

    return LCAS_MOD_SUCCESS;
}
/*\@}*/
```

```
/******************************************************************************
CVS Information:
    $Source: /cvs/fabric_mgt/gridification/lcas/modules/example/lcas_plugin_example.c,v $
    $Date: 2003/08/08 16:25:00 $
    $Revision: 1.1 $
    $Author: martijn $
******************************************************************************/
```

The new plugin can be tested without having a functioning edg-gatekeeper by running the program `lcas-test` in the `src` directory of the LCAS cvs repository, which is basically a copy of the part of the edg-gatekeeper that contacts the LCAS.

5 User guide

Empty.