edg-lcmaps Reference Manual

Generated by Doxygen 1.2.8.1

Wed Jul 16 16:33:34 2003

Contents

Chapter 1

LCMAPS - Local Credential MAPping Service

1.1 Introduction

This document describes the LCMAPS API and the LCMAPS plugins. Please check the links above.

1.2 the LCMAPS Interfaces

1. The interface to the LCMAPS credential mapping framework is described in Interface to LCMAPS (library)
2. The LCMAPS plugins should use the LCMAPS API described in The API to be used by the LCMAPS plugins
3. The interface that the plugins should provide to the LCMAPS framework is described in The interface to the LCMAPS plugins

1.3 The LCMAPS plugins

A description of the LCMAPS plugins can be found here ...

... the basic plugins:

1. posix enforcement plugin
2. ldap enforcement plugin
3. localaccount plugin

4. poolaccount plugin

... the voms-aware plugins:

1. voms plugin
2. voms poolaccount plugin
3. voms localgroup plugin
4. voms poolgroup plugin

Chapter 2

edg-lcmaps Module Index

2.1 edg-lcmaps Modules

Here is a list of all modules:

Chapter 3

edg-lcmaps Data Structure Index

3.1 edg-lcmaps Data Structures

Here are the data structures with brief descriptions:

Chapter 4

edg-lcmaps File Index

4.1 edg-lcmaps File List

Here is a list of all documented files with brief descriptions:

Chapter 5

edg-lcmaps Page Index

5.1 edg-lcmaps Related Pages

Here is a list of all related documentation pages:

Chapter 6

edg-lcmaps Module Documentation

6.1 Interface to LCMAPS (library)

The API is available by including the header lcmaps.h.

Files

- file lcmaps.h

  API of the LCMAPS library.

6.1.1 Detailed Description

The API is available by including the header lcmaps.h.

6.2 The API to be used by the LCMAPS plugins

The API is available by including the header lcmaps_modules.h.

Files

- file lcmaps_arguments.h

  Public header file to be used by plugins.

- file lcmaps_cred_data.h

  Public header file to be used by plugins.

- file lcmaps_defines.h

  Public header file with common definitions for the LCMAPS (authorization modules).

- file lcmaps_log.h

  Logging API for the LCMAPS plugins and LCMAPS itself.

- file lcmaps_modules.h

  The LCMAPS authorization plugins/modules should "include" this file.

- file lcmaps_types.h

  Public header file with typedefs for LCMAPS.

- file lcmaps_utils.h

  API for the utilities for the LCMAPS.

- file lcmaps_vo_data.h

  LCMAPS module for creating and accessing VO data structures.

6.2.1 Detailed Description

The API is available by including the header lcmaps_modules.h.

6.3 The interface to the LCMAPS plugins

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

1. plugin_initialize()
2. plugin_run()
3. plugin_terminate()
4. plugin_introspect()

Chapter 7

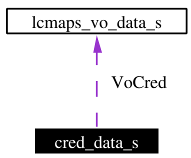
edg-lcmaps Class Documentation

7.1 cred_data_s Struct Reference

structure that contains the gathered (local) credentials en VOMS info.

`#include <lcmaps_cred_data.h>`

Collaboration diagram for cred_data_s:



Data Fields

- char* dn
- uid_t* uid
- gid_t* priGid
- gid_t* secGid
- lcmaps_vo_data_t* VoCred
- char** VoCredString
- int cntUid
- int cntPriGid
- int cntSecGid
- int cntVoCred
- int cntVoCredString

7.1.1 Detailed Description

structure that contains the gathered (local) credentials en VOMS info.

Definition at line 55 of file lcmaps_cred_data.h.

7.1.2 Field Documentation

7.1.2.1 lcmaps_vo_data_t * cred_data_s::VoCred

list of VO data structures

Definition at line 61 of file lcmaps_cred_data.h.

7.1.2.2 char ** cred_data_s::VoCredString

list of VO data strings

Definition at line 62 of file lcmaps_cred_data.h.

7.1.2.3 int cred_data_s::cntPriGid

number of primary groupIDs (in principle only one)

Definition at line 64 of file lcmaps_cred_data.h.

7.1.2.4 int cred_data_s::cntSecGid

number of secondary groupIDs (could be any number)

Definition at line 65 of file lcmaps_cred_data.h.

7.1.2.5 int cred_data_s::cntUid

number of userIDs

Definition at line 63 of file lcmaps_cred_data.h.

7.1.2.6 int cred_data_s::cntVoCred

number of VO data structures

Definition at line 66 of file lcmaps_cred_data.h.

7.1.2.7 int cred_data_s::cntVoCredString

number of VO data strings

Definition at line 67 of file lcmaps_cred_data.h.

7.1.2.8 char ∗ cred_data_s::dn

user globus DN

Definition at line 57 of file lcmaps_cred_data.h.

7.1.2.9 gid_t ∗ cred_data_s::priGid

list of primary groupIDs

Definition at line 59 of file lcmaps_cred_data.h.

7.1.2.10 gid_t ∗ cred_data_s::secGid

list of secondary groupIDs

Definition at line 60 of file lcmaps_cred_data.h.

7.1.2.11 uid_t ∗ cred_data_s::uid

list of userIDs

Definition at line 58 of file lcmaps_cred_data.h.

The documentation for this struct was generated from the following file:

- lcmaps_cred_data.h

7.2 lcmaps_argument_s Struct Reference

structure representing an LCMAPS plugin run argument.

```
#include <lcmaps_arguments.h>
```

Data Fields

- char* argName
- char* argType
- int argInOut
- void* value

7.2.1 Detailed Description

structure representing an LCMAPS plugin run argument.

Definition at line 42 of file lcmaps_arguments.h.

7.2.2 Field Documentation

7.2.2.1 int lcmaps_argument_s::argInOut

input or output argument (0 = false = Input / 1 = true = Out)

Definition at line 46 of file lcmaps_arguments.h.

7.2.2.2 char * lcmaps_argument_s::argName

name of argument

Definition at line 44 of file lcmaps_arguments.h.

7.2.2.3 char * lcmaps_argument_s::argType

type of the argument

Definition at line 45 of file lcmaps_arguments.h.

7.2.2.4 void * lcmaps_argument_s::value

value of argument

Definition at line 47 of file lcmaps_arguments.h.

The documentation for this struct was generated from the following file:

- lcmaps_arguments.h

7.3 lcmaps_cred_id_s Struct Reference

structure representing an LCMAPS credential.

`#include <lcmaps_types.h>`

Data Fields

- gss_cred_id_t cred
- char* dn

7.3.1 Detailed Description

structure representing an LCMAPS credential.

Definition at line 47 of file lcmaps_types.h.

7.3.2 Field Documentation

7.3.2.1 gss_cred_id_t lcmaps_cred_id_s::cred

the original gss (globus) credential

Definition at line 49 of file lcmaps_types.h.

7.3.2.2 char * lcmaps_cred_id_s::dn

the user distinguished name (DN)

Definition at line 50 of file lcmaps_types.h.

The documentation for this struct was generated from the following file:

- lcmaps_types.h

7.4 lcmaps_db_entry_s Struct Reference

LCMAPS data base element structure.

`#include <lcmaps_db_read.h>`

Collaboration diagram for lcmaps_db_entry_s:

Data Fields

- char pluginname [LCMAPS_MAXPATHLEN+1]
- char pluginargs [LCMAPS_MAXARGSTRING+1]
- struct lcmaps_db_entry_s* next

7.4.1 Detailed Description

LCMAPS data base element structure.

For internal use only.

Definition at line 42 of file lcmaps_db_read.h.

7.4.2 Field Documentation

7.4.2.1 struct lcmaps_db_entry_s * lcmaps_db_entry_s::next

handle to next db element

Definition at line 46 of file lcmaps_db_read.h.

7.4.2.2 char lcmaps_db_entry_s::pluginargs

Argument list to be passed to authorization plugin/module

Definition at line 45 of file lcmaps_db_read.h.

7.4.2.3 char lcmaps_db_entry_s::pluginname

Name of authorization plugin/module

Definition at line 44 of file lcmaps_db_read.h.

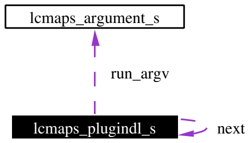
The documentation for this struct was generated from the following file:

- lcmaps_db_read.h

7.5 lcmaps_plugindl_s Struct Reference

the lcmaps plugin module structure.

Collaboration diagram for lcmaps_plugindl_s:



Data Fields

- void* handle
- lcmaps_proc_t procs [MAXPROCS]
- char pluginname [LCMAPS_MAXPATHLEN+1]
- char pluginargs [LCMAPS_MAXARGSTRING+1]
- int init_argc
- char* init_argv [LCMAPS_MAXARGS]
- int run_argc
- lcmaps_argument_t* run_argv
- struct lcmaps_plugindl_s* next

7.5.1 Detailed Description

the lcmaps plugin module structure.

For internal use only.

Definition at line 102 of file lcmaps_pluginmanager.c.

7.5.2 Field Documentation

7.5.2.1 void * lcmaps_plugindl_s::handle

dlopen handle to plugin module

Definition at line 104 of file lcmaps_pluginmanager.c.

7.5.2.2 int lcmaps_plugindl_s::init_argc

number of arguments for the initialization function

Definition at line 108 of file lcmaps_pluginmanager.c.

7.5.2.3 char * lcmaps_plugindl_s::init_argv

list of arguments for the initialization function

Definition at line 109 of file lcmaps_pluginmanager.c.

7.5.2.4 struct lcmaps_plugindl_s * lcmaps_plugindl_s::next

pointer to the next plugin in the plugin list

Definition at line 112 of file lcmaps_pluginmanager.c.

7.5.2.5 char lcmaps_plugindl_s::pluginargs

argument string

Definition at line 107 of file lcmaps_pluginmanager.c.

7.5.2.6 char lcmaps_plugindl_s::pluginname

name of plugin

Definition at line 106 of file lcmaps_pluginmanager.c.

7.5.2.7 lcmaps_proc_t lcmaps_plugindl_s::procs

list of handles to interface functions of plugin

Definition at line 105 of file lcmaps_pluginmanager.c.

7.5.2.8 int lcmaps_plugindl_s::run_argc

number of arguments for the plugin run function (get credentials)

Definition at line 110 of file lcmaps_pluginmanager.c.

7.5.2.9 lcmaps_argument_t * lcmaps_plugindl_s::run_argv

list of arguments for the plugin run function (get credentials)

Definition at line 111 of file lcmaps_pluginmanager.c.

The documentation for this struct was generated from the following file:

- lcmaps_pluginmanager.c

7.6 lcmaps_vo_data_s Struct Reference

structure that contains the VO information found in the user's gss credential.

```
#include <lcmaps_vo_data.h>
```

Data Fields

- char* vo
- char* group
- char* subgroup
- char* role
- char* capability

7.6.1 Detailed Description

structure that contains the VO information found in the user's gss credential.

Definition at line 46 of file lcmaps_vo_data.h.

7.6.2 Field Documentation

7.6.2.1 char * lcmaps_vo_data_s::capability

the user's capability

Definition at line 52 of file lcmaps_vo_data.h.

7.6.2.2 char * lcmaps_vo_data_s::group

group within the VO

Definition at line 49 of file lcmaps_vo_data.h.

7.6.2.3 char * lcmaps_vo_data_s::role

the user's role

Definition at line 51 of file lcmaps_vo_data.h.

7.6.2.4 char * lcmaps_vo_data_s::subgroup

subgroup name

Definition at line 50 of file lcmaps_vo_data.h.

7.6.2.5 char * lcmaps_vo_data_s::vo

name of the VO to which the user belongs

Definition at line 48 of file lcmaps_vo_data.h.

The documentation for this struct was generated from the following file:

- lcmaps_vo_data.h

7.7 plugin_s Struct Reference

Structure holds a plugin name and its arguments, as well as the line number the plugin is first mentioned.

#include <pdl.h>

Collaboration diagram for plugin_s:

Data Fields

- char∗ name

  Plugin name.

- char∗ args

  Arguments of the plugin.

- unsigned int lineno

  Line number where the plugin is first seen in the configuration file.

- struct plugin_s∗ next

  Next plugin, or 0 if there are no-more plugins.

7.7.1 Detailed Description

Structure holds a plugin name and its arguments, as well as the line number the plugin is first mentioned.

Definition at line 94 of file pdl.h.

The documentation for this struct was generated from the following file:

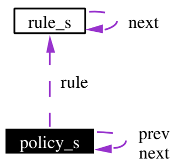
- pdl.h

7.8 policy_s Struct Reference

Keeping track of found policies.

`#include <pdl_policy.h>`

Collaboration diagram for policy_s:



Data Fields

- const char* name
  Name of the policy.
- rule_t* rule
  Pointer to the first rule of the policy.
- unsigned int lineno
  Line number where the polict was found.
- struct policy_s* next
  Next policy, or 0 if none.
- struct policy_s* prev
  Previous policy, or 0 if none.

7.8.1 Detailed Description

Keeping track of found policies.

Definition at line 41 of file pdl_policy.h.

The documentation for this struct was generated from the following file:

- pdl_policy.h

7.9 record_s Struct Reference

Structure is used to keep track of strings and the line they appear on.

```
#include <pdl.h>
```

Data Fields

- char∗ string

  Hold the symbol that lex has found.

- int lineno

  Hold the line number the symbol has been found.

7.9.1 Detailed Description

Structure is used to keep track of strings and the line they appear on.

When lex finds a match, this structure is used to keep track of the relevant information. The matchig string as well as the line number are saved. The line number can be used for later references when an error related to the symbol has occured. This allows for easier debugging of the configuration file.

Definition at line 83 of file pdl.h.

The documentation for this struct was generated from the following file:

- pdl.h

7.10 rule_s Struct Reference

Structure keeps track of the state and the true/false braches.

`#include <pdl_rule.h>`

Collaboration diagram for rule_s:

Data Fields

- const char* state

  Name of the state.

- const char* true_branch

  Name of the true_branch, or 0 if none.

- const char* false_branch

  Name of the false_branch, or 0 if none.

- unsigned int lineno

  Line number where rule appeared.

- struct rule_s* next

  Next rule, or 0 if none.

7.10.1 Detailed Description

Structure keeps track of the state and the true/false braches.

Definition at line 40 of file pdl_rule.h.

The documentation for this struct was generated from the following file:

- pdl_rule.h

7.11 var_s Struct Reference

Structure keeps track of the variables, their value and the line number they are defined on.

```
#include <pdl_variable.h>
```

Collaboration diagram for var_s:

Data Fields

- const char* name

  Name of the variable.

- const char* value

  Value of the variable.

- unsigned int lineno

  Line number the variable appears on.

- struct var_s* next

  Next variable, or 0 if none.

7.11.1 Detailed Description

Structure keeps track of the variables, their value and the line number they are defined on.

Definition at line 44 of file pdl_variable.h.

The documentation for this struct was generated from the following file:

- pdl_variable.h

Chapter 8

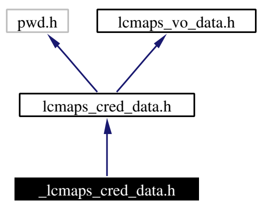
edg-lcmaps File Documentation

8.1 _lcmaps_cred_data.h File Reference

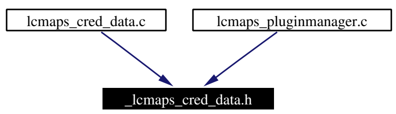
Internal header file of LCMAPS credential data.

`#include "lcmaps_cred_data.h"`

Include dependency graph for _lcmaps_cred_data.h:



This graph shows which files directly or indirectly include this file:



Functions

- int cleanCredentialData ()

    Clean the credData structure.

8.1.1 Detailed Description

Internal header file of LCMAPS credential data.

Author:
Oscar Koeroo and Martijn Steenbakkers for the EU DataGrid.

For internal use only.

Definition in file _lcmaps_cred_data.h.

8.1.2 Function Documentation

8.1.2.1 int cleanCredentialData ()

Clean the credData structure.

Returns:
0
For internal use only.

Definition at line 237 of file lcmaps_cred_data.c.

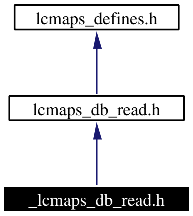
Referenced by stopPluginManager().

8.2 _lcmaps_db_read.h File Reference

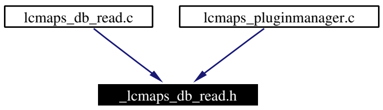
Internal header file of LCMAPS database reader.

`#include "lcmaps_db_read.h"`

Include dependency graph for _lcmaps_db_read.h:



This graph shows which files directly or indirectly include this file:



Functions

- lcmaps_db_entry_t* lcmaps_db_fill_entry (lcmaps_db_entry_t **plcmaps_db, lcmaps_db_entry_t *db_-entry)

  Add a database entry to a list.

- lcmaps_db_entry_t** lcmaps_db_read (char *lcmaps_db_fname)

  Read database from file.

- int lcmaps_db_clean_list (lcmaps_db_entry_t **list)

  Clean/remove the database list.

- int lcmaps_db_clean ()

  Clean/remove the database structure.

8.2.1 Detailed Description

Internal header file of LCMAPS database reader.

Author:

Martijn Steenbakkers for the EU DataGrid.

This header contains the declarations of the LCMAPS database reader functions and typedefs.

For internal use only.

Definition in file _lcmaps_db_read.h.

8.2.2 Function Documentation

8.2.2.1 int lcmaps_db_clean ()

Clean/remove the database structure.

Return values:

0 succes

1 failure

For internal use only.

Definition at line 588 of file lcmaps_db_read.c.

Referenced by startPluginManager().

8.2.2.2 int lcmaps_db_clean_list (lcmaps_db_entry_t ** *list*)

Clean/remove the database list.

Parameters:

list pointer to the database list

Return values:

0 succes.

1 failure.

For internal use only.

Definition at line 558 of file lcmaps_db_read.c.

8.2.2.3 lcmaps_db_entry_t * lcmaps_db_fill_entry (lcmaps_db_entry_t ** *list*, lcmaps_db_entry_t * *entry*)

Add a database entry to a list.

Parameters:

list database list (array of database entry pointers)

entry the database entry to be added

Returns:

a pointer to the newly created database entry in the list or NULL (error)
For internal use only.

Definition at line 198 of file lcmaps_db_read.c.

8.2.2.4 lcmaps_db_entry_t ** lcmaps_db_read (char * *lcmaps_db_fname*)

Read database from file.

Parameters:

lcmaps_db_fname database file.

Returns:

a pointer to the database list
For internal use only.

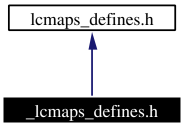
Definition at line 89 of file lcmaps_db_read.c.

8.3 _lcmaps_defines.h File Reference

Internal header file with some common defines for LCMAPS.

`#include "lcmaps_defines.h"`

Include dependency graph for _lcmaps_defines.h:



Defines

- #define MAXPATHLEN 100
- #define MAXARGSTRING 500
- #define MAXARGS 51

8.3.1 Detailed Description

Internal header file with some common defines for LCMAPS.

Author:
Martijn Steenbakkers for the EU DataGrid.
For internal use only.

Definition in file _lcmaps_defines.h.

8.3.2 Define Documentation

8.3.2.1 #define MAXARGS 51

maximum number of arguments (+1) to be passed to LCAS authorization plugins/modules.

For internal use only.

Definition at line 33 of file _lcmaps_defines.h.

8.3.2.2 #define MAXARGSTRING 500

maximum length of the plugin argument string as specified in the LCAS database.

For internal use only.

Definition at line 31 of file _lcmaps_defines.h.

8.3.2.3 #define MAXPATHLEN 100

maximum path lengths of files, used in plugin and database structures.

For internal use only.

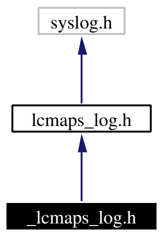
Definition at line 29 of file _lcmaps_defines.h.

8.4 _lcmaps_log.h File Reference

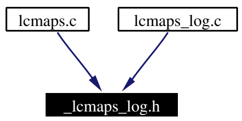
Internal header file for LCMAPS logging routines.

```
#include "lcmaps_log.h"
```

Include dependency graph for _lcmaps_log.h:



This graph shows which files directly or indirectly include this file:



Defines

- #define MAX_LOG_BUFFER_SIZE 2048
- #define DO_USRLOG ((unsigned short)0x0001)
- #define DO_SYSLOG ((unsigned short)0x0002)

Functions

- int lcmaps_log_open (char ∗path, FILE ∗fp, unsigned short logtype)

    Start logging.

- int lcmaps_log_close ()

    Stop logging.

8.4.1 Detailed Description

Internal header file for LCMAPS logging routines.

Author:

Martijn Steenbakkers for the EU DataGrid.
For internal use only.

Definition in file _lcmaps_log.h.

8.4.2 Define Documentation

8.4.2.1 #define DO_SYSLOG ((unsigned short)0x0002)

flag to indicate that syslogging has to be done

For internal use only.

Definition at line 34 of file _lcmaps_log.h.

8.4.2.2 #define DO_USRLOG ((unsigned short)0x0001)

flag to indicate that user logging has to be done

For internal use only.

Definition at line 32 of file _lcmaps_log.h.

8.4.2.3 #define MAX_LOG_BUFFER_SIZE 2048

Maximum logging buffer size, length of log may not exceed this number

For internal use only.

Definition at line 29 of file _lcmaps_log.h.

8.4.3 Function Documentation

8.4.3.1 int lcmaps_log_close ()

Stop logging.

For internal use only.

Definition at line 247 of file lcmaps_log.c.

8.4.3.2 int lcmaps_log_open (char ∗ *path*, FILE ∗ *fp*, unsigned short *logtype*)

Start logging.

This function should only be used by the LCMAPS itself.

Parameters:

- ***path*** path of logfile.
- ***fp*** file pointer to already opened file (or NULL)
- ***logtype*** DO_USRLOG, DO_SYSLOG

Return values:

0 succes.

1 failure.

For internal use only.

Definition at line 74 of file lcmaps_log.c.

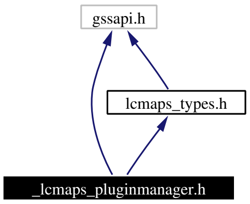
8.5 _lcmaps_pluginmanager.h File Reference

API of the PluginManager.

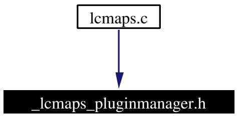
`#include <gssapi.h>`

`#include "lcmaps_types.h"`

Include dependency graph for _lcmaps_pluginmanager.h:



This graph shows which files directly or indirectly include this file:



Functions

- int startPluginManager ()

  start the PluginManager.

- int stopPluginManager ()

  Terminate the PluginManager module.

- int runPluginManager (lcmaps_request_t request, lcmaps_cred_id_t lcmaps_cred)

  Run the PluginManager.

- int runPlugin (const char *pluginname)

  Run a plugin.

8.5.1 Detailed Description

API of the PluginManager.

Author:

Martijn Steenbakkers for the EU DataGrid.

This header contains the declarations of the LCMAPS library functions:

1. startPluginManager(): start the PluginManager –> load plugins, start evaluation manager
2. runPluginManager(): run the PluginManager –> run evaluation manager –> run plugins
3. stopPluginManager(): stop the PluginManager
4. runPlugin(): run the specified plugin. (used by Evaluation Manager)

Definition in file _lcmaps_pluginmanager.h.

8.5.2 Function Documentation

8.5.2.1 int runPlugin (const char ∗ *pluginname*)

Run a plugin.

Run a plugin for the Evaluation Manager the result (succes or not will be passed to the Evaluation Manager)

Parameters:

pluginname the name of the plugin module

Return values:

0 plugin run succeeded

1 plugin run failed

Definition at line 960 of file lcmaps_pluginmanager.c.

8.5.2.2 int runPluginManager (lcmaps_request_t *request*, lcmaps_cred_id_t *lcmaps_cred*)

Run the PluginManager.

This function runs the PluginManager for user mapping. Contact Evaluation Manager –> runs plugins

Parameters:

request RSL request (job request)

lcmaps_cred user credential

Return values:

0 user mapping succeeded

1 user mapping failed

Definition at line 849 of file lcmaps_pluginmanager.c.

8.5.2.3 int startPluginManager ()

start the PluginManager.

start the PluginManager –> load plugins, start evaluation manager

Return values:

0 succes

1 failure

Definition at line 154 of file lcmaps_pluginmanager.c.

Referenced by lcmaps_init().

8.5.2.4 int stopPluginManager ()

Terminate the PluginManager module.

stop the PluginManager –> terminate plugins, clean plugin list, (stop evaluation manager)

Return values:

0 succes

1 failure

Definition at line 1017 of file lcmaps_pluginmanager.c.

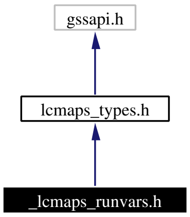
Referenced by lcmaps_term().

8.6 _lcmaps_runvars.h File Reference

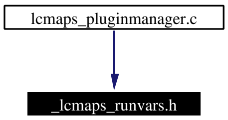
API of runvars structure.

`#include "lcmaps_types.h"`

Include dependency graph for _lcmaps_runvars.h:



This graph shows which files directly or indirectly include this file:



Functions

- int lcmaps_extractRunVars (lcmaps_request_t request, lcmaps_cred_id_t lcmaps_cred)

  extract the variables from user credential that can be used by the plugins.

- void* lcmaps_getRunVars (char *argName, char *argType)

  returns a void pointer to the requested value.

- int lcmaps_setRunVars (char *argName, char *argType, void *value)

  fill the runvars_list with a value for argName and argType.

8.6.1 Detailed Description

API of runvars structure.

Author:

Martijn Steenbakkers for the EU DataGrid.

This module takes the data that are presented to LCMAPS (the global credential and Job request) and extracts the variables that will be used by the plugins from it and stores them into a list. The interface to the LCMAPS module is composed of:

1. lcmaps_extractRunVars(): takes the global credential and Job request and extracts run variables from them

2. lcmaps_setRunVars(): adds run variables to a list

3. lcmaps_getRunVars(): gets run variables from list

Definition in file _lcmaps_runvars.h.

8.6.2 Function Documentation

8.6.2.1 int lcmaps_extractRunVars (lcmaps_request_t *request*, lcmaps_cred_id_t *lcmaps_cred*)

extract the variables from user credential that can be used by the plugins.

This function takes the user credential and job request (in RSL) and extracts the information which is published in the runvars_list. These variables can be accessed by the plugins.

Parameters:

request the job request (RSL)

lcmaps_cred the credential presented by the user

Return values:

0 succes.

1 failure.

For internal use only.

Definition at line 96 of file lcmaps_runvars.c.

8.6.2.2 void ∗ lcmaps_getRunVars (char ∗ *argName*, char ∗ *argType*)

returns a void pointer to the requested value.

This function returns a void pointer to the requested variable with name argName and type argType in the runvars_list. Internally it uses lcmaps_getArgValue().

Parameters:

argName name of the variable

argType type of the variable

Returns:

void pointer to the value or NULL
For internal use only.

Definition at line 191 of file lcmaps_runvars.c.

8.6.2.3 int lcmaps_setRunVars (char * *argName*, char * *argType*, void * *value*)

fill the runvars_list with a value for argName and argType.

This function fills the (internal) runvars_list with the value for the variable with name argName and type argType. Internally lcmaps_setArgValue() is used.

Parameters:

argName name of the runvars variable

argType type of the runvars variable

values void pointer to the value

Return values:

0 succes.

-1 failure.

For internal use only.

Definition at line 232 of file lcmaps_runvars.c.

8.7 _lcmaps_utils.h File Reference

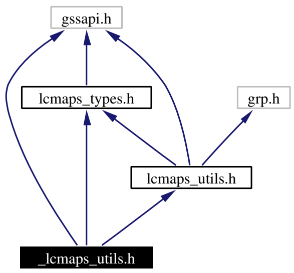
Internal header for the LCMAPS utilities.

`#include <gssapi.h>`

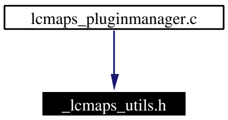
`#include "lcmaps_types.h"`

`#include "lcmaps_utils.h"`

Include dependency graph for _lcmaps_utils.h:



This graph shows which files directly or indirectly include this file:



CREDENTIAL FUNCTIONS

- int lcmaps_fill_cred (char ∗dn, gss_cred_id_t cred, lcmaps_cred_id_t ∗lcmaps_credential)

  Fill cedential from distinghuished name and globus credential.

- int lcmaps_release_cred (lcmaps_cred_id_t ∗lcmaps_credential)

  Release the LCMAPS credential.

OTHER FUNCTIONS

- int lcmaps_tokenize (const char ∗command, char ∗∗args, int ∗n, char ∗sep)

  Break the argument string up into tokens.

8.7.1 Detailed Description

Internal header for the LCMAPS utilities.

Author:
Martijn Steenbakkers for the EU DataGrid.

This header contains the declarations of the LCMAPS utility functions:

1. lcmaps_fill_cred():
2. lcmaps_release_cred():
3. lcmaps_tokenize():
   For internal use only.

Definition in file _lcmaps_utils.h.

8.7.2 Function Documentation

8.7.2.1 int lcmaps_fill_cred (char ∗ *dn*, gss_cred_id_t *cred*, lcmaps_cred_id_t ∗ *plcmaps_cred*)

Fill cedential from distinghuished name and globus credential.

The LCMAPS credential only differs from the GLOBUS credential by the extra entry for the dn. This allows (temporarily) the passed delegated GLOBUS credential to be empty.

Parameters:
dn distinguished name
cred GLOBUS credential
lcmaps_cred pointer to LCMAPS credential to be filled.

Return values:
0 succes.
1 failure.
For internal use only.

Definition at line 74 of file lcmaps_utils.c.

8.7.2.2 int lcmaps_release_cred (lcmaps_cred_id_t ∗ *plcmaps_cred*)

Release the LCMAPS credential.

Parameters:
lcmaps_cred pointer to LCMAPS credential to be released

Return values:
0 succes.
1 failure.
For internal use only.

Definition at line 115 of file lcmaps_utils.c.

8.7.2.3 int lcmaps_tokenize (const char * *command*, char ** *args*, int * *n*, char * *sep*)

Break the argument string up into tokens.

Breakup the command in to arguments, pointing the args array at the tokens. Replace white space at the end of each token with a null. A token maybe in quotes. (Copied (and modified) from GLOBUS gatekeeper.c)

Parameters:

command the command line to be parsed

args pointer to an array of pointers to be filled

n size of the array, on input, and set to size used on output

sep string of separating characters

Return values:

0 succes

-1 malloc error

-2 too many args

-3 quote not matched

For internal use only.

Definition at line 455 of file lcmaps_utils.c.

8.8 evaluationmanager.c File Reference

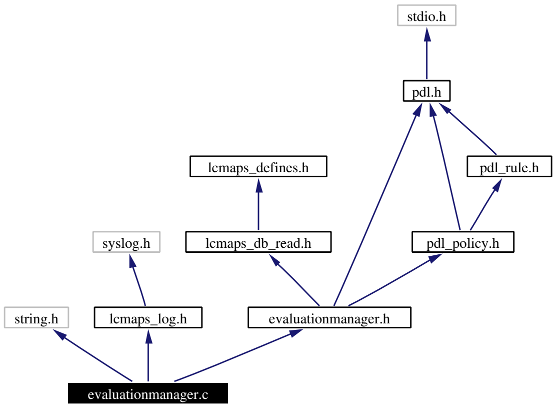
Implementation of the evaluation manager interface.

`#include <string.h>`

`#include "lcmaps_log.h"`

`#include "evaluationmanager.h"`

Include dependency graph for evaluationmanager.c:



Functions

- int free_lcmaps_db_entry ()
- int startEvaluationManager (const char ∗name)
- int getPluginNameAndArgs (lcmaps_db_entry_t ∗∗plugins)
- int runEvaluationManager (void)
- int stopEvaluationManager (void)

Variables

- lcmaps_db_entry_t∗ global_plugin_list = NULL

8.8.1 Detailed Description

Implementation of the evaluation manager interface.

Besides the implementation of the interface of the evaluation manager some additional functions are implemented here. Please note that these are **not** part of the interface and hence should not be used. Look in evaluationmanager.h for the functions that can be called by external sources.

Author:
G.M. Venekamp (venekamp@nikhef.nl)

Version:

Revision:
1.14

Date:

Date:
2003/07/16 09:30:57

Definition in file evaluationmanager.c.

8.8.2 Function Documentation

8.8.2.1 int free_lcmaps_db_entry ()

During the getPluginsAndArgs() call, a list structure is created. This structure is never cleaned automatically, nor can it be. When it is necessay and safe to free the resources, call this function

Return values:

0 when the call is successful,

1 otherwise.

Definition at line 240 of file evaluationmanager.c.

Referenced by stopEvaluationManager().

8.8.2.2 int getPluginNameAndArgs (lcmaps_db_entry_t ** *plugin*)

Get a list of plugins and their arguments based on the configuration file. The memory that is allocted is freed during the stopEvaluationManager() call.

Parameters:

plugins Pointer to be intialized with the first entry of the plugin list.

Return values:

0 when the call is successful,

1 otherwise.

Definition at line 102 of file evaluationmanager.c.

8.8.2.3 int runEvaluationManager (void)

Run the evaluation manager. The evaluation manager has to be initialized by calling statrEvaluation Manager first.

Return values:

0 when the call is successful,

1 otherwise.

Definition at line 185 of file evaluationmanager.c.

Referenced by runPluginManager().

8.8.2.4 int startEvaluationManager (const char ∗ *name*)

Start the evaluation manager.

Parameters:

name Name of the configure script.

Return values:

0 when the call is successful,

1 otherwise.

Definition at line 62 of file evaluationmanager.c.

8.8.2.5 int stopEvaluationManager (void)

Stop the evaluation manager after is has run successfully. Strictly speaking, the evalauation manager needs no stopping. This call is a good point to clean up the resources used by the evaluation manager.

Return values:

0 when the call is successful,

1 otherwise.

Definition at line 219 of file evaluationmanager.c.

Referenced by startEvaluationManager(), and stopPluginManager().

8.8.3 Variable Documentation

8.8.3.1 lcmaps_db_entry_t ∗ global_plugin_list = NULL `[static]`

When the getPluginNameAndArgs() function has been called, the global_plugin_list variable get initialized with the first element of the list. This variable is later used to free the resources held by the list. In addition, multiple calls to getPluginNameAndArgs() result in returning the value of this pointer.

Definition at line 49 of file evaluationmanager.c.

8.9 evaluationmanager.h File Reference

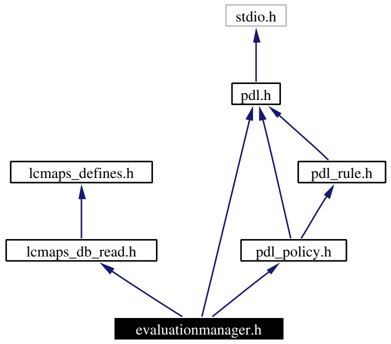
Evaluation Manager interface definition.

`#include "lcmaps_db_read.h"`

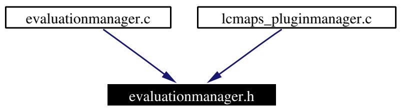
`#include "pdl.h"`

`#include "pdl_policy.h"`

Include dependency graph for evaluationmanager.h:



This graph shows which files directly or indirectly include this file:



Functions

- int startEvaluationManager (const char ∗name)
- int getPluginNameAndArgs (lcmaps_db_entry_t ∗∗plugin)
- int runEvaluationManager (void)
- int stopEvaluationManager (void)

8.9.1 Detailed Description

Evaluation Manager interface definition.

The function listed in here are accessible to anyone. This is the way to communicate with the evaluation manager. The evaluation manager deligates the becessary work to the Policy Language Description module (PDL).

Author:

G.M. Venekamp (venekamp@nikhef.nl)

Version:

Revision:
1.6

Date:

Date:
2003/05/26 10:50:26

Definition in file evaluationmanager.h.

8.9.2 Function Documentation

8.9.2.1 int getPluginNameAndArgs (lcmaps_db_entry_t ** *plugin*)

Get a list of plugins and their arguments based on the configuration file. The memory that is allocted is freed during the stopEvaluationManager() call.

Parameters:
plugins Pointer to be intialized with the first entry of the plugin list.

Return values:
0 when the call is successful,
1 otherwise.

Definition at line 102 of file evaluationmanager.c.

Referenced by startPluginManager().

8.9.2.2 int runEvaluationManager (void)

Run the evaluation manager. The evaluation manager has to be initialized by calling statrEvaluation Manager first.

Return values:
0 when the call is successful,
1 otherwise.

Definition at line 185 of file evaluationmanager.c.

8.9.2.3 int startEvaluationManager (const char * *name*)

Start the evaluation manager.

Parameters:
name Name of the configure script.

Return values:
0 when the call is successful,
1 otherwise.

Definition at line 62 of file evaluationmanager.c.

Referenced by startPluginManager().

8.9.2.4 int stopEvaluationManager (void)

Stop the evaluation manager after is has run successfully. Strictly speaking, the evalauation manager needs no stopping. This call is a good point to clean up the resources used by the evaluation manager.

Return values:
0 when the call is successful,
1 otherwise.

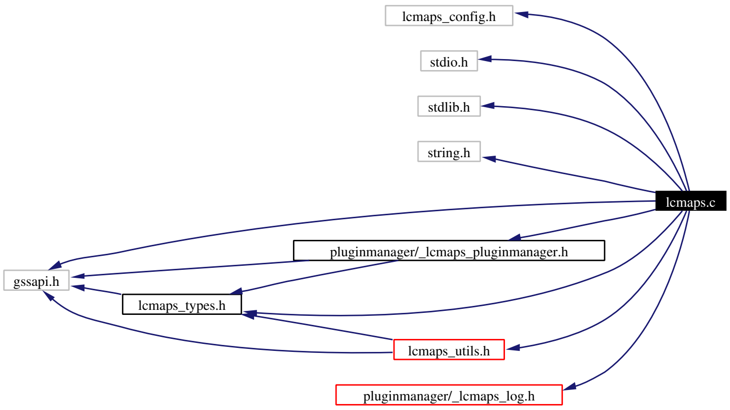
Definition at line 219 of file evaluationmanager.c.

8.10 lcmaps.c File Reference

the LCMAPS module - the local credential mapping service.

```
#include "lcmaps_config.h"
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <gssapi.h>
#include "pluginmanager/_lcmaps_pluginmanager.h"
#include "pluginmanager/_lcmaps_log.h"
#include "lcmaps_types.h"
#include "lcmaps_utils.h"
```

Include dependency graph for lcmaps.c:



Variables

- lcmaps_cred_id_t lcmaps_cred
- int lcmaps_initialized = 0

8.10.1 Detailed Description

the LCMAPS module - the local credential mapping service.

Author:

Martijn Steenbakkers for the EU DataGrid.

The interface to the LCMAPS module is composed of:

1. lcmaps_init(): start the PluginManager –> load plugins, start evaluation manager

2. lcmaps_run(): run the PluginManager –> run evaluation manager –> run plugins
3. lcmaps_term(): stop the PluginManager

Definition in file lcmaps.c.

8.10.2 Variable Documentation

8.10.2.1 lcmaps_cred_id_t lcmaps_cred `[static]`

For internal use only.

Definition at line 68 of file lcmaps.c.

8.10.2.2 int lcmaps_initialized = 0 `[static]`

For internal use only.

Definition at line 69 of file lcmaps.c.

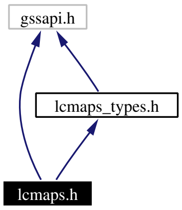
8.11 lcmaps.h File Reference

API of the LCMAPS library.

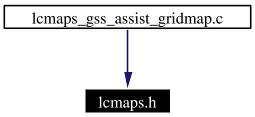
`#include <gssapi.h>`

`#include "lcmaps_types.h"`

Include dependency graph for lcmaps.h:



This graph shows which files directly or indirectly include this file:



Functions

- int lcmaps_init (FILE *fp)

  Initialize the LCMAPS module.

- int lcmaps_term ()

  Terminate the LCMAPS module.

- int lcmaps_run (gss_cred_id_t user_cred, lcmaps_request_t request)

  let LCMAPS handle the user mapping.

- int lcmaps_run_without_credentials (char *user_dn_tmp)

  do the user mapping without credentials, only the user DN.

8.11.1 Detailed Description

API of the LCMAPS library.

Author:

Martijn Steenbakkers for the EU DataGrid.

This header contains the declarations of the LCMAPS library functions:

1. lcmaps_init(): To initialize the LCMAPS module
2. lcmaps_run(): To do the user mapping
3. lcmaps_run_without_credentials(): To do the user mapping, without credentials
4. lcmaps_term(): To cleanly terminate the module

Definition in file lcmaps.h.

8.11.2 Function Documentation

8.11.2.1 int lcmaps_init (FILE * *fp*)

Initialize the LCMAPS module.

The function does the following:

- initialize LCMAPS module.
- setup logging, error handling (not yet).
- start PluginManager

Parameters:

fp file handle for logging (from gatekeeper)

Return values:

0 initialization succeeded.

1 initialization failed.

Definition at line 98 of file lcmaps.c.

8.11.2.2 int lcmaps_run (gss_cred_id_t *user_cred*, lcmaps_request_t *request*)

let LCMAPS handle the user mapping.

This function runs the PluginManager for user mapping.

Parameters:

request authorization request in RSL (later JDL)

user_cred GLOBUS user credential

Return values:

0 mapping succeeded.

1 mapping failed.

Definition at line 168 of file lcmaps.c.

8.11.2.3 int lcmaps_run_without_credentials (char * *user_dn_tmp*)

do the user mapping without credentials, only the user DN.

This function runs the PluginManager for user mapping without credentials.

Parameters:
user_dn_tmp user DN

Return values:
0 mapping succeeded.
1 mapping failed.

Definition at line 239 of file lcmaps.c.

8.11.2.4 int lcmaps_term ()

Terminate the LCMAPS module.

The function does the following:

- terminate the LCMAPS module
- terminate the plugins

Return values:
0 termination succeeded.
1 termination failed.

Definition at line 308 of file lcmaps.c.

8.12 lcmaps_arguments.c File Reference

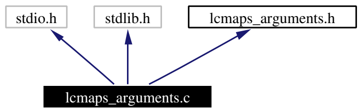
LCMAPS module for creating and passing introspect/run argument lists.

`#include <stdio.h>`

`#include <stdlib.h>`

`#include "lcmaps_arguments.h"`

Include dependency graph for lcmaps_arguments.c:



8.12.1 Detailed Description

LCMAPS module for creating and passing introspect/run argument lists.

Author:
Oscar Koeroo and Martijn Steenbakkers for the EU DataGrid.

The interface is composed of:

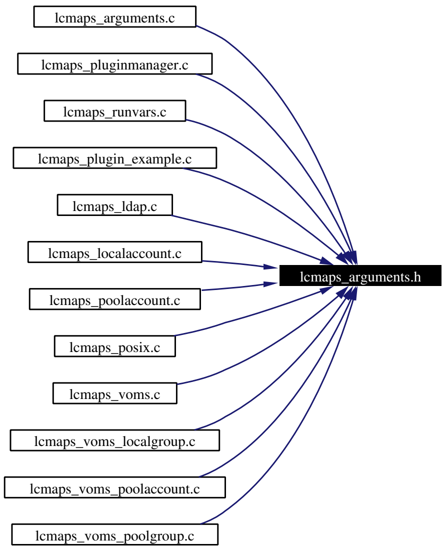
1. lcmaps_setArgValue(): Set the value of argument with name argName of argType to value
2. lcmaps_getArgValue(): Get the value of argument with name argName of argType
3. lcmaps_findArgName(): Get index of argument with name argName
4. lcmaps_cntArgs(): Count the number of arguments

Definition in file lcmaps_arguments.c.

8.13 lcmaps_arguments.h File Reference

Public header file to be used by plugins.

This graph shows which files directly or indirectly include this file:



Data Structures

- struct lcmaps_argument_s

  structure representing an LCMAPS plugin run argument.

Typedefs

- typedef struct lcmaps_argument_s lcmaps_argument_t

  Type of LCMAPS plugin run argument (to be passed to the plugin by plugin_run()).

Functions

- int lcmaps_setArgValue (char ∗argName, char ∗argType, void ∗value, int argcx, lcmaps_argument_t ∗∗argvx)

  Set the value of argument with name argName of argType to value.

- void∗ lcmaps_getArgValue (char ∗argName, char ∗argType, int argcx, lcmaps_argument_t ∗argvx)

  Get the value of argument with name argName of argType.

- int lcmaps_findArgName (char ∗argName, int argcx, lcmaps_argument_t ∗argvx)

Get index of argument with name argName.

- int lcmaps_findArgNameAndType (char ∗argName, char ∗argType, int argcx, lcmaps_argument_t ∗argvx)

  Get index of argument with name argName.

- int lcmaps_cntArgs (lcmaps_argument_t ∗argvx)

  Count the number of arguments.

8.13.1 Detailed Description

Public header file to be used by plugins.

Author:
Martijn Steenbakkers and Oscar Koeroo for the EU DataGrid.

Routines to access the plugin arguments.

The interface is composed of:

1. lcmaps_setArgValue(): Set the value of argument with name argName of argType to value
2. lcmaps_getArgValue(): Get the value of argument with name argName of argType
3. lcmaps_findArgName(): Get index of argument with name argName
4. lcmaps_cntArgs(): Count the number of arguments

Definition in file lcmaps_arguments.h.

8.13.2 Function Documentation

8.13.2.1 int lcmaps_cntArgs (lcmaps_argument_t ∗ *argvx*)

Count the number of arguments.

Count the number of arguments that are defined in a plug-in Returns this number.

Parameters:
argvx array of arguments structures

Returns:
the number of arguments

Definition at line 272 of file lcmaps_arguments.c.

8.13.2.2 int lcmaps_findArgName (char * *argName*, int *argcx*, lcmaps_argument_t * *argvx*)

Get index of argument with name argName.

Search for argName in the arguments list. Returns the index to lcmaps_argument_t element.

Parameters:

argName name of argument

argcx number of arguments

argvx array of arguments structures

Returns:

index to lcmaps_argument_t element

Definition at line 178 of file lcmaps_arguments.c.

8.13.2.3 int lcmaps_findArgNameAndType (char * *argName*, char * *argType*, int *argcx*, lcmaps_argument_t * *argvx*)

Get index of argument with name argName.

Search for argName in the arguments list. Returns the index to lcmaps_argument_t element.

Parameters:

argName name of argument

argType type of argument

argcx number of arguments

argvx array of arguments structures

Returns:

index to lcmaps_argument_t element

Definition at line 229 of file lcmaps_arguments.c.

8.13.2.4 void * lcmaps_getArgValue (char * *argName*, char * *argType*, int *argcx*, lcmaps_argument_t * *argvx*)

Get the value of argument with name argName of argType.

Set the value of argType on the reserved place in values. The place within values is determined by the place where argName is found in the arguments list Returns a void pointer to the value.

Parameters:

argName name of argument

argType type of argument

argcx number of arguments

argvx array of arguments structures

Returns:

void pointer to the value or NULL

Definition at line 130 of file lcmaps_arguments.c.

8.13.2.5 int lcmaps_setArgValue (char * *argName*, char * *argType*, void * *value*, int *argcx*, lcmaps_argument_t ** *argvx*)

Set the value of argument with name argName of argType to value.

Set the value of argType on the reserved place in values. The place within values is determined by the place where argName is found in the arguments list

Parameters:

- ***argName*** name of argument
- ***argType*** type of argument
- ***argcx*** number of arguments
- ***argvx*** array of arguments structures

Returns:

0 in case of succes

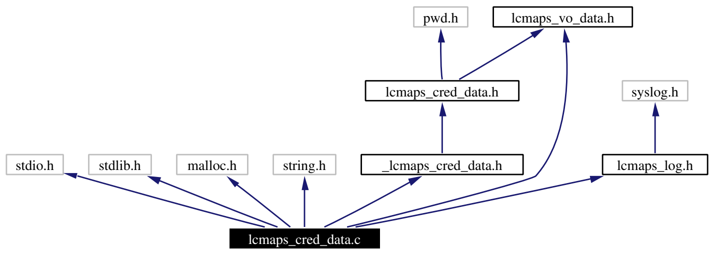
Definition at line 71 of file lcmaps_arguments.c.

8.14 lcmaps_cred_data.c File Reference

Routines to handle lcmaps credential data.

```
#include <stdio.h>
#include <stdlib.h>
#include <malloc.h>
#include <string.h>
#include "_lcmaps_cred_data.h"
#include "lcmaps_log.h"
#include "lcmaps_vo_data.h"
```

Include dependency graph for lcmaps_cred_data.c:



Functions

- void printCredData (int debug_level)

    Get pointer to a list of credential data of a certain type.

8.14.1 Detailed Description

Routines to handle lcmaps credential data.

Author:

Oscar Koeroo and Martijn Steenbakkers for the EU DataGrid.

Definition in file lcmaps_cred_data.c.

8.14.2 Function Documentation

8.14.2.1 void printCredData (int *debug_level*)

Get pointer to a list of credential data of a certain type.

Parameters:

debug_level the debug level

Returns:

nothing

Definition at line 287 of file lcmaps_cred_data.c.

Referenced by stopPluginManager().

8.14.3 Variable Documentation

8.14.3.1 cred_data_t credData [static]

Initial value:

```
{
    (char *) NULL,
    (uid_t *) NULL, (gid_t *) NULL, (gid_t *) NULL,
    (lcmaps_vo_data_t *) NULL, (char **) NULL,
    0, 0, 0, 0, 0,
}
```

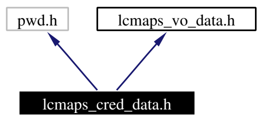
Definition at line 41 of file lcmaps_cred_data.c.

8.15 lcmaps_cred_data.h File Reference

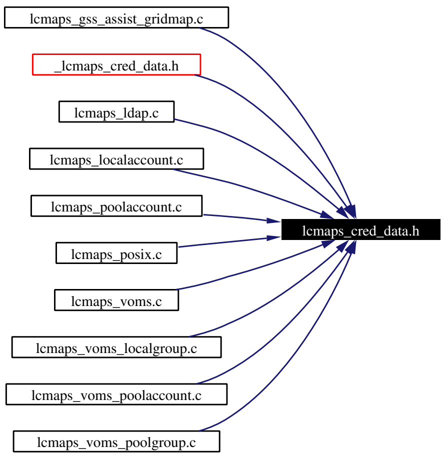
Public header file to be used by plugins.

```
#include <pwd.h>
#include "lcmaps_vo_data.h"
```

Include dependency graph for lcmaps_cred_data.h:



This graph shows which files directly or indirectly include this file:



Data Structures

- struct cred_data_s

  structure that contains the gathered (local) credentials en VOMS info.

Typedefs

- typedef struct cred_data_s cred_data_t

  Type of credential data.

Functions

- int addCredentialData (int datatype, void *data)

Add a credential to the list of found credentials (uids, gids etc).

- void* getCredentialData (int datatype, int *count)

  Get pointer to a list of credential data of a certain type.

8.15.1 Detailed Description

Public header file to be used by plugins.

Routines to access the credential data that are gathered by the plugins.

Author:

Martijn Steenbakkers and Oscar Koeroo for the EU DataGrid.

Definition in file lcmaps_cred_data.h.

8.15.2 Function Documentation

8.15.2.1 int addCredentialData (int *datatype*, void * *data*)

Add a credential to the list of found credentials (uids, gids etc).

The credential value is copied into list (memory is allocated for this)

Parameters:

datatype type of credential

data pointer to credential

Returns:

0 in case of succes

Definition at line 75 of file lcmaps_cred_data.c.

8.15.2.2 void * getCredentialData (int *datatype*, int * *count*)

Get pointer to a list of credential data of a certain type.

Parameters:

datatype type of credential

count number of credentials found in list of datatype (filled by routine)

Returns:

pointer to list of credential data or NULL in case of failure

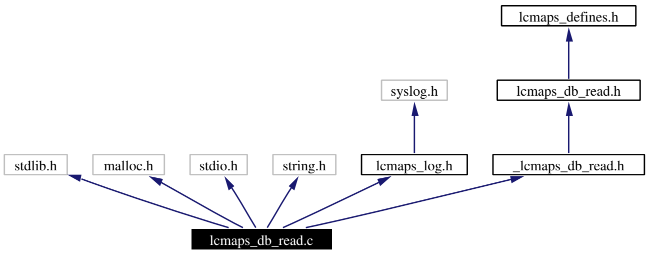
Definition at line 193 of file lcmaps_cred_data.c.

8.16 lcmaps_db_read.c File Reference

the LCMAPS database reader.

```
#include <stdlib.h>
#include <malloc.h>
#include <stdio.h>
#include <string.h>
#include "lcmaps_log.h"
#include "_lcmaps_db_read.h"
```

Include dependency graph for lcmaps_db_read.c:



Defines

- #define MAXDBENTRIES 250
- #define MAXPAIRS 2
- #define WHITESPACE_CHARS " \t\n"
- #define QUOTING_CHARS "\""
- #define ESCAPING_CHARS "\\"
- #define COMMENT_CHARS "#"
- #define PAIR_SEP_CHARS ","
- #define VARVAL_SEP_CHARS "="
- #define PAIR_TERMINATOR_CHARS PAIR_SEP_CHARS WHITESPACE_CHARS
- #define VARVAL_TERMINATOR_CHARS VARVAL_SEP_CHARS WHITESPACE_CHARS
- #define NUL '\0'

Functions

- int lcmaps_db_read_entries (FILE ∗)

  Read db entries from stream and fill a lsit of db entries.

- int lcmaps_db_parse_line (char ∗, lcmaps_db_entry_t ∗∗)

  Parses database line and fills database structure.

- int lcmaps_db_parse_pair (char *, char **, char **)

  Parses a database variable-value pair and returns the variable name and its value.

- int lcmaps_db_parse_string (char **)

  Takes a string and removes prepending and trailing spaces and quotes (unless escaped).

Variables

- lcmaps_db_entry_t* lcmaps_db_list = NULL

8.16.1 Detailed Description

the LCMAPS database reader.

Author:

Martijn Steenbakkers for the EU DataGrid.

Definition in file lcmaps_db_read.c.

8.16.2 Define Documentation

8.16.2.1 #define COMMENT_CHARS "#"

For internal use only.

Definition at line 37 of file lcmaps_db_read.c.

8.16.2.2 #define ESCAPING_CHARS "\\"

For internal use only.

Definition at line 36 of file lcmaps_db_read.c.

8.16.2.3 #define MAXDBENTRIES 250

maximum number of LCMAPS database entries

For internal use only.

Definition at line 30 of file lcmaps_db_read.c.

8.16.2.4 #define MAXPAIRS 2

maximum number of variable-value pairs that will be parsed per line

For internal use only.

Definition at line 31 of file lcmaps_db_read.c.

8.16.2.5 #define NUL '\0'

For internal use only.

Definition at line 60 of file lcmaps_db_read.c.

8.16.2.6 #define PAIR_SEP_CHARS ","

Characters separating variable-value pairs in the lcmaps database file

For internal use only.

Definition at line 40 of file lcmaps_db_read.c.

8.16.2.7 #define PAIR_TERMINATOR_CHARS PAIR_SEP_CHARS WHITESPACE_CHARS

Characters that terminate pairs in the lcmaps database file. This is a combination of whitespace and separators.

For internal use only.

Definition at line 52 of file lcmaps_db_read.c.

8.16.2.8 #define QUOTING_CHARS "\""

For internal use only.

Definition at line 35 of file lcmaps_db_read.c.

8.16.2.9 #define VARVAL_SEP_CHARS "="

Characters separating variables from values

For internal use only.

Definition at line 42 of file lcmaps_db_read.c.

8.16.2.10 #define VARVAL_TERMINATOR_CHARS VARVAL_SEP_CHARS WHITESPACE_CHARS

Characters that terminate variables and values in the lcmaps database file. This is a combination of whitespace and separators.

For internal use only.

Definition at line 57 of file lcmaps_db_read.c.

8.16.2.11 #define WHITESPACE_CHARS " \t\n"

For internal use only.

Definition at line 34 of file lcmaps_db_read.c.

8.16.3 Function Documentation

8.16.3.1 int lcmaps_db_parse_line (char * *line*, lcmaps_db_entry_t ** *entry*) [static]

Parses database line and fills database structure.

Parameters:

line database line

entry pointer to a pointer to a database structure (can/should be freed afterwards)

Return values:

1 succes.

0 failure.

For internal use only.

Definition at line 261 of file lcmaps_db_read.c.

Referenced by lcmaps_db_read_entries().

8.16.3.2 int lcmaps_db_parse_pair (char * *pair*, char ** *pvar*, char ** *pval*) [static]

Parses a database variable-value pair and returns the variable name and its value.

Parameters:

pair string containing the pair

pvar pointer to the variable string

pval pointer to the value string

Return values:

1 succes.

0 failure.

For internal use only.

Definition at line 400 of file lcmaps_db_read.c.

Referenced by lcmaps_db_parse_line().

8.16.3.3 int lcmaps_db_parse_string (char ** *pstring*) [static]

Takes a string and removes prepending and trailing spaces and quotes (unless escaped).

Parameters:

pstring pointer to a pointer to a char

Return values:

1 succes.

0 failure.

For internal use only.

Definition at line 497 of file lcmaps_db_read.c.

Referenced by lcmaps_db_parse_pair().

8.16.3.4 int lcmaps_db_read_entries (FILE * *dbstream*) [static]

Read db entries from stream and fill a lsit of db entries.

Parameters:

dbstream database stream

Returns:

the number of entries found (failure –> negative number)
For internal use only.

Definition at line 132 of file lcmaps_db_read.c.

Referenced by lcmaps_db_read().

8.16.4 Variable Documentation

8.16.4.1 lcmaps_db_entry_t * lcmaps_db_list = NULL [static]

list of database entries

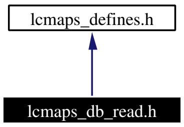
Definition at line 74 of file lcmaps_db_read.c.

8.17 lcmaps_db_read.h File Reference

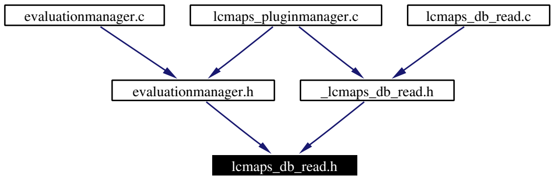
header file for LCMAPS database structure.

`#include "lcmaps_defines.h"`

Include dependency graph for lcmaps_db_read.h:



This graph shows which files directly or indirectly include this file:



Data Structures

- struct lcmaps_db_entry_s

    LCMAPS data base element structure.

Typedefs

- typedef struct lcmaps_db_entry_s lcmaps_db_entry_t

    type of LCMAPS data base element.

8.17.1 Detailed Description

header file for LCMAPS database structure.

Author:
Martijn Steenbakkers for the EU DataGrid.

This header contains the declarations of the LCMAPS database structure

For internal use only.

Definition in file lcmaps_db_read.h.

8.17.2 Typedef Documentation

8.17.2.1 typedef struct lcmaps_db_entry_s lcmaps_db_entry_t

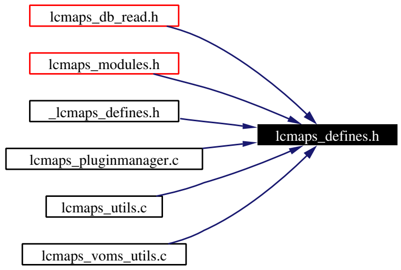
type of LCMAPS data base element.

For internal use only.

8.18 lcmaps_defines.h File Reference

Public header file with common definitions for the LCMAPS (authorization modules).

This graph shows which files directly or indirectly include this file:



Defines

- #define LCMAPS_MOD_SUCCESS (int)(0)
- #define LCMAPS_MOD_FAIL (int)(1)
- #define LCMAPS_MOD_NOFILE (int)(2)
- #define LCMAPS_MOD_ENTRY (int)(3)
- #define LCMAPS_MOD_NOENTRY (int)(4)
- #define LCMAPS_ETC_HOME "/opt/edg/etc/lcmaps"
- #define LCMAPS_LIB_HOME "/opt/edg/lib/lcmaps"
- #define LCMAPS_MOD_HOME "/opt/edg/lib/lcmaps/modules"
- #define LCMAPS_MAXPATHLEN 500
- #define LCMAPS_MAXARGSTRING 1000
- #define LCMAPS_MAXARGS 51

8.18.1 Detailed Description

Public header file with common definitions for the LCMAPS (authorization modules).

Author:

Martijn Steenbakkers for the EU DataGrid.

Here the return values for the LCMAPS plugins/modules are defined as well as the default locations of the LCMAPS "etc", "lib" and "modules" directories.

Definition in file lcmaps_defines.h.

8.18.2 Define Documentation

8.18.2.1 #define LCMAPS_ETC_HOME "/opt/edg/etc/lcmaps"

default directory for LCMAPS configuration data bases

Definition at line 39 of file lcmaps_defines.h.

8.18.2.2 #define LCMAPS_LIB_HOME "/opt/edg/lib/lcmaps"

default directory for the LCMAPS library

Definition at line 41 of file lcmaps_defines.h.

8.18.2.3 #define LCMAPS_MAXARGS 51

maximum number of arguments (+1) to be passed to LCMAPS authorization plugins/modules.

For internal use only.

Definition at line 50 of file lcmaps_defines.h.

8.18.2.4 #define LCMAPS_MAXARGSTRING 1000

maximum length of the plugin argument string as specified in the LCMAPS database.

For internal use only.

Definition at line 48 of file lcmaps_defines.h.

8.18.2.5 #define LCMAPS_MAXPATHLEN 500

maximum path lengths of files, used in plugin and database structures.

For internal use only.

Definition at line 46 of file lcmaps_defines.h.

8.18.2.6 #define LCMAPS_MOD_ENTRY (int)(3)

Return value of LCMAPS plugin module indicating that an entry was found

Definition at line 34 of file lcmaps_defines.h.

8.18.2.7 #define LCMAPS_MOD_FAIL (int)(1)

Return value of LCMAPS plugin module indicating failure (no authorization)

Definition at line 30 of file lcmaps_defines.h.

8.18.2.8 #define LCMAPS_MOD_HOME "/opt/edg/lib/lcmaps/modules"

default directory for the LCMAPS plugins/modules

Definition at line 43 of file lcmaps_defines.h.

8.18.2.9 #define LCMAPS_MOD_NOENTRY (int)(4)

Return value of LCMAPS plugin module indicating that no entry was found

Definition at line 36 of file lcmaps_defines.h.

8.18.2.10 #define LCMAPS_MOD_NOFILE (int)(2)

Return value of LCMAPS plugin module indicating that no file could be found

Definition at line 32 of file lcmaps_defines.h.

8.18.2.11 #define LCMAPS_MOD_SUCCESS (int)(0)

Return value of LCMAPS plugin module indicating succes (authorization granted)

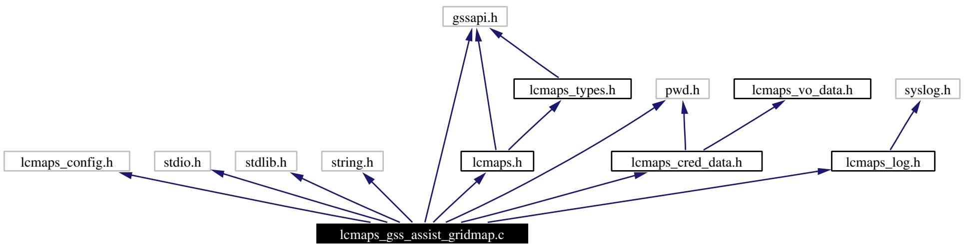
Definition at line 28 of file lcmaps_defines.h.

8.19 lcmaps_gss_assist_gridmap.c File Reference

legacy interface for LCMAPS.

```
#include "lcmaps_config.h"
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <gssapi.h>
#include <pwd.h>
#include "lcmaps.h"
#include "lcmaps_log.h"
#include "lcmaps_cred_data.h"
```

Include dependency graph for lcmaps_gss_assist_gridmap.c:



8.19.1 Detailed Description

legacy interface for LCMAPS.

Author:

Martijn Steenbakkers for the EU DataGrid.

The legacy interface to the LCMAPS module is the original gridmap interface provided by globus. Given the user distinguished name (DN) a username is returned, based on the gridmap file

1. globus_gss_assist_gridmap: the interface

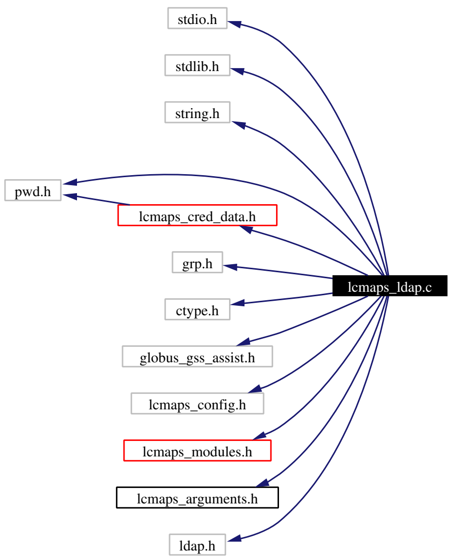
Definition in file lcmaps_gss_assist_gridmap.c.

8.20 lcmaps_ldap.c File Reference

Interface to the LCMAPS plugins.

```
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <pwd.h>
#include <grp.h>
#include <ctype.h>
#include "globus_gss_assist.h"
#include "lcmaps_config.h"
#include "lcmaps_modules.h"
#include "lcmaps_arguments.h"
#include "lcmaps_cred_data.h"
#include "ldap.h"
```

Include dependency graph for lcmaps_ldap.c:



Functions

- int lcmaps_add_username_to_ldapgroup (const char ∗username, const char ∗groupname, gid_t groupnumber, LDAP ∗ld_handle, const char ∗searchBase)

    Adds the username to the appropriate (LDAP) group.

- int lcmaps_set_pgid (const char ∗username, const char ∗pgroupname, gid_t pgroupnumber, LDAP ∗ld_handle, const char ∗searchBase)

  Sets the primary group ID.

- int lcmaps_get_ldap_pw (const char ∗path, char ∗∗ldap_passwd)

  Get the LDAP password from file.

8.20.1 Detailed Description

Interface to the LCMAPS plugins.

Author:
Wim Som de Cerff and Martijn Steenbakkers for the EU DataGrid.

This file contains the code for the ldap LCMAPS plugin The interface consists of the following functions:

1. plugin_initialize()
2. plugin_run()
3. plugin_terminate()
4. plugin_introspect()

The following internal functions are available:

1. lcmaps_set_pgid()
2. lcmaps_add_username_to_ldapgroup()

Definition in file lcmaps_ldap.c.

8.20.2 Function Documentation

8.20.2.1 int lcmaps_add_username_to_ldapgroup (const char ∗ *username*, const char ∗ *groupname*, gid_t *groupnumber*, LDAP ∗ *ld_handle*, const char ∗ *searchBase*)

Adds the username to the appropriate (LDAP) group.

This function tries to add the username to the list of usernames belonging to the group with name groupname and gid groupnumber in the posixGroup LDAP structure. If the group does not exist, -1 is returned.

Parameters:

username the name of the user

groupname the name of the group

groupnumber group id number

ld_handle handle to LDAP

searchBase dn search base

Return values:

0 success

-1 ldap failure

1 other failure

Definition at line 887 of file lcmaps_ldap.c.

8.20.2.2 int lcmaps_get_ldap_pw (const char * *path*, char ** *ldap_passwd*)

Get the LDAP password from file.

This function tries to read the LDAP password from the ldap_pw file. It also tests if the access bits of the file are correctly set.

Parameters:

path the path to the ldap_pw file containing the password.

ldap_passwd variable to set the password in

Return values:

0 success

1 other failure

Definition at line 1235 of file lcmaps_ldap.c.

8.20.2.3 int lcmaps_set_pgid (const char * *username*, const char * *pgroupname*, gid_t *pgroupnumber*, LDAP * *ld_handle*, const char * *searchBase*)

Sets the primary group ID.

This function tries to set the primary group in the posixAccount LDAP structure for the user "username".

Parameters:

username the name of the user

pgroupname the name of the primary group

pgroupnumber primary group id number

ld_handle handle to LDAP

searchBase dn search base

Return values:

0 success

-1 ldap failure

1 other failure

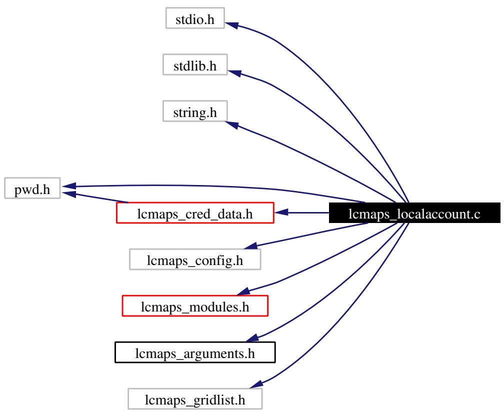
Definition at line 1129 of file lcmaps_ldap.c.

8.21 lcmaps_localaccount.c File Reference

Interface to the LCMAPS plugins.

```
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <pwd.h>
#include "lcmaps_config.h"
#include "lcmaps_modules.h"
#include "lcmaps_arguments.h"
#include "lcmaps_cred_data.h"
#include "lcmaps_gridlist.h"
```

Include dependency graph for lcmaps_localaccount.c:



8.21.1 Detailed Description

Interface to the LCMAPS plugins.

Author:
Martijn Steenbakkers for the EU DataGrid.

This file contains the code for localaccount plugin

1. plugin_initialize()
2. plugin_run()
3. plugin_terminate()
4. plugin_introspect()

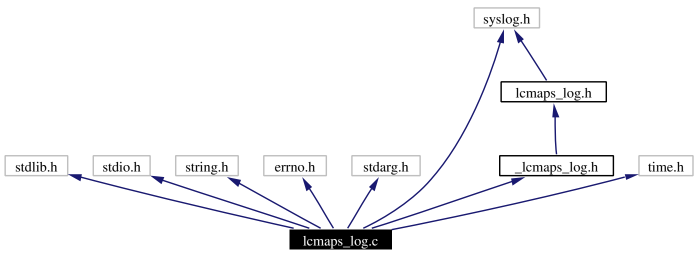
Definition in file lcmaps_localaccount.c.

8.22 lcmaps_log.c File Reference

Logging routines for LCMAPS.

```
#include <stdlib.h>
#include <stdio.h>
#include <string.h>
#include <errno.h>
#include <stdarg.h>
#include <syslog.h>
#include <time.h>
#include "_lcmaps_log.h"
```

Include dependency graph for lcmaps_log.c:



Defines

- #define DEBUG_LEVEL 0

Variables

- FILE∗ lcmaps_logfp = NULL
- int logging_usrlog = 0
- int logging_syslog = 0
- int debug_level = DEBUG_LEVEL

8.22.1 Detailed Description

Logging routines for LCMAPS.

Author:

Martijn Steenbakkers for the EU DataGrid.

Definition in file lcmaps_log.c.

8.22.2 Define Documentation

8.22.2.1 #define DEBUG_LEVEL 0

default debugging level

Definition at line 34 of file lcmaps_log.c.

8.22.3 Variable Documentation

8.22.3.1 int debug_level = DEBUG_LEVEL `[static]`

debugging level

For internal use only.

Definition at line 43 of file lcmaps_log.c.

8.22.3.2 FILE * lcmaps_logfp = NULL `[static]`

logfile descriptor.

For internal use only.

Definition at line 40 of file lcmaps_log.c.

8.22.3.3 int logging_syslog = 0 `[static]`

flag to use syslog

For internal use only.

Definition at line 42 of file lcmaps_log.c.

8.22.3.4 int logging_usrlog = 0 `[static]`

flag to do user logging

For internal use only.

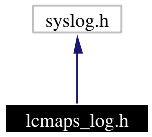
Definition at line 41 of file lcmaps_log.c.

8.23 lcmaps_log.h File Reference

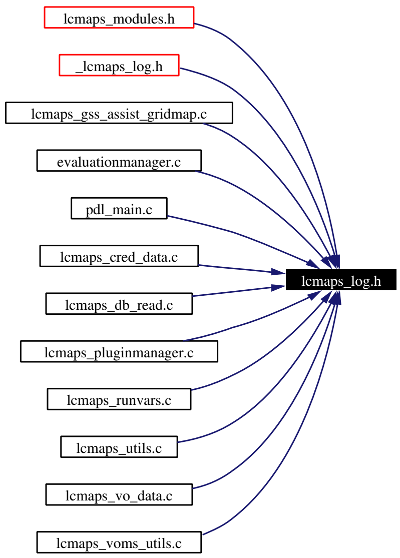
Logging API for the LCMAPS plugins and LCMAPS itself.

`#include <syslog.h>`

Include dependency graph for lcmaps_log.h:



This graph shows which files directly or indirectly include this file:



Functions

- int lcmaps_log (int prty, char ∗fmt,...)

  log information.

- int lcmaps_log_debug (int debug_lvl, char ∗fmt,...)

  Print debugging information.

- int lcmaps_log_time (int prty, char ∗fmt,...)

  log information with timestamp.

8.23.1 Detailed Description

Logging API for the LCMAPS plugins and LCMAPS itself.

Author:
Martijn Steenbakkers for the EU DataGrid.

This header contains the declarations of the LCMAPS logging functions The LCMAPS plugins can use this API to write output to the LCMAPS logging devices.

1. lcmaps_log(): Log to LCMAPS logging devices.
2. lcmaps_log_debug(): Produce debugging output.

Definition in file lcmaps_log.h.

8.23.2 Function Documentation

8.23.2.1 int lcmaps_log (int *prty*, char * *fmt*, ...)

log information.

This function logs information for LCMAPS and its plugins. Syslog() is called with the specified priority. No syslog() is done if the priority is 0.

Parameters:
prty syslog priority (if 0 don't syslog).
fmt string format
... variable argument list

Return values:
0 succes.
1 failure.

Definition at line 147 of file lcmaps_log.c.

8.23.2.2 int lcmaps_log_debug (int *debug_lvl*, char * *fmt*, ...)

Print debugging information.

This function prints debugging information (using lcmaps_log with priority 0) provided debug_lvl <= DEBUG_LEVEL (default is 0).

Parameters:
debug_lvl debugging level
fmt string format
... variable argument list

Return values:

- ***0*** succes.
- ***1*** failure.

Definition at line 207 of file lcmaps_log.c.

8.23.2.3 int lcmaps_log_time (int *prty*, char ∗ *fmt*, ...)

log information with timestamp.

This function logs information with a timestamp for LCMAPS and its plugins. Syslog() is called with the specified priority. No syslog() is done if the priority is 0.

Parameters:

- ***prty*** syslog priority (if 0 don't syslog).
- ***fmt*** string format
- **...** variable argument list

Return values:

- ***0*** succes.
- ***1*** failure.

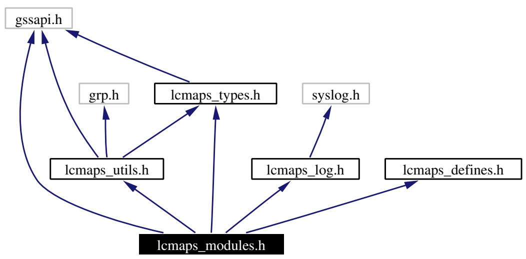
Definition at line 287 of file lcmaps_log.c.

8.24 lcmaps_modules.h File Reference

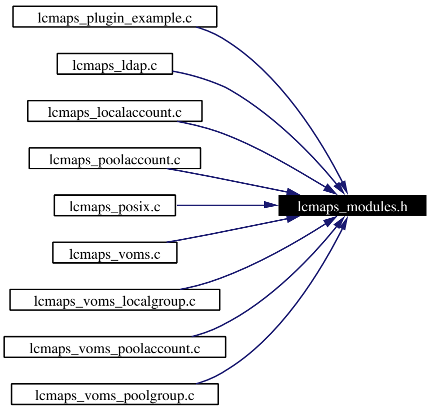
The LCMAPS authorization plugins/modules should "include" this file.

```
#include <gssapi.h>
#include "lcmaps_utils.h"
#include "lcmaps_log.h"
#include "lcmaps_types.h"
#include "lcmaps_defines.h"
```

Include dependency graph for lcmaps_modules.h:



This graph shows which files directly or indirectly include this file:



8.24.1 Detailed Description

The LCMAPS authorization plugins/modules should "include" this file.

Author:

Martijn Steenbakkers for the EU DataGrid.

This file includes the header files that are needed by the LCMAPS authorization plugins/modules.

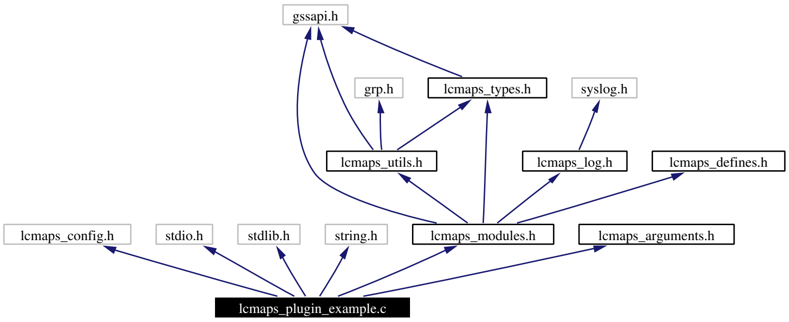
Definition in file lcmaps_modules.h.

8.25 lcmaps_plugin_example.c File Reference

Interface to the LCMAPS plugins.

```
#include "lcmaps_config.h"
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include "lcmaps_modules.h"
#include "lcmaps_arguments.h"
```

Include dependency graph for lcmaps_plugin_example.c:



Functions

- int plugin_introspect (int ∗argc, lcmaps_argument_t ∗∗argv)

  Plugin asks for required arguments.

- int plugin_initialize (int argc, char ∗∗argv)

  initialize the plugin.

- int plugin_run (int argc, lcmaps_argument_t ∗argv)

  Gather credentials for user making use of the ordered arguments.

- int plugin_terminate ()

  Whatever is needed to terminate the plugin module goes in here.

8.25.1 Detailed Description

Interface to the LCMAPS plugins.

Author:

Martijn Steenbakkers for the EU DataGrid.

This file contains the code for an example LCMAPS plugin and shows the interface the plugin has to provide to the LCMAPS. The interface consists of the following functions:

1. plugin_initialize()
2. plugin_run()
3. plugin_terminate()
4. plugin_introspect()

Definition in file lcmaps_plugin_example.c.

8.25.2 Function Documentation

8.25.2.1 int plugin_initialize (int *argc*, char ∗∗ *argv*)

initialize the plugin.

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

Parameters:

argc number of passed arguments.

argv argument list. argv[0] contains the name of the plugin.

Return values:

LCMAPS_MOD_SUCCESS successful initialization

LCMAPS_MOD_FAIL failure in the plugin initialization

LCMAPS_MOD_NOFILE private plugin database could not be found (same effect as LCMAPS_-MOD_FAIL)

Definition at line 139 of file lcmaps_plugin_example.c.

8.25.2.2 int plugin_introspect (int ∗ *argc*, lcmaps_argument_t ∗∗ *argv*)

Plugin asks for required arguments.

Parameters:

int ∗ argc

lcmaps_argument_t ∗∗ argv

Return values:

LCMAPS_MOD_SUCCESS success

LCMAPS_MOD_FAIL failure (will result in a lcmaps failure)

Definition at line 87 of file lcmaps_plugin_example.c.

8.25.2.3 int plugin_run (int *argc*, lcmaps_argument_t * *argv*)

Gather credentials for user making use of the ordered arguments.

Ask for credentials by passing the arguments (like JDL, globus DN, VOMS roles etc.) that were ordered earlier by the plugin_introspect() function

Parameters:

- ***argc*** number of arguments
- ***argv*** list of arguments

Return values:

- ***LCMAPS_MOD_SUCCESS*** authorization succeeded
- ***LCMAPS_MOD_FAIL*** authorization failed

Definition at line 182 of file lcmaps_plugin_example.c.

8.25.2.4 int plugin_terminate ()

Whatever is needed to terminate the plugin module goes in here.

Return values:

- ***LCMAPS_MOD_SUCCESS*** success
- ***LCMAPS_MOD_FAIL*** failure (will result in an authorization failure)

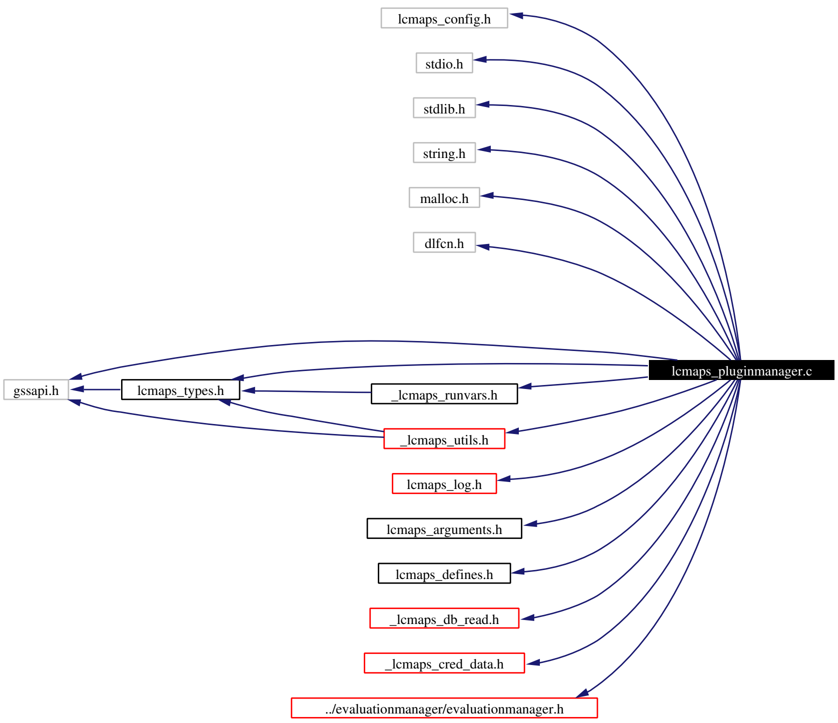
Definition at line 252 of file lcmaps_plugin_example.c.

8.26 lcmaps_pluginmanager.c File Reference

the plugin manager for LCMAPS.

```
#include "lcmaps_config.h"
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <malloc.h>
#include <dlfcn.h>
#include <gssapi.h>
#include "lcmaps_types.h"
#include "lcmaps_log.h"
#include "lcmaps_arguments.h"
#include "lcmaps_defines.h"
#include "_lcmaps_utils.h"
#include "_lcmaps_db_read.h"
#include "_lcmaps_runvars.h"
#include "_lcmaps_cred_data.h"
#include "../evaluationmanager/evaluationmanager.h"
```

Include dependency graph for lcmaps_pluginmanager.c:



Data Structures

- struct lcmaps_plugindl_s

    the lcmaps plugin module structure.

Defines

- #define NUL '\0'
- #define MAXPROCS 4

Typedefs

- typedef int (∗ lcmaps_proc_t)()

    this type corresponds to the types of the plugin interface functions.

- typedef struct lcmaps_plugindl_s lcmaps_plugindl_t

    the type definition of the lcmaps plugin module structure.

Enumerations

- enum lcmaps_proctype_e { INITPROC, RUNPROC, TERMPROC, INTROPROC, **ENDOFPROCS** }

    This enumeration type gives the different plugin symbol/function types.

Functions

- lcmaps_plugindl_t* PluginInit (lcmaps_db_entry_t *, lcmaps_plugindl_t **)

    Initialize the plugin.

- lcmaps_proc_t get_procsymbol (void *, char *)

    get procedure symbol from dlopen-ed library.

- int print_lcmaps_plugin (int, lcmaps_plugindl_t *)

    print the lcmaps_plugindl_t structure.

- int parse_args_plugin (const char *, const char *, char **, int *)

    convert plugin argument string into xargc, xargv.

- int clean_plugin_list (lcmaps_plugindl_t **)

    clean (free) the list of plugins and call the plugin termination functions.

Variables

- char* lcmaps_db_file_default = NULL
- char* lcmaps_dir = NULL
- lcmaps_plugindl_t* plugin_list = NULL

8.26.1 Detailed Description

the plugin manager for LCMAPS.

Author:

Martijn Steenbakkers for the EU DataGrid.

The interface to the PluginManager module is composed of:

1. startPluginManager(): start the PluginManager –> load plugins, start evaluation manager
2. runPluginManager(): run the PluginManager –> run evaluation manager –> run plugins
3. stopPluginManager(): stop the PluginManager
4. reloadPluginManager(): reload the PluginManager ? (will we implement this ?)
5. runPlugin(): run the specified plugin. (used by Evaluation Manager)

Definition in file lcmaps_pluginmanager.c.

8.26.2 Define Documentation

8.26.2.1 #define MAXPROCS 4

maximum number of interface symbols in plugin modules

For internal use only.

Definition at line 64 of file lcmaps_pluginmanager.c.

8.26.2.2 #define NUL '\0'

NUL character

For internal use only.

Definition at line 60 of file lcmaps_pluginmanager.c.

8.26.3 Typedef Documentation

8.26.3.1 typedef struct lcmaps_plugindl_s lcmaps_plugindl_t

the type definition of the lcmaps plugin module structure.

For internal use only.

8.26.3.2 typedef int(∗ lcmaps_proc_t)()

this type corresponds to the types of the plugin interface functions.

For internal use only.

Definition at line 90 of file lcmaps_pluginmanager.c.

8.26.4 Enumeration Type Documentation

8.26.4.1 enum lcmaps_proctype_e

This enumeration type gives the different plugin symbol/function types.

For internal use only.

Enumeration values:

- ***INITPROC*** this value corresponds to the plugin initialization function
- ***RUNPROC*** this value corresponds to the plugin run function (get credentials)
- ***TERMPROC*** this value corresponds to the plugin termination function
- ***INTROPROC*** this value corresponds to the plugin introspect function

Definition at line 76 of file lcmaps_pluginmanager.c.

8.26.5 Function Documentation

8.26.5.1 lcmaps_plugindl_t * PluginInit (lcmaps_db_entry_t * *db_handle*, lcmaps_plugindl_t ** *list*) [static]

Initialize the plugin.

This function takes a plugin LCMAPS database entry and performs the following tasks:

- Create entry in (plugin)list
- Open the plugins and check the symbols plugin_init and confirm_authorization
- run plugin_init

Parameters:

db_handle handle to LCMAPS db (containing pluginname and pluginargs)

list pointer to plugin structure list to which (plugin) module has to be added

Returns:

pointer to newly created plugin structure or NULL in case of failure
For internal use only.

Definition at line 354 of file lcmaps_pluginmanager.c.

Referenced by startPluginManager().

8.26.5.2 int clean_plugin_list (lcmaps_plugindl_t ** *list*) [static]

clean (free) the list of plugins and call the plugin termination functions.

Parameters:

list

list pointer to list of plugins which has to be freeed.

Return values:

0 succes.

1 failure.
For internal use only.

Definition at line 781 of file lcmaps_pluginmanager.c.

Referenced by startPluginManager(), and stopPluginManager().

8.26.5.3 lcmaps_proc_t get_procsymbol (void * *handle*, char * *symname*) [static]

get procedure symbol from dlopen-ed library.

Parameters:

handle handle of dynamic library

symname name of procedure symbol

Returns:

handle to procedure symbol or NUll
For internal use only.

Definition at line 639 of file lcmaps_pluginmanager.c.

Referenced by PluginInit().

8.26.5.4 int parse_args_plugin (const char * *name*, const char * *argstring*, char ** *xargv*, int * *xargc*) [static]

convert plugin argument string into xargc, xargv.

Parse the argument string of the plugin and create xargv and xargc

Parameters:

name name of the plugin (goes into xargv[0])

argstring string containing the arguments

xargv array of argument strings (has to be freed later)

xargc number of arguments

Return values:

0 succes.

1 failure.

For internal use only.

Definition at line 578 of file lcmaps_pluginmanager.c.

Referenced by PluginInit().

8.26.5.5 int print_lcmaps_plugin (int *debug_lvl*, lcmaps_plugindl_t * *plugin*) [static]

print the lcmaps_plugindl_t structure.

Parameters:

debug_lvl debugging level

plugin plugin structure

Return values:

0 succes.

1 failure.

For internal use only.

Definition at line 680 of file lcmaps_pluginmanager.c.

Referenced by runPluginManager(), and startPluginManager().

8.26.6 Variable Documentation

8.26.6.1 char * lcmaps_db_file_default = NULL [static]

For internal use only.

Definition at line 128 of file lcmaps_pluginmanager.c.

8.26.6.2 char * lcmaps_dir = NULL [static]

For internal use only.

Definition at line 129 of file lcmaps_pluginmanager.c.

8.26.6.3 lcmaps_plugindl_t * plugin_list = NULL [static]

For internal use only.

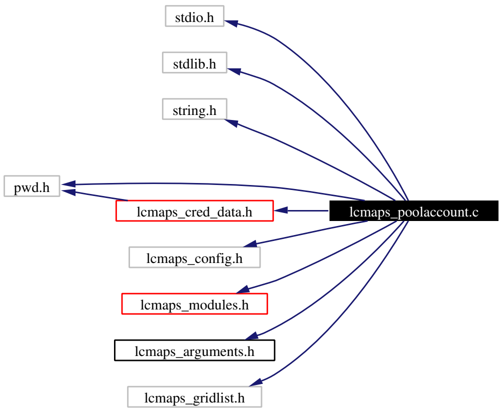
Definition at line 130 of file lcmaps_pluginmanager.c.

8.27 lcmaps_poolaccount.c File Reference

Interface to the LCMAPS plugins.

```
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <pwd.h>
#include "lcmaps_config.h"
#include "lcmaps_modules.h"
#include "lcmaps_arguments.h"
#include "lcmaps_cred_data.h"
#include "lcmaps_gridlist.h"
```

Include dependency graph for lcmaps_poolaccount.c:



8.27.1 Detailed Description

Interface to the LCMAPS plugins.

Author:
Martijn Steenbakkers for the EU DataGrid.

This file contains the code of the poolaccount plugin

1. plugin_initialize()
2. plugin_run()
3. plugin_terminate()
4. plugin_introspect()

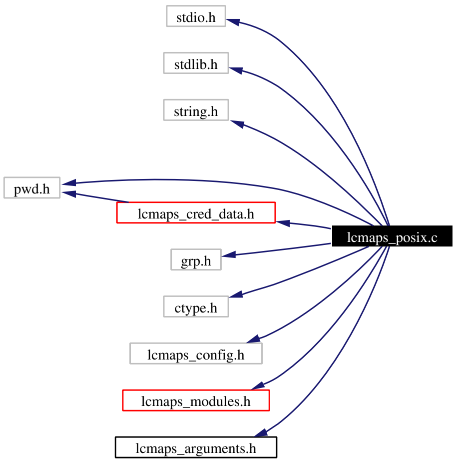
Definition in file lcmaps_poolaccount.c.

8.28 lcmaps_posix.c File Reference

Interface to the LCMAPS plugins.

```
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <pwd.h>
#include <grp.h>
#include <ctype.h>
#include "lcmaps_config.h"
#include "lcmaps_modules.h"
#include "lcmaps_arguments.h"
#include "lcmaps_cred_data.h"
```

Include dependency graph for lcmaps_posix.c:



8.28.1 Detailed Description

Interface to the LCMAPS plugins.

Author:
Martijn Steenbakkers for the EU DataGrid.

This file contains the code for the posix process enforcement LCMAPS plugin

1. plugin_initialize()
2. plugin_run()

3. plugin_terminate()
4. plugin_introspect()

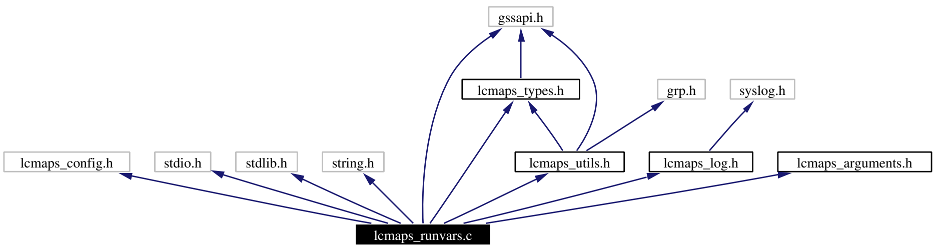
Definition in file lcmaps_posix.c.

8.29 lcmaps_runvars.c File Reference

Extract variables that will be used by the plugins.

```
#include "lcmaps_config.h"
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <gssapi.h>
#include "lcmaps_log.h"
#include "lcmaps_types.h"
#include "lcmaps_utils.h"
#include "lcmaps_arguments.h"
```

Include dependency graph for lcmaps_runvars.c:



Variables

- lcmaps_argument_t runvars_list [ ]

8.29.1 Detailed Description

Extract variables that will be used by the plugins.

Author:
Martijn Steenbakkers for the EU DataGrid.

This module takes the data that are presented to LCMAPS (the global credential and Job request) and extracts the variables that will be used by the plugins from it and stores them into a list. The interface to the LCMAPS module is composed of:

1. lcmaps_extractRunVars(): takes the global credential and Job request and extracts run variables from them
2. lcmaps_setRunVars(): adds run variables to a list

3. lcmaps_getRunVars(): gets run variables from list

   For internal use only.

Definition in file lcmaps_runvars.c.

8.29.2 Variable Documentation

8.29.2.1 lcmaps_argument_t runvars_list [static]

Initial value:

```
{
    { "user_dn"     , "char *"             ,  0,    NULL},
    { "user_cred"   , "gss_cred_id_t"      ,  0,    NULL},
    { "lcmaps_cred" , "lcmaps_cred_id_t" ,  0,    NULL},
    { "job_request" , "lcmaps_request_t" ,  0,    NULL},
    { "job_request" , "char *"             ,  0,    NULL},
    { NULL          , NULL                 , -1,    NULL}
}
```

For internal use only.

Definition at line 57 of file lcmaps_runvars.c.

8.30 lcmaps_test.c File Reference

Program to test the LCMAPS and its plugins.

`#include "lcmaps_config.h"`

`#include <stdio.h>`

`#include <stdlib.h>`

`#include <string.h>`

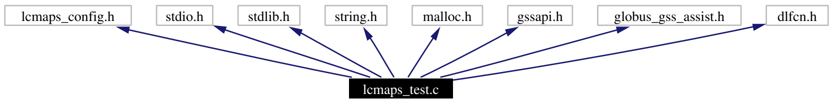
`#include <malloc.h>`

`#include <gssapi.h>`

`#include "globus_gss_assist.h"`

`#include <dlfcn.h>`

Include dependency graph for lcmaps_test.c:



8.30.1 Detailed Description

Program to test the LCMAPS and its plugins.

Author:
Martijn Steenbakkers for the EU DataGrid.

This program has elements of the edg-gatekeeper to be able to test the LCMAPS and its plugins without having the edg-gatekeeper installed. To run it : just run ./lcmaps-test It is not possible (yet) to feed a user credential (proxy) to the program.

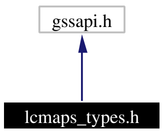
Definition in file lcmaps_test.c.

8.31 lcmaps_types.h File Reference

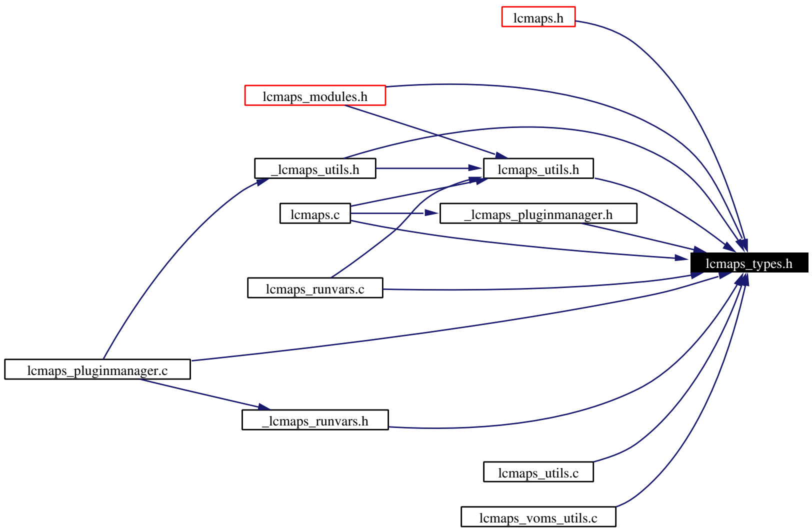
Public header file with typedefs for LCMAPS.

`#include <gssapi.h>`

Include dependency graph for lcmaps_types.h:



This graph shows which files directly or indirectly include this file:



Data Structures

- struct lcmaps_cred_id_s

  structure representing an LCMAPS credential.

Typedefs

- typedef char∗ lcmaps_request_t

  Type of the LCMAPS request expressed in RSL/JDL.

- typedef struct lcmaps_cred_id_s lcmaps_cred_id_t

  Type of LCMAPS credentials.

8.31.1 Detailed Description

Public header file with typedefs for LCMAPS.

Author:
Martijn Steenbakkers for the EU DataGrid.

Definition in file lcmaps_types.h.

8.31.2 Typedef Documentation

8.31.2.1 typedef char * lcmaps_request_t

Type of the LCMAPS request expressed in RSL/JDL.

(Internal) just a string.

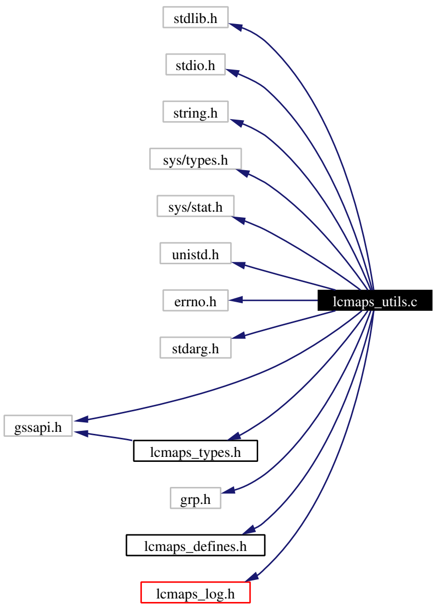
Definition at line 37 of file lcmaps_types.h.

8.32 lcmaps_utils.c File Reference

the utilities for the LCMAPS.

```
#include <stdlib.h>
#include <stdio.h>
#include <string.h>
#include <sys/types.h>
#include <sys/stat.h>
#include <unistd.h>
#include <errno.h>
#include <stdarg.h>
#include <gssapi.h>
#include <grp.h>
#include "lcmaps_defines.h"
#include "lcmaps_types.h"
#include "lcmaps_log.h"
```

Include dependency graph for lcmaps_utils.c:



Functions

- char∗ cred_to_dn (gss_cred_id_t)

  Get the globus DN from GLOBUS credential (gssapi).

- int fexist (char ∗)
  check the existence of file corresponding to <path>.

8.32.1 Detailed Description

the utilities for the LCMAPS.

Author:
Martijn Steenbakkers for the EU DataGrid.

Definition in file lcmaps_utils.c.

8.32.2 Function Documentation

8.32.2.1 char ∗ cred_to_dn (gss_cred_id_t *globus_cred*) [static]

Get the globus DN from GLOBUS credential (gssapi).

(copied and modified from GLOBUS gatekeeper.c)

Parameters:
globus_cred GLOBUS credential

Returns:
globus DN string (which may be freed)
For internal use only.

Definition at line 176 of file lcmaps_utils.c.

Referenced by lcmaps_fill_cred().

8.32.2.2 int fexist (char ∗ *path*) [static]

check the existence of file corresponding to <path>.

Parameters:
path absolute filename to be checked.

Return values:
1 file exists.
0 failure.

Definition at line 304 of file lcmaps_utils.c.

Referenced by lcmaps_getfexist().

8.33 lcmaps_utils.h File Reference

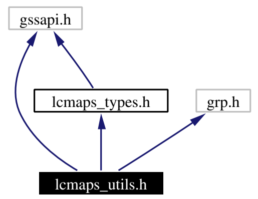
API for the utilities for the LCMAPS.

`#include <gssapi.h>`

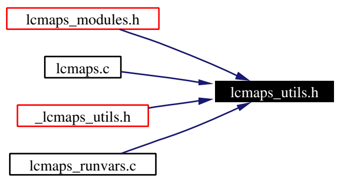
`#include "lcmaps_types.h"`

`#include <grp.h>`

Include dependency graph for lcmaps_utils.h:



This graph shows which files directly or indirectly include this file:



CREDENTIAL FUNCTIONS

- char∗ lcmaps_get_dn (lcmaps_cred_id_t lcmaps_credential)

  Retrieve user DN from (LCMAPS) credential.

FILENAME FUNCTIONS

- char∗ lcmaps_genfilename (char ∗prefix, char ∗path, char ∗suffix)

  Generate an absolute file name.

- char∗ lcmaps_getfexist (int n,...)

  Picks the first existing file in argument list.

- char∗ lcmaps_findfile (char ∗name)

  Checks for file in standard directories.

Functions

- int lcmaps_get_gidlist (const char ∗username, int ∗ngroups, gid_t ∗∗group_list)
  Finds the list of gids for user in the group file (/etc/group).

8.33.1 Detailed Description

API for the utilities for the LCMAPS.

Author:
Martijn Steenbakkers for the EU DataGrid.

This header contains the declarations of the LCMAPS utility functions:

1. lcmaps_get_dn():
2. lcmaps_genfilename():
3. lcmaps_getfexist():
4. lcmaps_findfile():
5. lcmaps_findfile():
6. lcmaps_get_gidlist():

Definition in file lcmaps_utils.h.

8.33.2 Function Documentation

8.33.2.1 char ∗ lcmaps_findfile (char ∗ *name*)

Checks for file in standard directories.

The directories that are checked are:

- current directory
- "modules"
- LCMAPS_ETC_HOME
- LCMAPS_MOD_HOME
- LCMAPS_LIB_HOME

Parameters:
name string containing the file name

Returns:
pointer to a string containing the absolute path to the file, which has to be freed or NULL.

Definition at line 382 of file lcmaps_utils.c.

8.33.2.2 char ∗ lcmaps_genfilename (char ∗ *prefixp*, char ∗ *pathp*, char ∗ *suffixp*)

Generate an absolute file name.

Given a starting prefix, a relative or absolute path, and a suffix an absolute file name is generated. Uses the prefix only if the path is relative. (Copied (and modified) from GLOBUS gatekeeper.c)

Parameters:

prefix string containing the prefix to be prepended.

path relative/absolute path to file name.

suffix string containing the suffix to be appended.

Returns:

pointer to a string containing the absolute path to the file, which has to be freed.

Definition at line 247 of file lcmaps_utils.c.

8.33.2.3 char ∗ lcmaps_get_dn (lcmaps_cred_id_t *lcmaps_cred*)

Retrieve user DN from (LCMAPS) credential.

This function takes an LCMAPS credential as input and returns the corresponding user distinguished name (DN).

(Internal:) If the GLOBUS credential part of the LCMAPS credential is empty the user DN is already included in the LCMAPS credential.

Parameters:

lcmaps_credential the LCMAPS credential

Returns:

a string containing the user DN

Definition at line 151 of file lcmaps_utils.c.

8.33.2.4 int lcmaps_get_gidlist (const char ∗ *username*, int ∗ *ngroups*, gid_t ∗∗ *group_list*)

Finds the list of gids for user in the group file (/etc/group).

Returns a list of gid_t which should be freed by calling program.

Parameters:

username the name of the user

ngroups ptr to int which will be filled with the number of gids

group_list ptr to an array of gid_t

Return values:

0 success

-1 realloc failure

-2 getgrent failure

1 failure

Definition at line 566 of file lcmaps_utils.c.

8.33.2.5 char * lcmaps_getfexist (int *n*, ...)

Picks the first existing file in argument list.

Parameters:

- ***n*** the number of paths presented in the following argument list.
- **...** variable argument list of paths.

Returns:

filename found or NULL

Definition at line 340 of file lcmaps_utils.c.

8.34 lcmaps_vo_data.c File Reference

LCMAPS utilities for creating and accessing VO data structures.

`#include <stdio.h>`

`#include <stdlib.h>`

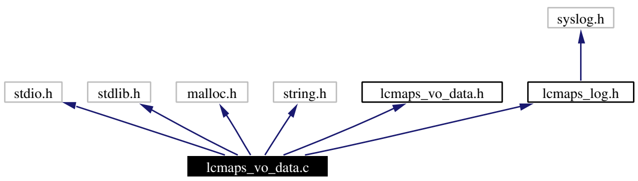
`#include <malloc.h>`

`#include <string.h>`

`#include "lcmaps_vo_data.h"`

`#include "lcmaps_log.h"`

Include dependency graph for lcmaps_vo_data.c:



8.34.1 Detailed Description

LCMAPS utilities for creating and accessing VO data structures.

Author:

Martijn Steenbakkers for the EU DataGrid.

The interface is composed of:

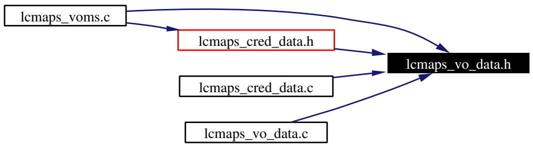
1. lcmaps_createVoData(): create a VoData structure
2. lcmaps_deleteVoData(): delete a VoData structure
3. lcmaps_copyVoData(): copy (the contents of) a VoData structure
4. lcmaps_printVoData(): print the contents of a VoData structure
5. lcmaps_stringVoData(): cast a VoData structure into a string

Definition in file lcmaps_vo_data.c.

8.35 lcmaps_vo_data.h File Reference

LCMAPS module for creating and accessing VO data structures.

This graph shows which files directly or indirectly include this file:



Data Structures

- struct lcmaps_vo_data_s

  structure that contains the VO information found in the user's gss credential.

Functions

- lcmaps_vo_data_t* lcmaps_createVoData (const char *vo, const char *group, const char *subgroup, const char *role, const char *capability)

  Create a VoData structure.

- int lcmaps_deleteVoData (lcmaps_vo_data_t **vo_data)

  Delete a VoData structure.

- int lcmaps_cleanVoData (lcmaps_vo_data_t *vo_data)

  Clean a VoData structure.

- int lcmaps_copyVoData (lcmaps_vo_data_t *dst_vo_data, const lcmaps_vo_data_t *src_vo_data)

  Copy a VoData structure into an empty VoData structure.

- int lcmaps_printVoData (int debug_level, const lcmaps_vo_data_t *vo_data)

  Print the contents of a VoData structure.

- int lcmaps_stringVoData (const lcmaps_vo_data_t *vo_data, char *buffer, int nchars)

  Cast a VoData structure into a string.

8.35.1 Detailed Description

LCMAPS module for creating and accessing VO data structures.

Author:

Martijn Steenbakkers for the EU DataGrid.

The interface is composed of:

1. lcmaps_createVoData(): create a VoData structure
2. lcmaps_deleteVoData(): delete a VoData structure
3. lcmaps_copyVoData(): copy (the contents of) a VoData structure
4. lcmaps_printVoData(): print the contents of a VoData structure
5. lcmaps_stringVoData(): cast a VoData structure into a string

Definition in file lcmaps_vo_data.h.

8.35.2 Function Documentation

8.35.2.1 int lcmaps_cleanVoData (lcmaps_vo_data_t ∗ *vo_data*)

Clean a VoData structure.

Clean a VoData structure that was previously filled with lcmaps_copyVoData(). The contents are freed and set to zero.

Parameters:

vo_data a pointer to a VoData structure

Return values:

0 in case of success

-1 in case of failure

Definition at line 192 of file lcmaps_vo_data.c.

8.35.2.2 int lcmaps_copyVoData (lcmaps_vo_data_t ∗ *dst_vo_data*, const lcmaps_vo_data_t ∗ *src_vo_data*)

Copy a VoData structure into an empty VoData structure.

Copy a VoData structure into an empty VoData structure which has to exist.

Parameters:

dst_vo_data pointer to a empty VoData structure that should be filled

src_vo_data pointer to the VoData structure that should be copied

Return values:

0 success

-1 failure (either src_vo_data or dst_vo_data was empty)

Definition at line 260 of file lcmaps_vo_data.c.

8.35.2.3 lcmaps_vo_data_t * lcmaps_createVoData (const char * *vo*, const char * *group*, const char * *subgroup*, const char * *role*, const char * *capability*)

Create a VoData structure.

Create a VoData structure (store a VO, group, (subgroup,) role, capability combination). Allocate the memory. To be freed with lcmaps_deleteVoData().

Parameters:

vo name of the VO

group name of the group

subgroup name of the subgroup (ignored for the moment)

role the role

capability the capability (whatever it is)

Returns:

pointer to the VoData structure or NULL

Definition at line 78 of file lcmaps_vo_data.c.

8.35.2.4 int lcmaps_deleteVoData (lcmaps_vo_data_t ** *vo_data*)

Delete a VoData structure.

Delete a VoData structure that was previously created with lcmaps_createVoData(). The pointer to the VoData structure is finally set to NULL;

Parameters:

vo_data pointer to a pointer to a VoData structure

Return values:

0 in case of success

-1 in case of failure

Definition at line 138 of file lcmaps_vo_data.c.

8.35.2.5 int lcmaps_printVoData (int *debug_level*, const lcmaps_vo_data_t * *vo_data*)

Print the contents of a VoData structure.

Parameters:

vo_data pointer to a VoData structure

debug_level debug_level for which the contents will be printed

Returns:

0 (always)

Definition at line 323 of file lcmaps_vo_data.c.

8.35.2.6 int lcmaps_stringVoData (const lcmaps_vo_data_t * *vo_data*, char * *buffer*, int *nchars*)

Cast a VoData structure into a string.

The user of this function should create the buffer of size nchars beforehand. In buffer a string like the following will be written: "/VO=fred/GROUP=fred/flintstone/ROLE=director/CAPABILITY=destroy"

Currently the SUBGROUP entry is ignored. Only if the information is present in the VoData structure, it is added to the string. Both data for VO and GROUP are required (might change).

Parameters:

vo_data pointer to a VoData structure

buffer pointer to character array of size nchars

nchars size of character array

Return values:

0 in case of success

-1 in case of failure

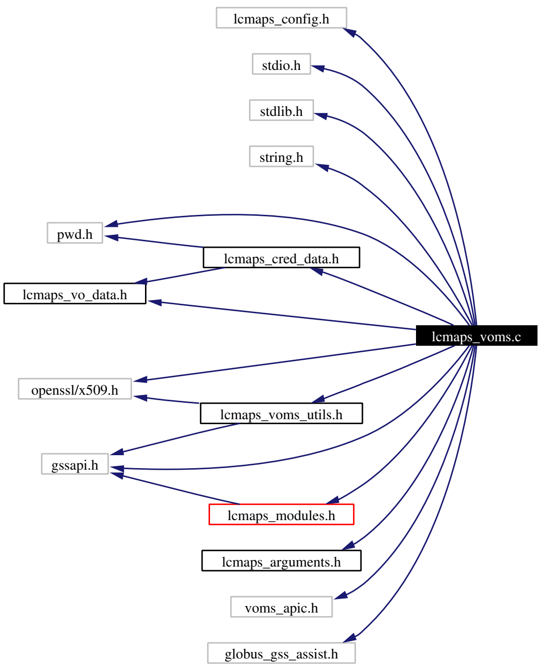
Definition at line 391 of file lcmaps_vo_data.c.

8.36 lcmaps_voms.c File Reference

Interface to the LCMAPS plugins.

```
#include "lcmaps_config.h"
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <pwd.h>
#include <openssl/x509.h>
#include "gssapi.h"
#include "lcmaps_modules.h"
#include "lcmaps_arguments.h"
#include "lcmaps_cred_data.h"
#include "lcmaps_voms_utils.h"
#include "lcmaps_vo_data.h"
#include "voms_apic.h"
#include "globus_gss_assist.h"
```

Include dependency graph for lcmaps_voms.c:



8.36.1 Detailed Description

Interface to the LCMAPS plugins.

Author:

Martijn Steenbakkers for the EU DataGrid.

This file contains the code for the voms plugin (extracts the VOMS info from the certificate). The interface consists of the following functions:

1. plugin_initialize()
2. plugin_run()
3. plugin_terminate()
4. plugin_introspect()

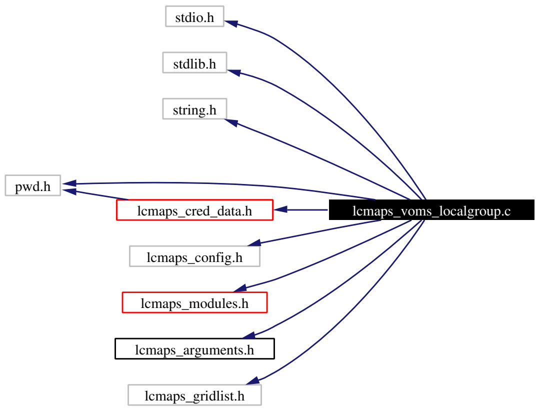
Definition in file lcmaps_voms.c.

8.37 lcmaps_voms_localgroup.c File Reference

Interface to the LCMAPS plugins.

```
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <pwd.h>
#include "lcmaps_config.h"
#include "lcmaps_modules.h"
#include "lcmaps_arguments.h"
#include "lcmaps_cred_data.h"
#include "lcmaps_gridlist.h"
```

Include dependency graph for lcmaps_voms_localgroup.c:



8.37.1 Detailed Description

Interface to the LCMAPS plugins.

Author:

Martijn Steenbakkers for the EU DataGrid.

This file contains the code of the voms_localgroup plugin

1. plugin_initialize()
2. plugin_run()
3. plugin_terminate()
4. plugin_introspect()

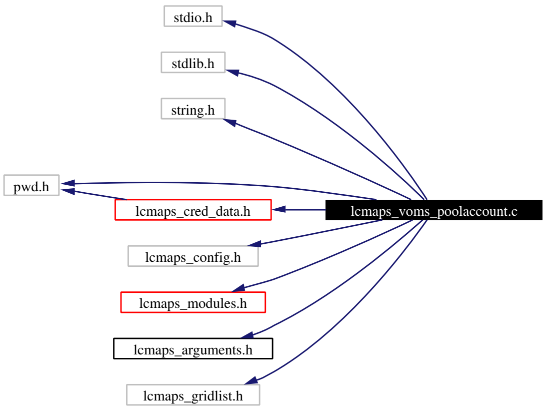
Definition in file lcmaps_voms_localgroup.c.

8.38 lcmaps_voms_poolaccount.c File Reference

Interface to the LCMAPS plugins.

```
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <pwd.h>
#include "lcmaps_config.h"
#include "lcmaps_modules.h"
#include "lcmaps_arguments.h"
#include "lcmaps_cred_data.h"
#include "lcmaps_gridlist.h"
```

Include dependency graph for lcmaps_voms_poolaccount.c:



8.38.1 Detailed Description

Interface to the LCMAPS plugins.

Author:

Martijn Steenbakkers for the EU DataGrid.

This file contains the code of the voms_poolaccount plugin

1. plugin_initialize()
2. plugin_run()
3. plugin_terminate()
4. plugin_introspect()

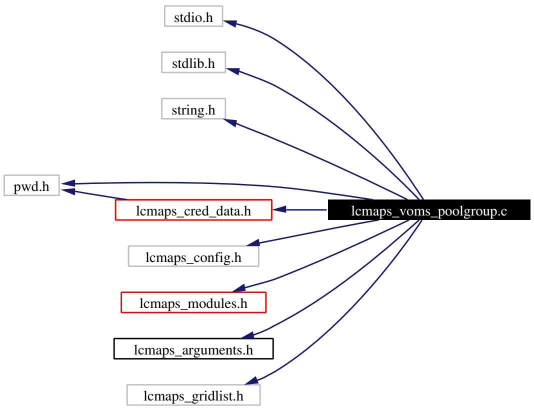
Definition in file lcmaps_voms_poolaccount.c.

8.39 lcmaps_voms_poolgroup.c File Reference

Interface to the LCMAPS plugins.

```
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <pwd.h>
#include "lcmaps_config.h"
#include "lcmaps_modules.h"
#include "lcmaps_arguments.h"
#include "lcmaps_cred_data.h"
#include "lcmaps_gridlist.h"
```

Include dependency graph for lcmaps_voms_poolgroup.c:



8.39.1 Detailed Description

Interface to the LCMAPS plugins.

Author:
Martijn Steenbakkers for the EU DataGrid.

This file contains the code of the voms_poolgroup plugin

1. plugin_initialize()
2. plugin_run()
3. plugin_terminate()
4. plugin_introspect()

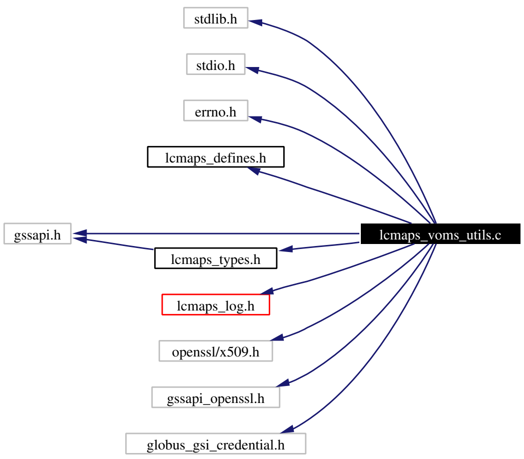
Definition in file lcmaps_voms_poolgroup.c.

8.40 lcmaps_voms_utils.c File Reference

the utilities for the LCMAPS voms plugin.

```
#include <stdlib.h>
#include <stdio.h>
#include <errno.h>
#include "lcmaps_defines.h"
#include "lcmaps_types.h"
#include "lcmaps_log.h"
#include <openssl/x509.h>
#include <gssapi.h>
#include "gssapi_openssl.h"
#include "globus_gsi_credential.h"
```

Include dependency graph for lcmaps_voms_utils.c:



Functions

- X509∗ lcmaps_cred_to_x509 (gss_cred_id_t cred)

  Return the pointer to X509 structure from gss credential.

8.40.1 Detailed Description

the utilities for the LCMAPS voms plugin.

Author:

Martijn Steenbakkers for the EU DataGrid.

This header contains the definitions of the LCMAPS utility functions:

1. lcmaps_cred_to_x509():
2. lcmaps_cred_to_x509_chain():

Definition in file lcmaps_voms_utils.c.

8.40.2 Function Documentation

8.40.2.1 X509 ∗ lcmaps_cred_to_x509 (gss_cred_id_t *cred*)

Return the pointer to X509 structure from gss credential.

This function takes a gss credential as input and returns the corresponding X509 structure, which is allocated for this purpose (should be freed)

Parameters:
cred the gss credential

Returns:
a pointer to a X509 struct or NULL

Definition at line 85 of file lcmaps_voms_utils.c.

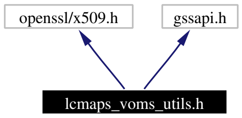
8.41 lcmaps_voms_utils.h File Reference

API for the utilities for the LCMAPS voms plugin.

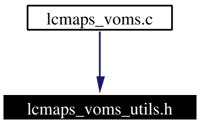
`#include <openssl/x509.h>`

`#include <gssapi.h>`

Include dependency graph for lcmaps_voms_utils.h:



This graph shows which files directly or indirectly include this file:



8.41.1 Detailed Description

API for the utilities for the LCMAPS voms plugin.

Author:
Martijn Steenbakkers for the EU DataGrid.

This header contains the declarations of the LCMAPS utility functions:

1. lcmaps_cred_to_x509():
2. lcmaps_cred_to_x509_chain():

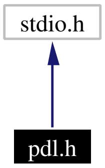
Definition in file lcmaps_voms_utils.h.

8.42 pdl.h File Reference

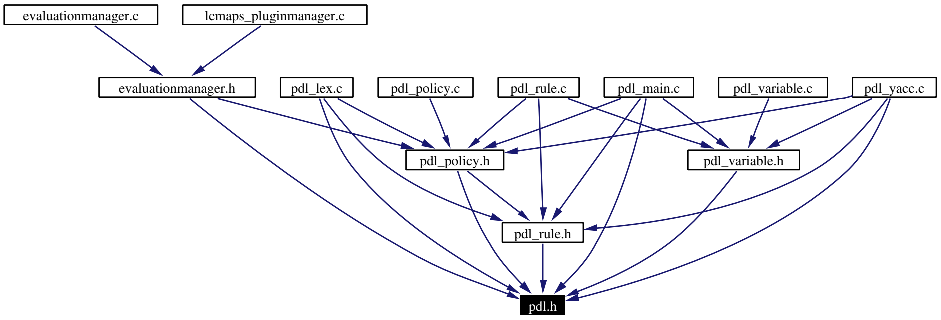
General include file.

`#include <stdio.h>`

Include dependency graph for pdl.h:



This graph shows which files directly or indirectly include this file:



Data Structures

- struct plugin_s

  Structure holds a plugin name and its arguments, as well as the line number the plugin is first mentioned.

- struct record_s

  Structure is used to keep track of strings and the line they appear on.

Defines

- #define TRUE 1

Typedefs

- typedef struct record_s record_t

  Structure is used to keep track of strings and the line they appear on.

- typedef struct plugin_s plugin_t

    Structure holds a plugin name and its arguments, as well as the line number the plugin is first mentioned.

Enumerations

- enum pdl_error_t { PDL_UNKNOWN, PDL_INFO, PDL_WARNING, PDL_ERROR, PDL_SAME }
- enum plugin_status_t { EVALUATION_START, EVALUATION_SUCCESS, EVALUATION_FAILURE }

Functions

- int pdl_init (const char ∗name)
- const char∗ pdl_path (void)
- int yyparse_errors (void)
- int yyerror (const char ∗)
- const char∗ pdl_next_plugin (plugin_status_t status)
- void set_path (record_t ∗_path)
- record_t∗ concat_strings (record_t ∗s1, record_t ∗s2)
- const plugin_t∗ get_plugins (void)
- void warning (pdl_error_t error, const char ∗s,...)

Variables

- unsigned int lineno = 1

    The first line of a configuration sctipt is labeled 1.

8.42.1 Detailed Description

General include file.

In this include file all general "things" can be found.

Author:

G.M. Venekamp (venekamp@nikhef.nl)

Version:

Revision:

1.10

Date:

Date:

2003/07/15 11:38:06

Definition in file pdl.h.

8.42.2 Define Documentation

8.42.2.1 #define TRUE 1

The evaluation manager defines its own boolean type. It first undefines any existing type defenitions before it defines it itself.

Definition at line 44 of file pdl.h.

8.42.3 Typedef Documentation

8.42.3.1 typedef struct plugin_s plugin_t

Structure holds a plugin name and its arguments, as well as the line number the plugin is first mentioned.

8.42.3.2 typedef struct record_s record_t

Structure is used to keep track of strings and the line they appear on.

When lex finds a match, this structure is used to keep track of the relevant information. The matchig string as well as the line number are saved. The line number can be used for later references when an error related to the symbol has occured. This allows for easier debugging of the configuration file.

8.42.4 Enumeration Type Documentation

8.42.4.1 enum pdl_error_t

Different levels of error logging.

Enumeration values:

PDL_UNKNOWN Unknown error level.

PDL_INFO Informational level.

PDL_WARNING Warning level.

PDL_ERROR Error level.

PDL_SAME Repeat the previous level.

Definition at line 52 of file pdl.h.

8.42.4.2 enum plugin_status_t

Guide the selection of the next plugin.

Enumeration values:

EVALUATION_START The evaluation process has just started.

EVALUATION_SUCCESS The evaluation of the plugin was successful.

EVALUATION_FAILURE The evaluation of the plugin was unsuccessfyl.

Definition at line 65 of file pdl.h.

8.42.5 Function Documentation

8.42.5.1 record_t * concat_strings (record_t * *s1*, record_t * *s2*)

Concatenate two strings. The orginal two strings are freed. When the concatenation fails, the origial strings are still freed. The actual concatenation is done by _concat_strings().

Parameters:

s1 First string.

s2 Second string

Returns:

Concatenated strings of s1 + s2.

Definition at line 481 of file pdl_main.c.

8.42.5.2 const plugin_t * get_plugins (void)

Get a list of plugins as known by the configuration file.

Returns:

Plugin list (linked list).

Definition at line 130 of file pdl_main.c.

Referenced by getPluginNameAndArgs().

8.42.5.3 int pdl_init (const char * *name*)

Init the pdl engine. The function takes one arguments, the name of a configuration file to use.

Parameters:

name Name of the configuration file to use.

Returns:

0 in case the initialization is successful; -1 in case of not being successful.

Definition at line 72 of file pdl_main.c.

8.42.5.4 const char * pdl_next_plugin (plugin_status_t *status*)

Find the next plugin to evaluate based on the return status of the previous plugin evaluation. There are three statuses, two of which are rather obvious: either the previous evaluation has succeeded (EVALUATION_SUCCESS), or it has failed (EVALUATION_FAILURE). Based on these results, the next plugin should be the true_branch or false_branch respectively. There is one situation where there is no previous evaluation and that is at the very beginning. The very first call to this function should have (EVALUATION_START) as arguments. In this case the current state of the rule is returned as the next plugin to evaluate.

Parameters:

status Status of previous evaluation.

Returns:

plugin name to be evaluation according to the configuration file.

Definition at line 316 of file pdl_main.c.

8.42.5.5 const char * pdl_path (void)

Get the path.

Returns:

Path.

Definition at line 394 of file pdl_main.c.

Referenced by getPluginNameAndArgs(), and pdl_next_plugin().

8.42.5.6 void set_path (record_t * *path*)

Function is called when the parser has found the value of the reserved path word. This function acts as a wrapper for the _set_path() function.

Parameters:

path

Definition at line 424 of file pdl_main.c.

8.42.5.7 void warning (pdl_error_t *error*, const char * *s*, ...)

Display a warning message.

Parameters:

error Severity of the error.

s The text string.

... Additional values; much like printf(char *, ...);

Definition at line 562 of file pdl_main.c.

8.42.5.8 int yyerror (const char * *s*)

When yacc encounters an error during the parsing process of the configuration file, it calls yyerror(). The actual message formatting is done in waring();

Parameters:

s error string.

Definition at line 407 of file pdl_main.c.

8.42.5.9 int yyparse_errors (void)

Tell if there were errors/warning during parsing.

Returns:

0, if the are no errors/warnings, -1 otherwise.

Definition at line 118 of file pdl_main.c.

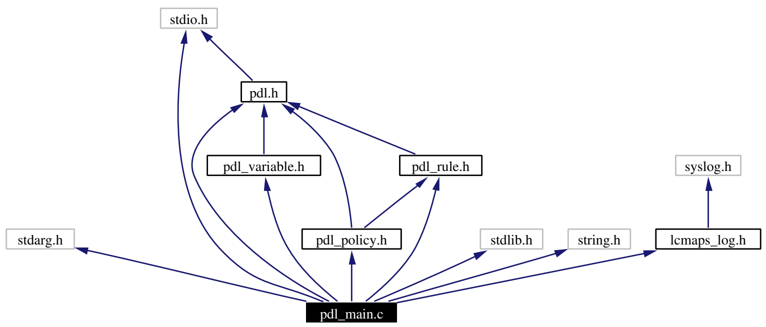
Referenced by startEvaluationManager().

8.43 pdl_main.c File Reference

All functions that do not fit elsewhere can be found here.

```
#include <stdarg.h>
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include "lcmaps_log.h"
#include "pdl.h"
#include "pdl_variable.h"
#include "pdl_policy.h"
#include "pdl_rule.h"
```

Include dependency graph for pdl_main.c:



Functions

- void _set_path (const record_t ∗_path)
- record_t∗ _concat_strings (const record_t ∗s1, const record_t ∗s2)
- void reduce_policies (void)
- BOOL plugin_exists (const char ∗string)
- int pdl_init (const char ∗name)
- int yyparse_errors (void)
- const plugin_t∗ get_plugins (void)
- int find_first_space (const char ∗string)
- const char∗ pdl_next_plugin (plugin_status_t status)
- const char∗ pdl_path (void)
- int yyerror (const char ∗s)
- void set_path (record_t ∗path)
- void free_path (void)
- record_t∗ concat_strings (record_t ∗s1, record_t ∗s2)
- void free_resources (void)
- void warning (pdl_error_t error, const char ∗s,...)

Variables

- const char* script_name = NULL

  If non NULL, the name of the configuration script.

- const char* d_path = "/usr/lib"

  Default path where plugins can be found.

- const char* path = 0

  Path where plugins can be found.

- int path_lineno = 0

  ???

- plugin_t* top_plugin = NULL

  First node of the list.

- BOOL default_path = TRUE

  Has the default vallue of the path been changed.

- BOOL parse_error = FALSE

  Tell if there have been any error during parsing.

- char* level_str [PDL_SAME]

  When a message is printed, how do we spell warning in a given language.

- unsigned int lineno = 1

  The first line of a configuration sctipt is labeled 1.

8.43.1 Detailed Description

All functions that do not fit elsewhere can be found here.

In here one can find the more general functions. Most of them are accessible to outside sources. For a complete list of usable function to out side sources,

See also:
pdl.h.

Author:
G.M. Venekamp (venekamp@nikhef.nl)

Version:

Revision:
1.25

Date:

Date:
2003/07/16 09:30:57

Definition in file pdl_main.c.

8.43.2 Function Documentation

8.43.2.1 record_t ∗ _concat_strings (const record_t ∗ *s1*, const record_t ∗ *s2*)

Concatenate two string.

Parameters:
- ***s1*** first half of the string.
- ***s2*** second half of the string.

Returns:
new string which is the concatenation of s1 and s2.

Definition at line 502 of file pdl_main.c.

Referenced by concat_strings().

8.43.2.2 void _set_path (const record_t ∗ *_path*)

Overwrite the default path with the new value. If this function is called more than once, a warning message is displayed for each occurent.

Parameters:
- ***_path*** The new path.

Definition at line 441 of file pdl_main.c.

Referenced by set_path().

8.43.2.3 record_t∗ concat_strings (record_t ∗ *s1*, record_t ∗ *s2*)

Concatenate two strings. The orginal two strings are freed. When the concatenation fails, the origial strings are still freed. The actual concatenation is done by _concat_strings().

Parameters:
- ***s1*** First string.
- ***s2*** Second string

Returns:
Concatenated strings of s1 + s2.

Definition at line 481 of file pdl_main.c.

8.43.2.4 int find_first_space (const char * *string*)

Find the first occurrence of a space in a string.

Parameters:
string String where the first space needs to be found.

Returns:
Position of the first occurrence of the space. If no space could be found, the position is set to the length of the string.

Definition at line 287 of file pdl_main.c.

Referenced by plugin_exists().

8.43.2.5 void free_path (void)

Free the string allocated to hold the path

Definition at line 460 of file pdl_main.c.

Referenced by free_resources().

8.43.2.6 void free_resources (void)

Free the resources.

Definition at line 526 of file pdl_main.c.

Referenced by stopEvaluationManager().

8.43.2.7 const plugin_t* get_plugins (void)

Get a list of plugins as known by the configuration file.

Returns:
Plugin list (linked list).

Definition at line 130 of file pdl_main.c.

8.43.2.8 int pdl_init (const char * *name*)

Init the pdl engine. The function takes one arguments, the name of a configuration file to use.

Parameters:
name Name of the configuration file to use.

Returns:
0 in case the initialization is successful; -1 in case of not being successful.

Definition at line 72 of file pdl_main.c.

Referenced by startEvaluationManager().

8.43.2.9 const char* pdl_next_plugin (plugin_status_t *status*)

Find the next plugin to evaluate based on the return status of the previous plugin evaluation. There are three statuses, two of which are rather obvious: either the previous evaluation has succeeded (EVALUATION_SUCCESS), or it has failed (EVALUATION_FAILURE). Based on these results, the next plugin should be the true_branch or false_branch respectively. There is one situation where there is no previous evaluation and that is at the very beginning. The very first call to this function should have (EVALUATION_START) as arguments. In this case the current state of the rule is returned as the next plugin to evaluate.

Parameters:

status Status of previous evaluation.

Returns:

plugin name to be evaluation according to the configuration file.

Definition at line 316 of file pdl_main.c.

Referenced by runEvaluationManager().

8.43.2.10 const char* pdl_path (void)

Get the path.

Returns:

Path.

Definition at line 394 of file pdl_main.c.

8.43.2.11 BOOL plugin_exists (const char * *string*)

Check if a plugin as specified by the string argument exists.

Parameters:

string Name of the plugin.

Returns:

TRUE if the plugin exists, FALSE otherwise.

Definition at line 185 of file pdl_main.c.

8.43.2.12 void reduce_policies (void)

Reduce_policies to its elemantry form, i.e. each policy has a list of rules which need to be reduced.

Definition at line 187 of file pdl_policy.c.

Referenced by startEvaluationManager().

8.43.2.13 void set_path (record_t * *path*)

Function is called when the parser has found the value of the reserved path word. This function acts as a wrapper for the _set_path() function.

Parameters:

path

Definition at line 424 of file pdl_main.c.

8.43.2.14 void warning (pdl_error_t *error*, const char * *s*, ...)

Display a warning message.

Parameters:

error Severity of the error.

s The text string.

... Additional values; much like printf(char *, ...);

Definition at line 562 of file pdl_main.c.

Referenced by _add_policy(), _add_rule(), _add_variable(), _concat_strings(), _set_path(), pdl_init(), and yyerror().

8.43.2.15 int yyerror (const char * *s*)

When yacc encounters an error during the parsing process of the configuration file, it calls yyerror(). The actual message formatting is done in waring();

Parameters:

s error string.

Definition at line 407 of file pdl_main.c.

8.43.2.16 int yyparse_errors (void)

Tell if there were errors/warning during parsing.

Returns:

0, if the are no errors/warnings, -1 otherwise.

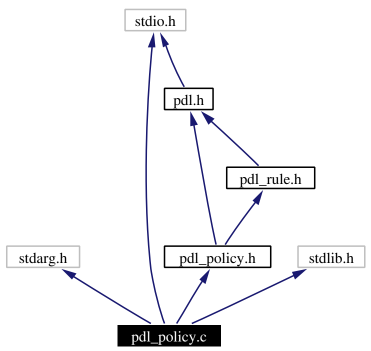
Definition at line 118 of file pdl_main.c.

8.44 pdl_policy.c File Reference

Implementation of the pdl policies.

```
#include <stdarg.h>
#include <stdio.h>
#include <stdlib.h>
#include "pdl_policy.h"
```

Include dependency graph for pdl_policy.c:



Functions

- BOOL _add_policy (const record_t ∗name, const rule_t ∗rules)
- policy_t∗ current_policy (void)
- void allow_rules (BOOL allow)
- void add_policy (record_t ∗policy, rule_t ∗rules)
- void remove_policy (record_t ∗policy)
- policy_t∗ find_policy (const char ∗name)
- void reduce_policies (void)
- policy_t∗ get_policies (void)
- void show_policies (void)
- void free_policies (void)
- BOOL policies_have_been_reduced (void)

Variables

- BOOL policies_reduced = FALSE

  Tell if reduce_policy() has been called.

8.44.1 Detailed Description

Implementation of the pdl policies.

Author:
G.M. Venekamp (venekamp@nikhef.nl)

Version:

Revision:
1.10

Date:

Date:
2003/07/16 09:30:57

Definition in file pdl_policy.c.

8.44.2 Function Documentation

8.44.2.1 BOOL _add_policy (const record_t * *name*, const rule_t * *rules*)

Add a policy with its rules to the list of policies.

Before the policy name is actually added to list of policies, a check is made to see weather or not a policy by the same name exists. if it does, the policy name will not be added and an error message is displayed, letting the user know that the configuration file contains multiple policy rules with the same name.

Parameters:
name Name of the new policy.
rules List of associated rules for the policy.

Returns:
TRUE, If the policy has been added successfully; FALSE otherwise.

Definition at line 117 of file pdl_policy.c.

Referenced by add_policy().

8.44.2.2 void add_policy (record_t * *policy*, rule_t * *rules*)

Wrapper around the _add_policy(name) function.

When the _add_policy() call fails, this function cleans up the data structure allocated for holding information about the policy that was found. See _add_policy() for information about the kind of reasons it can fail.

Parameters:

name Name of the policy.

rules List of associated rules for the policy.

Definition at line 83 of file pdl_policy.c.

8.44.2.3 void allow_rules (BOOL *allow*)

Allow or disallow the additions of rules depending on the argument. When for example a policy is defined for the second time, an error should be generated, but the parsing should still continue. However, no rules can be added to the policy as there is currently no policy defined.

Parameters:

allow TRUE if addition of new rules is allowd, FALSE otherwise.

Definition at line 65 of file pdl_policy.c.

8.44.2.4 policy_t ∗ current_policy (void)

Return the current policy.

Returns:

Current policy.

Definition at line 49 of file pdl_policy.c.

8.44.2.5 policy_t ∗ find_policy (const char ∗ *name*)

Find a policy based.

Parameters:

name Name of the policy to be found. \retrun The policy if a polict with name 'name' exists, 0 otherwise.

Definition at line 170 of file pdl_policy.c.

8.44.2.6 void free_policies (void)

Free all policies and their allocated resources.

Definition at line 239 of file pdl_policy.c.

Referenced by free_resources().

8.44.2.7 policy_t ∗ get_policies (void)

Get the list of policies.

Returns:

First policy in the list.

Definition at line 212 of file pdl_policy.c.

Referenced by get_plugins(), pdl_next_plugin(), and reduce_policies().

8.44.2.8 BOOL policies_have_been_reduced (void)

Tell if the reduce_policy() call has been called.

Returns:
TRUE if reduce_policy() has been called; FALSE otherwise.

Definition at line 261 of file pdl_policy.c.

Referenced by get_plugins().

8.44.2.9 void reduce_policies (void)

Reduce_policies to its elemantry form, i.e. each policy has a list of rules which need to be reduced.

Definition at line 187 of file pdl_policy.c.

8.44.2.10 void remove_policy (record_t * *name*)

Remove a policy from the list of policies and free all associated resources of the policy.

Parameters:
name Policy to be removed.

Definition at line 156 of file pdl_policy.c.

8.44.2.11 void show_policies (void)

Display the policies and the rules associated with the policy.

Definition at line 222 of file pdl_policy.c.

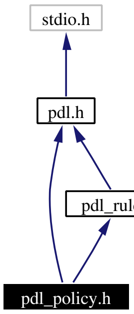
8.45 pdl_policy.h File Reference

Include file for using the pdl policies.

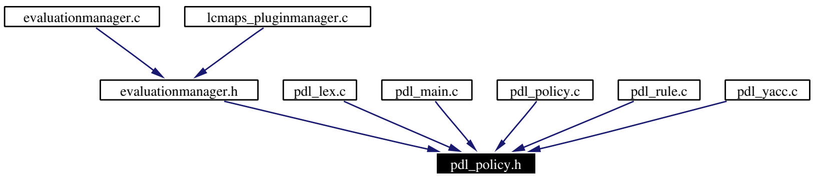
`#include "pdl.h"`

`#include "pdl_rule.h"`

Include dependency graph for pdl_policy.h:



This graph shows which files directly or indirectly include this file:



Data Structures

- struct policy_s

  Keeping track of found policies.

Typedefs

- typedef struct policy_s policy_t

  Keeping track of found policies.

Functions

- void add_policy (record_t *policy, rule_t *rules)
- void remove_policy (record_t *name)

- void show_policies (void)
- void free_policies (void)
- void allow_rules (BOOL allow)
- policy_t* find_policy (const char *name)
- policy_t* current_policy (void)
- policy_t* get_policies (void)

8.45.1 Detailed Description

Include file for using the pdl policies.

Author:
G.M. Venekamp (venekamp@nikhef.nl)

Version:

Revision:
1.6

Date:

Date:
2003/07/14 07:59:14

Definition in file pdl_policy.h.

8.45.2 Typedef Documentation

8.45.2.1 typedef struct policy_s policy_t

Keeping track of found policies.

8.45.3 Function Documentation

8.45.3.1 void add_policy (record_t * *policy*, rule_t * *rules*)

Wrapper around the _add_policy(name) function.

When the _add_policy() call fails, this function cleans up the data structure allocated for holding information about the policy that was found. See _add_policy() for information about the kind of reasons it can fail.

Parameters:
name Name of the policy.
rules List of associated rules for the policy.

Definition at line 83 of file pdl_policy.c.

8.45.3.2 void allow_rules (BOOL *allow*)

Allow or disallow the additions of rules depending on the argument. When for example a policy is defined for the second time, an error should be generated, but the parsing should still continue. However, no rules can be added to the policy as there is currently no policy defined.

Parameters:

allow TRUE if addition of new rules is allowd, FALSE otherwise.

Definition at line 65 of file pdl_policy.c.

Referenced by _add_policy().

8.45.3.3 policy_t* current_policy (void)

Return the current policy.

Returns:

Current policy.

Definition at line 49 of file pdl_policy.c.

8.45.3.4 policy_t* find_policy (const char * *name*)

Find a policy based.

Parameters:

name Name of the policy to be found. \retrun The policy if a polict with name 'name' exists, 0 otherwise.

Definition at line 170 of file pdl_policy.c.

Referenced by _add_policy(), and _add_rule().

8.45.3.5 void free_policies (void)

Free all policies and their allocated resources.

Definition at line 239 of file pdl_policy.c.

8.45.3.6 policy_t* get_policies (void)

Get the list of policies.

Returns:

First policy in the list.

Definition at line 212 of file pdl_policy.c.

8.45.3.7 void remove_policy (record_t * *name*)

Remove a policy from the list of policies and free all associated resources of the policy.

Parameters:

name Policy to be removed.

Definition at line 156 of file pdl_policy.c.

8.45.3.8 void show_policies (void)

Display the policies and the rules associated with the policy.

Definition at line 222 of file pdl_policy.c.

8.46 pdl_rule.c File Reference

Implementation of the pdl rules.

`#include <stdarg.h>`

`#include <stdio.h>`

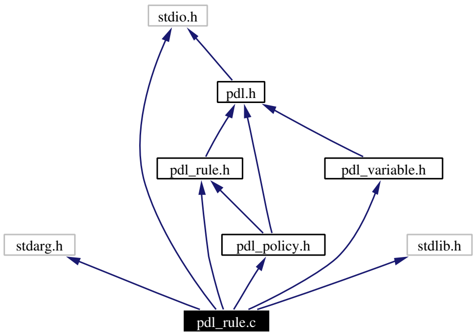
`#include <stdlib.h>`

`#include "pdl_rule.h"`

`#include "pdl_policy.h"`

`#include "pdl_variable.h"`

Include dependency graph for pdl_rule.c:



Functions

- rule_t∗ _add_rule (const record_t ∗state, const record_t ∗true_branch, const record_t ∗false_branch)
- const rule_t∗ find_state (const rule_t ∗rule, const char ∗state)
- void start_new_rules (void)
- void allow_new_rules (BOOL allow)
- rule_t∗ add_rule (record_t ∗state, record_t ∗true_branch, record_t ∗false_branch)
- void reduce_rule (rule_t ∗rule)
- void show_rules (const rule_t ∗rule)
- void free_rules (rule_t ∗rule)
- rule_t∗ get_top_rule (void)

8.46.1 Detailed Description

Implementation of the pdl rules.

Author:
G.M. Venekamp (venekamp@nikhef.nl)

Version:

Revision:
1.12

Date:

Date:
2003/07/16 09:30:57

Definition in file pdl_rule.c.

8.46.2 Function Documentation

8.46.2.1 rule_t * _add_rule (const record_t * *state*, const record_t * *true_branch*, const record_t * *false_branch*)

Rules come in three different forms:

1. a -> b
2. a -> b | c
3. ∼a ->b

They share a common structure. First the left hand side gives the starting state and right hand side the states to transit to. This means that each rule has a starting state and depending on the form one or two transit states:

- The first form has only the true transit state;
- The second form had both true and false transit states;
- The thrird for has only the false transit state. When either the true or false transit state for a rule does not exists, 0 should be supplied.

Parameters:
state Starting state

true_branch True transit state

false_branch False transit state

Returns:
TRUE if the rule has been added successfully, FALSE otherwise.

Definition at line 134 of file pdl_rule.c.

Referenced by add_rule().

8.46.2.2 rule_t * add_rule (record_t * *state*, record_t * *true_branch*, record_t * *false_branch*)

Add a new rule to the list of rules. This function acts as a wrapper function for _add_rule().

Parameters:

state Starting state

true_branch True transit state

false_branch False transit state

Definition at line 76 of file pdl_rule.c.

8.46.2.3 void allow_new_rules (BOOL *allow*)

Is it allowed to add new rules?

Parameters:

allows TRUE if adding new rules is allowed, FALSE otherwise.

Definition at line 61 of file pdl_rule.c.

8.46.2.4 const rule_t * find_state (const rule_t * *rule*, const char * *state*)

Find a state with name state.

Parameters:

state Name of the state to be found.

Returns:

Rule which contains the state or 0 when no such rule could be found.

Definition at line 192 of file pdl_rule.c.

8.46.2.5 void free_rules (rule_t * *rule*)

Free all resource associated with the rule.

Parameters:

rule Rule for which the resources must be freed.

Definition at line 278 of file pdl_rule.c.

8.46.2.6 rule_t * get_top_rule (void)

Get the top rule.

Returns:

Top rule.

Definition at line 299 of file pdl_rule.c.

8.46.2.7 void reduce_rule (rule_t * *rule*)

Reduce a rule to its elementry form, i.e. all variables in the rule are substituted by their respective values.

Parameters:
rule Rule to reduce.

Definition at line 208 of file pdl_rule.c.

Referenced by reduce_policies().

8.46.2.8 void show_rules (const rule_t * *rule*)

Show a rule and its descendants.

Parameters:
rule Rule to display.

Definition at line 257 of file pdl_rule.c.

8.46.2.9 void start_new_rules (void)

Start a new list of rules.

Definition at line 48 of file pdl_rule.c.

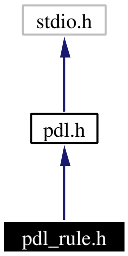
Referenced by add_policy().

8.47 pdl_rule.h File Reference

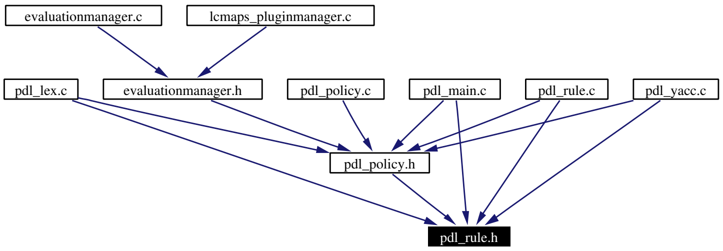
Include file for using the pdl rules.

`#include "pdl.h"`

Include dependency graph for pdl_rule.h:



This graph shows which files directly or indirectly include this file:



Data Structures

- struct rule_s

  Structure keeps track of the state and the true/false braches.

Typedefs

- typedef struct rule_s rule_t

  Structure keeps track of the state and the true/false braches.

Enumerations

- enum rule_type_t { STATE, TRUE_BRANCH, FALSE_BRANCH }

  Which type is the current rule.

Functions

- rule_t* add_rule (record_t *state, record_t *true_branch, record_t *false_branch)
- void free_rules (rule_t *rule)
- void show_rules (const rule_t *rule)
- void start_new_rules (void)
- rule_t* get_top_rule (void)
- void allow_new_rules (BOOL allow)

8.47.1 Detailed Description

Include file for using the pdl rules.

Author:
G.M. Venekamp (`venekamp@nikhef.nl`)

Version:

Revision:
1.8

Date:

Date:
2003/07/14 07:59:14

Definition in file pdl_rule.h.

8.47.2 Typedef Documentation

8.47.2.1 typedef struct rule_s rule_t

Structure keeps track of the state and the true/false braches.

8.47.3 Enumeration Type Documentation

8.47.3.1 enum rule_type_t

Which type is the current rule.

Enumeration values:

STATE State.

TRUE_BRANCH True branch.

FALSE_BRANCH False branch.

Definition at line 53 of file pdl_rule.h.

8.47.4 Function Documentation

8.47.4.1 rule_t* add_rule (record_t * *state*, record_t * *true_branch*, record_t * *false_branch*)

Add a new rule to the list of rules. This function acts as a wrapper function for _add_rule().

Parameters:

state Starting state

true_branch True transit state

false_branch False transit state

Definition at line 76 of file pdl_rule.c.

8.47.4.2 void allow_new_rules (BOOL *allow*)

Is it allowed to add new rules?

Parameters:

allows TRUE if adding new rules is allowed, FALSE otherwise.

Definition at line 61 of file pdl_rule.c.

Referenced by allow_rules().

8.47.4.3 void free_rules (rule_t * *rule*)

Free all resource associated with the rule.

Parameters:

rule Rule for which the resources must be freed.

Definition at line 278 of file pdl_rule.c.

Referenced by add_policy(), and free_policies().

8.47.4.4 rule_t* get_top_rule (void)

Get the top rule.

Returns:

Top rule.

Definition at line 299 of file pdl_rule.c.

8.47.4.5 void show_rules (const rule_t * *rule*)

Show a rule and its descendants.

Parameters:
rule Rule to display.

Definition at line 257 of file pdl_rule.c.

Referenced by show_policies().

8.47.4.6 void start_new_rules (void)

Start a new list of rules.

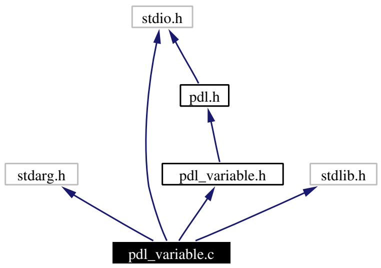
Definition at line 48 of file pdl_rule.c.

8.48 pdl_variable.c File Reference

Implementation of the pdl variables.

```
#include <stdarg.h>
#include <stdio.h>
#include <stdlib.h>
#include "pdl_variable.h"
```

Include dependency graph for pdl_variable.c:



Functions

- BOOL _add_variable (const record_t *name, const record_t *value)
- var_t* find_variable (const char *name)
- var_t* detect_loop (const char *name, const char *value)
- void add_variable (record_t *name, record_t *value)
- void free_variables (void)
- const char* reduce_to_var (const char *name)
- var_t* get_variables (void)
- void show_variables (void)

8.48.1 Detailed Description

Implementation of the pdl variables.

Not all functions defined in this file are accessible to everyone. A subset is used by the pdl variable functions themselves. For the list API functions look in pdl_variables.h.

Author:
G.M. Venekamp (venekamp@nikhef.nl)

Version:

Revision:
1.8

Date:

Date:
2003/07/16 09:30:58

Definition in file pdl_variable.c.

8.48.2 Function Documentation

8.48.2.1 BOOL _add_variable (const record_t ∗ *name*, const record_t ∗ *value*)

Actual implementation of the add_variable call. When the variable has been added the call returns TRUE, otherwise its FALSE. There can be several reasons for failure:

- Variable allready exists;
- Variable refers to itself through a loop;
- No more resources to allocate for variable.

Parameters:
name Name of the variable to be added.
value Value of the variable.

Returns:
TRUE in case the variable has been added, FALSE otherwise.

Definition at line 86 of file pdl_variable.c.

Referenced by add_variable().

8.48.2.2 void add_variable (record_t ∗ *name*, record_t ∗ *value*)

Wrapper function for the _add_variable() function call. The hard work is done in the _add_variable() call. When that call succeeds only the resources allocated for holding the name and value parameters are freed, i.e. the structures name and value. In case the _add_variable() calls fails, the string that is contained within the name and value strutures is freed as well.

Parameters:
name Name of the variable.
value Value of the variable.

Definition at line 61 of file pdl_variable.c.

8.48.2.3 var_t ∗ detect_loop (const char ∗ *name*, const char ∗ *value*)

Try to detect a loop in the variable references. When e.g. a=b, b=c and c=a, then the call should detect a loop.

Parameters:

name Name of the variable.

value Value of the variable.

Returns:

0 if no loop was detected. When a loop is detected, the first variable in the loop is returned.

Definition at line 189 of file pdl_variable.c.

Referenced by _add_variable().

8.48.2.4 var_t * find_variable (const char * *name*)

Find a variable based on the variable name. This way the value of a variable can be retrieved.

Parameters:

name Name of the variable to find.

Returns:

Pointer to the corresponding variable, or 0 when not found.

Definition at line 164 of file pdl_variable.c.

8.48.2.5 void free_variables (void)

Free the resources allocated for the variables.

Definition at line 138 of file pdl_variable.c.

Referenced by free_resources().

8.48.2.6 var_t * get_variables (void)

Get a list of all variables in the configure file.

Returns:

First variable of the list.

Definition at line 257 of file pdl_variable.c.

8.48.2.7 const char * reduce_to_var (const char * *name*)

Reduce the variable to its real value. When a variable has another variable as its value, the variable will be reduced to the value of the refering variable.

Parameters:

name Name of the variable to be reduced.

Returns:

Real value of the redunced variable.

Definition at line 235 of file pdl_variable.c.

8.48.2.8 void show_variables (void)

Print all variables and their value as described in the configure file to stdout.

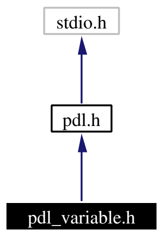
Definition at line 268 of file pdl_variable.c.

8.49 pdl_variable.h File Reference

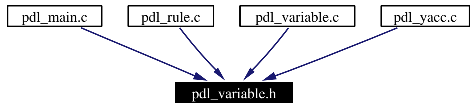
Include file for using the pdl variables.

`#include "pdl.h"`

Include dependency graph for pdl_variable.h:



This graph shows which files directly or indirectly include this file:



Data Structures

- struct var_s

    Structure keeps track of the variables, their value and the line number they are defined on.

Typedefs

- typedef struct var_s var_t

    Structure keeps track of the variables, their value and the line number they are defined on.

Functions

- void add_variable (record_t ∗name, record_t ∗value)
- const char∗ reduce_to_var (const char ∗name)
- void show_variables (void)
- void free_variables (void)
- var_t∗ get_variables (void)

8.49.1 Detailed Description

Include file for using the pdl variables.

All functions listed in here are accessible and usable for external "modules".

Author:
G.M. Venekamp (venekamp@nikhef.nl)

Version:

Revision:
1.5

Date:

Date:
2003/05/26 10:50:27

Definition in file pdl_variable.h.

8.49.2 Typedef Documentation

8.49.2.1 typedef struct var_s var_t

Structure keeps track of the variables, their value and the line number they are defined on.

8.49.3 Function Documentation

8.49.3.1 void add_variable (record_t ∗ *name*, record_t ∗ *value*)

Wrapper function for the _add_variable() function call. The hard work is done in the _add_variable() call. When that call succeeds only the resources allocated for holding the name and value parameters are freed, i.e. the structures name and value. In case the _add_variable() calls fails, the string that is contained within the name and value strutures is freed as well.

Parameters:

name Name of the variable.

value Value of the variable.

Definition at line 61 of file pdl_variable.c.

8.49.3.2 void free_variables (void)

Free the resources allocated for the variables.

Definition at line 138 of file pdl_variable.c.

8.49.3.3 var_t* get_variables (void)

Get a list of all variables in the configure file.

Returns:

First variable of the list.

Definition at line 257 of file pdl_variable.c.

8.49.3.4 const char* reduce_to_var (const char * *name*)

Reduce the variable to its real value. When a variable has another variable as its value, the variable will be reduced to the value of the refering variable.

Parameters:

name Name of the variable to be reduced.

Returns:

Real value of the redunced variable.

Definition at line 235 of file pdl_variable.c.

Referenced by reduce_rule().

8.49.3.5 void show_variables (void)

Print all variables and their value as described in the configure file to stdout.

Definition at line 268 of file pdl_variable.c.

Chapter 9

edg-lcmaps Page Documentation

9.1 example plugin

9.2 bescrijving

beschrijf beschrijf ...

9.3 ldap enforcement plugin

9.4 SYNOPSIS

lcmaps_ldap_enf.mod -maxuid <maxuid> -maxpgid <maxpgid> -maxsgid <maxsgid> -hostname <hostname> -port <port> [-require_all_groups [yes|no]] -dn_manager <DN> -ldap_pw <path/filename> -sb_groups <seachbase> -sb_user <searchbase>

9.5 DESCRIPTION

Ldap enforcement plugin will alter the user and group settings in the ldap database, using the user and groups settings provided by the credential acquisition plugins. Note that LDAP has to be used as the source of account information for PAM or NSS and has to be RFC 2307 complient. (see documentation)

9.6 OPTIONS

9.6.1 -maxuid <maxuid>

Maximum number of uids to be used. Strongly advised is to set this to 1.

9.6.2 -maxpgid <maxpgid>

Maximum number of primary gids to be used.

9.6.3 -maxsgid <maxsgid>

Maximum number of (secondairy) gids to be used (not including primary group). Advised is to set this to 1.

9.6.4 -hostname <hostname>

The hostname on which the LDAP server is running, e.g. asen.nikhef.nl

9.6.5 -port <port>

The port number to which to connect, e.g. 389

9.6.6 -require_all_groups [yes|no]

Specify if all groups set by the PluginManager shall be used. Default is'yes'

9.6.7 -dn_manager <DN>

DN of the LDAP manager, e.g. "cn=Manager,dc=root"

9.6.8 -ldap_pw <path/filename>

Path to the file containing the password of the LDAP manager. Note: the mode of the file containing the password must be read-only for root (400), otherwise the plugin will not run.

9.6.9 -sb_groups <seachbase>

Search base for the (secondairy) groups, e.g. "ou=LocalGroups, dc=foobar, dc=ough"

9.6.10 -sb_user <searchbase>

Search base for the user, e.g. "ou=LocalUsers, dc=foobar, dc=ough"

9.7 RETURN VALUE

- LCMAPS_MOD_SUCCESS : succes
- LCMAPS_MOD_FAIL : failure
- LCMAPS_MOD_NOFILE : db file not found (will halt LCMAPS initialization)

9.8 ERRORS

See bugzilla for known errors (`http://marianne.in2p3.fr/datagrid/bugzilla/`)

9.9 SEE ALSO

lcmaps_localaccount.mod, **lcmaps_poolaccount.mod**, **lcmaps_posix_enf.mod**, **lcmaps_voms.mod**

9.10 localaccount plugin

9.11 SYNOPSIS

lcmaps_localaccount.mod [-gridmapfile|-GRIDMAPFILE|-gridmap|-GRIDMAP <location gridmapfile>]

9.12 DESCRIPTION

The plugin is a Acquisition Plugin and will provide the LCMAPS system with Local Account credential information. To do this it needs to look up the Distinghuished Name (DN) from a user's certificate in the grid-mapfile. If this DN is found in the grid-mapfile the plugin knows the mapped local (system) account username. By knowing the username of the local account the plugin can gather additional information about this account. The plugin will resolve the UID, GID and all the secundary GIDs. When this all has been done and there weren't any problems detected the plugin will add this information to a datastructure in the Plugin Manager. The plugin will finish it's run with a LCMAPS_MOD_SUCCESS. This result will be reported to the Plugin Manager which started this plugin and it will forward this result to the Evaluation Manager which will take appropriate actions for the next plugin to run. Normally this plugin would be followed by a Enforcement plugin that can apply these gathered credentials in a way that is appropriate to a system administration's needs.

9.13 OPTIONS

9.13.1 -GRIDMAPFILE <gridmapfile>

See -gridmap

9.13.2 -gridmapfile <gridmapfile>

See -gridmap

9.13.3 -GRIDMAP <gridmapfile>

See -gridmap

9.13.4 -gridmap <gridmapfile>

When this option is set in the initialization string it will override the default path of to the grid-mapfile. It is advised to use a absolute path to the grid-mapfile to avoid usage of the wrong file(path). When this option is set but without a path to the grid-mapfile will fail the initialisation of the plugin and the plugin will not run untill it has been disposed and reloaded.

9.14 RETURN VALUES

- LCMAPS_MOD_SUCCESS : Success
- LCMAPS_MOD_FAIL : Failure

9.15 ERRORS

See bugzilla for known errors (http://marianne.in2p3.fr/datagrid/bugzilla/)

9.16 SEE ALSO

lcmaps_ldap_enf.mod, **lcmaps_poolaccount.mod**, **lcmaps_posix_enf.mod**, **lcmaps_voms.mod**

9.17 poolaccount plugin

9.18 SYNOPSIS

lcmaps_poolaccount.mod [-gridmapfile|-GRIDMAPFILE|-gridmap|-GRIDMAP <location gridmapfile>] [-gridmapdir|-GRIDMAPDIR <location gridmapdir>]

9.19 DESCRIPTION

The plugin is a Acquisition Plugin and will provide the LCMAPS system with Pool Account credential information. To do this it needs to look up the Distinghuished Name (DN) from a user's certificate in the grid-mapfile. If this DN is found in the grid-mapfile the plugin now knows to which pool of local system account the user wil be mapped. To convert the poolname (starting with a dot or point in stead of a alfanumeric character) will be checked with a special list of availeble local accounts. This list is located in the \i gridmapdir and is made of filenames. These filenames correspond to the system account's username. (Like a DN is mapped to .test and you will find a bunch of test001, test002, etc. in the gridmapdir)

When there are no pool accounts taken and the user is new the plugin will get a directory listing of the gridmapdir. This list will contain usernames corrisponding to system accounts specially designated for pool accounting. The plugin resolved the mapping of a certain pool name, let say '.test'. The plugin will look in the directory list en will find the first file in the list corrisponding with 'test', like the string 'test001'. This 'test001' is linked to an i-node (a filename 'in' a directory is linked to an pointer structure that forms the base of a Unix file system). This i-node can belinked to another file, besides this 'test001' file. Since we indicated a clean setup there is no other link, just the link between 'test001' and a i-node. This means that this username 'test001' corrisponding to a system account is not yet used by anyone else. To make a link between the user and this pool account the plugin will make a new file named as the Distinghuished Name (in a URL-Encode string) of the user. The nice part is that this new file will be attached to the same i-node as the file 'test001' indicating a link between the pool account and the user.

When a user returns to this site the plugin will look for the distinghuished name of the user (URL encoded) in this directory. Nice the user already left his trace in the directory with a link to it's already assigned pool account the user will now be mapped again to this (already) assigned pool account.

When the plugin assigned the pool account it will resolve all the data that can be known about this system account. The plugin will resolve the UID, GID and all the secundary GIDs. When this all has been done and there weren't any problems detected the plugin will add this information to a datastructure in the Plugin Manager. The plugin will finish it's run with a LCMAPS_MOD_SUCCESS. This result will be reported to the Plugin Manager which started this plugin and it will forward this result to the Evaluation Manager which will take appropriate actions for the next plugin to run. Normally this plugin would be followed by a Enforcement plugin that can apply these gathered credentials in a way that is appropriate to a system administration's needs.

9.20 OPTIONS

9.20.1 -GRIDMAPFILE <gridmapfile>

See -gridmap

9.20.2 -gridmapfile <gridmapfile>

See -gridmap

9.20.3 -GRIDMAP <gridmapfile>

See -gridmap

9.20.4 -gridmap <gridmapfile>

When this option is set in the initialization string it will override the default path of to the grid-mapfile. It is advised to use a absolute path to the grid-mapfile to avoid usage of the wrong file(path). When this option is set but without a path to the grid-mapfile will fail the initialisation of the plugin and the plugin will not run untill it has been disposed and reloaded.

9.20.5 -GRIDMAPDIR <gridmapdir>

See -gridmapdir

9.20.6 -gridmapdir <gridmapdir>

When this option is set in the initialization string it will override the default path of to the gridmapdir. It is advised to use a absolute path to the gridmapdir to avoid usage of the wrong path. When this option is set but without a path to the gridmapdir will fail the initialisation of the plugin and the plugin will not run untill it has been disposed and reloaded.

9.21 RETURN VALUES

- LCMAPS_MOD_SUCCESS : Success

- LCMAPS_MOD_FAIL : Failure

9.22 ERRORS

See bugzilla for known errors (http://marianne.in2p3.fr/datagrid/bugzilla/)

9.23 SEE ALSO

lcmaps_ldap_enf.mod, **lcmaps_localaccount.mod**, **lcmaps_posix_enf.mod**, **lcmaps_voms.mod**

9.24 posix enforcement plugin

9.25 SYNOPSIS

lcmaps_posix_enf.mod [-maxuid|-MAXUID <number of uids>] [-maxpgid|-MAXPGID <number of primary gids>] [-maxsgid|-MAXSGID <number of secundary gids>]

9.26 DESCRIPTION

The Posix Enforcement plugin will enforce or apply the gathered credentials that are stashed in the datastructure of the Plugin Manager. The plugin will get the credential information that is gathered by one or more Acquisition plugins. As this indicates there has the a Acquisition plugin already runned prior to this Enforcement. All of the gathered information will be checked by looking into the 'passwd' file of the system. These files have information about all registered system account and it's user groups.

The Posix Enforcent plugin does not validate the secundary GIDs. It does check the existance of the GID and the UID. They must exist although it is not needed that the GID and UID are a pair of each other.

With the usage of setuid, setgid and setgroups will the process be changes of it's ownership by root. The new owner will be the user by his credentials gathered aan system account.

9.27 OPTIONS

9.27.1 -MAXUID <number of uids>

See -maxuid

9.27.2 -maxuid <number of uids>

This will set the maximum allowed UIDs that this plugin will handle. On this moment there can never be more than one or less than one UID. In the final part of the code where the setuid() is given there will only be made use of the first UID. Al the others will never be touched untill the code is changed by a developer. By setting the value to a maximum there will be a failure raised when the amount of UIDs exceed the set maximum. Without this value the plugin will continue and will enforce only the first found value in the credential data structure.

9.27.3 -MAXPGID <number of primary gids>

See -maxpgid

9.27.4 -maxpgid <number of primary gids>

This will set the maximum allowed Primary GIDs that this plugin will handle. On this moment there can never be more than one or less than one Primary GIDs. In the final part of the code where the setgid() is given there will only be made use of the first Primary GID. Al the others will never be touched untill the code is changed by a developer. By setting the value to a maximum there will be a failure raised when the amount of Primary GIDs exceed the set maximum. Without this value the plugin will continue and will enforce only the first found value in the credential data structure.

9.27.5 -MAXSGID <number of secundary gids>

See -maxsgid

9.27.6 -maxsgid <number of secundary gids>

This will set the maximum allowed Secundary GIDs that this plugin will handle. On this moment the limit of the amount of Secundary GIDs is set in the system variable NGROUPS. This variable is usually 32. If NGROUPS is not set by the system, the limit will be set to 32 Secundary GIDs. In the final part of the code there is a setgroups() called. That function will apply all the gathered secundary groups availeble.

9.28 RETURN VALUES

- LCMAPS_MOD_SUCCESS : Success
- LCMAPS_MOD_FAIL : Failure

9.29 ERRORS

See bugzilla for known errors (`http://marianne.in2p3.fr/datagrid/bugzilla/`)

9.30 SEE ALSO

lcmaps_ldap_enf.mod, **lcmaps_localaccount.mod**, **lcmaps_poolaccount.mod**, **lcmaps_voms.mod**

9.31 voms plugin

9.32 SYNOPSIS

lcmaps_voms.mod -vomsdir <vomsdir> -certdir <certdir>

9.33 DESCRIPTION

This plugin forms the link between the voms data on a certificate and the lcmaps system. It will acquire voms data via the VOMS API. The API specifies a Retrieve function that will build a voms data structure in de plug-in. The Retrieve function need a OpenSSL x.509 (chain of) certificate(s). The plug-in will transfer the needed voms data to the Plugin Manager where this 'raw' credential data will be storaged and reachable like al the other know data (as gathered uid(s), Primary GID(s) and Secendary GIDs. By making use of this plugin other voms-'aware' plugins can transparently make use of the needed voms data without knowing the exact way of data extraction (OpenSSL/usefull Globus tools/etc.).

9.34 OPTIONS

9.34.1 -VOMSDIR <vomsdir>

See -vomsdir

9.34.2 -vomsdir <vomsdir>

This is the directory which contains the certificates of the VOMS servers

9.34.3 -CERTDIR <certdir>

See -vomsdir

9.34.4 -certdir <certdir>

This is the directory which contains the CA certificates

9.35 RETURN VALUES

- LCMAPS_MOD_SUCCESS : Success
- LCMAPS_MOD_FAIL : Failure

9.36 ERRORS

See bugzilla for known errors (http://marianne.in2p3.fr/datagrid/bugzilla/)

9.37 SEE ALSO

lcmaps_ldap_enf.mod, **lcmaps_poolaccount.mod**, **lcmaps_posix_enf.mod**, **lcmaps_localaccount.mod**, **lcmaps_poolaccount.mod**

9.38 voms localgroup plugin

9.39 SYNOPSIS

lcmaps_voms_localgroup.mod -GROUPMAPFILE|-groupmapfile|-GROUPMAP|-groupmap <groupmapfile> [-mapall]

9.40 DESCRIPTION

The localgroup acquisition plugin is a voms-'aware' plugin. The plugin's main purpose is to gather credential information from the given **Voms** \bAcquisition plugin. This plugin will gather a primary GID and additional secundary GIDs. In the credential data datastructure in the Plugin Manager are all the VO-GROUP-ROLE(-CAPABILITY) values stored. This plugin will get this data and compare all the VO-GROUP-ROLE values with the that is by default known as \b'groupmapfile'\b. The plugin will lookup each value (a VO-GROUP-ROLE combination) and will search in the groupmapfile for a match. Wildcards can be used in the groupmapfile to match VO-GROUP-ROLE combinations.

EXAMPLE 'groupmapfile':

/VO=atlas/GROUP=mcprod atmcprod

/VO=atlas/GROUP=* atlasgrps

/VO=atlas/GROUP=mcprod as VO-GROUP combination from the gathered credential data will match with /VO=atlas/GROUP=mcprod and there will be a mapping made to the GID of the 'atmcprod' group. All the other groups within the 'atlas' VO will be mapped to 'atlasgrps'. If there is a user with /VO=cms that user can not be mapped to any local system group unless there will be an extra row in the groupmapfile like '/VO=* allothers' making a mapping from anyother VO-GROUP-ROLE combination to 'allothers'. What u can allready read between the lines that the most significant row must be on top and the least significant row must be on the bottom side of the groupmapfile.

For every value in the Plugin Manager there will be a search in the groupmapfile. The first extracted and gathered VO-GROUP-ROLE combination will find it's way to be primary group. Unless there has been another plugin already run that filled up the primary group. The userinterface software has the possibility to set a userdefined order in the VOMS values that will be put on user's proxy certificate. With this feature the user can controle the primary group what could have more functionality in the future then of now (audit/billing/etc.).

9.41 OPTIONS

9.41.1 -GROUPMAPFILE <groupmapfile>

See -groupmap

9.41.2 -groupmapfile <groupmapfile>

See -groupmap

9.41.3 -GROUPMAP <groupmapfile>

See -groupmap

9.41.4 -groupmap <groupmapfile>

When this option is set in the initialization string it will override the default path of to the groupmapfile. It is advised to use a absolute path to the groupmapfile to avoid usage of the wrong file(path). When this option is set but without a path to the groupmapfile will fail the initialisation of the plugin and the plugin will not run untill it has been disposed and reloaded.

9.41.5 -mapall

If this parameter is set the plugin is forced to map all voms data entries to (system) groups and find there GID. If not all voms data (VO-GROUP-ROLE) entries on the certificate match with rows in the groupmapfile the plugin will fail. There is no communication between different plugins (like the poolgroup plugin) about the failures. A log entry will state the VO-GROUP-ROLE combination what made the plugin fail.

9.42 RETURN VALUES

- LCMAPS_MOD_SUCCESS : Success
- LCMAPS_MOD_FAIL : Failure

9.43 ERRORS

See bugzilla for known errors (http://marianne.in2p3.fr/datagrid/bugzilla/)

9.44 SEE ALSO

lcmaps_ldap_enf.mod, **lcmaps_poolaccount.mod**, **lcmaps_posix_enf.mod**, **lcmaps_voms.mod**

9.45 voms poolaccount plugin

9.46 SYNOPSIS

lcmaps_voms_poolaccount.mod lcmaps_poolaccount.mod [-gridmapfile|-GRIDMAPFILE|-gridmap|-GRIDMAP <location grid-mapfile>] [-gridmapdir|-GRIDMAPDIR <location gridmapdir>] [-do_not_use_secondary_gids] [-do_not_require_primary_gid]

9.47 DESCRIPTION

This poolaccount acquisition plugin is a voms-'aware' modificated from the 'poolaccount' plugin. The plugin's main purpose is to gather credential information from the given **Voms** Acquisition plugin. **This** plugin will gather a UID. In the credential data datastructure in the Plugin Manager are all the VO-GROUP-ROLE(-CAPABILITY) values stored. This plugin will get this data and compare the first known VO-GROUP-ROLE combination that has been extracted from the certificate with entries in the same 'grid-mapfile' as the localaccount and poolaccount plugin use. In that file there will be VO-GROUP-ROLE combinations stored with each entry a mapping to a poolaccount.

EXAMPLE:

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

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

When a user comes in with his certificate and his first known VO is 'wilma' the plugin will get a poolaccount from the '.test' pool. This could result in 'test001' as a poolaccount for this user. The linking between '/VO=wilma/GROUP=*', this user and a poolaccount must be made in the same directory as the \bPoolaccount \bPlugin otherwise there will be a great chance of inconsistancy when both are used on a site. The same filename and i-node link will be made as the Poolaccount Plugin with one little change in the filename of the user's Distinghuished Name. This will no longer be only it's DN but has al the gathered groups concatinated (attached) to this DN. So a linked DN could look like:

EXAMPLE DN with pool/localgroups attached: 2fo3ddutchgrid2fo3dusers2fo3dnikhef2fcn3dmartijn20steenbakkers3apool0013ab

This means when a user changes it's VO-GROUP-ROLE sublimentary VO-'identity' the gathered groups will change. Indicating a change in this sublimentary 'identity' and this will result in an other poolaccount on the site's system. The change has effect to the sublimentary identity because the Distinghuished Name of the user is not changed. Fysicaly and digitally it is the same user, but with different rights and obligations.

9.48 NOTE 1

This plugin will only be run succesfully when localgroup and/or poolgroup has already been run. There is no check if another plugin has ialready run but there will be a logical notice to the logs that it would.

9.49 NOTE 2

If '-do_not_require_primary_gid' and '-do_not_use_secondary_gids' is selected in the initialize part of the plugin it has become a little useless. This means a user doesn't need a primary GID, but also can do without any secundary GIDs. In other words the plugin will **not** fail when no credentials at all have been gathered from the voms credentials. It is prohibited to use these settings in combination of each other. Selection of the two settings is blocked.

9.50 OPTIONS

9.50.1 -GRIDMAPFILE <gridmapfile>

See -gridmap

9.50.2 -gridmapfile <gridmapfile>

See -gridmap

9.50.3 -GRIDMAP <gridmapfile>

See -gridmap

9.50.4 -gridmap <gridmapfile>

When this option is set in the initialization string it will override the default path of to the grid-mapfile. It is advised to use a absolute path to the grid-mapfile to avoid usage of the wrong file(path). When this option is set but without a path to the grid-mapfile will fail the initialisation of the plugin and the plugin will not run untill it has been disposed and reloaded.

9.50.5 -GRIDMAPDIR <gridmapdir>

See -gridmapdir

9.50.6 -gridmapdir <gridmapdir>

Here you can override the default directory path to the 'gridmapdir'. This directory should be the same directory as the one used by the 'normal' Poolaccount plugin. It holds all the poolaccount mappings that has/will be made by linking filenames to a i-node indicating a mapping between a Distinghuished Name with it's gathered VO-GROUP-ROLE combinations and a poolaccount.

9.50.7 -do_not_use_secondary_gids

This make a DN and VO-GROUP-ROLE mapping to a poolaccount based on only the DN and the group that has been designated as the primary group for this user with it's credential data. This will prevent the user from making constantly new mappings to other poolaccounts because of a slight change in the user's voms credentials from it's proxy certificate.

9.50.8 -do_not_require_primary_gid

The user will always need a primary GID. The plugin will check this value and fail if another plugin didn't presented Plugin Manager's credential data structure with a primary GID. If there is still a possibility of getting a primary GID then there can be made use of this cmdline option. It will disable the checking (and plugin failure) of the primary GID and it's existance. The primary GID is a (logical) needed value at this point because there will be made a link for the mapping process in the **groupmapdir**. To make sure that the credentials are correct and complete the system should have a primary GID.

9.51 RETURN VALUES

- LCMAPS_MOD_SUCCESS : Success
- LCMAPS_MOD_FAIL : Failure

9.52 ERRORS

See bugzilla for known errors (http://marianne.in2p3.fr/datagrid/bugzilla/)

9.53 SEE ALSO

lcmaps_ldap_enf.mod, **lcmaps_poolaccount.mod**, **lcmaps_posix_enf.mod**, **lcmaps_voms.mod**

9.54 voms poolgroup plugin

9.55 SYNOPSIS

lcmaps_voms_poolgroup.mod -GROUPMAPFILE|-groupmapfile|-GROUPMAP|-groupmap <groupmapfile> [-mapall] -GROUPMAPDIR|-groupmapdir <groupmapdir>

9.56 DESCRIPTION

The poolgroup acquisition plugin is a voms-'aware' plugin. The plugin's main purpose is to gather credential information from the given **Voms** \bAcquisition plugin. This plugin will gather a primary GID and additional secundary GIDs. In the credential data datastructure in the Plugin Manager are all the VO-GROUP-ROLE(-CAPABILITY) values stored. This plugin will get this data and compare all the VO-GROUP-ROLE values with the row entries in a file that is by default known as '\bgroupmapfile'. The plugin will lookup each value (a VO-GROUP-ROLE combination) and will search in the groupmapfile for a match. Wildcards can be used in the groupmapfile to match VO-GROUP-ROLE combinations.

EXAMPLE 'groupmapfile':

/VO=atlas/GROUP=mcprod mcprod

/VO=atlas/GROUP=mcprod .atlas

/VO=atlas/GROUP=dev .atlas

/VO=atlas/GROUP=∗ .atlas

/VO=atlas/GROUP=mcprod as VO-GROUP combination starts with a alfanumeric character (not a point or dot) and indicates a localgroup entry in the groupmapfile. The /VO=atlas/GROUP=∗ as VO-GROUP combi. secification indicates that all users from the Atlas VO with every other group than 'mcprod' will be mapped to the '.atlas' pool of (system) groups. Just like the \ipoolaccount\i plugin this plugin will link a entry (in this case a VO-GROUP-ROLE combination) to a locally known group (from this 'atlas'-pool there for a.k.a. pool group). This mapping between the VO-GROUP-ROLE combination and a pool group will be made with the use of 'multiple filename linking to a i-node'. For more information about this way of linking information in a filename to a i-node that represents a specific values please look at the poolaccount way of working. The difference with the poolaccount is that there is not a Distinghuished Name but a VO-GROUP-ROLE combination and there is no poolaccount but poolgroup defined in de groupmapfile (simulaire to the grid-mapfile). Also there is a new directory in use of this plugin. This directory is called (by default) \igroupdmapdir\i. This directory holds the i-nodes that are used for the mapping between poolgroups and the VO-GROUP-ROLE combination.

As you can see the in the example the 'mcprod' GROUP can be found by using the localgroup plugin and the poolgroup plugin. With the poolgroup plugin there can be made a mapping between '/VO=atlas/GROUP=mcprod' and the group 'atlas001' (based on the .atlas pool). The '/VO=atlas/GROUP=dev' entry will also get a group from this '.atlas' pool but will be mapped to a different group like 'atlas002'. Last but not least we have random other groups not predefined in the groupmapfile like '/VO=atlas/GROUP=foo'. This VO-GROUP combi. will be found with the

'/VO=atlas/GROUP=*' row in the groupmapfile. This VO-GROUP combi. will be mapped to a poolgroup (probably) called 'atlas003'. If someone makes use of a VO-GROUP combi. like '/VO=atlas/GROUP=bar' it will find an link in the i-node structure between '/VO=atlas/GROUP=*' and 'atlas003' indicating that '/VO=atlas/GROUP=bar' will get 'atlas003' designated as a mapping for this voms data.

For every value in the Plugin Manager there will be a search in the groupmapfile. The first extracted and gathered VO-GROUP-ROLE combination will find it's way to be primary group. Unless there has been another plugin already run that filled up the primary group. The userinterface software has the possibility to set a userdefined order in the VOMS values that will be put on user's proxy certificate. With this feature the user can controle the primary group what could have more functionality in the future then of now (audit/billing/etc.).

9.57 OPTIONS

9.57.1 -GROUPMAPFILE <groupmapfile>

See -groupmap

9.57.2 -groupmapfile <groupmapfile>

See -groupmap

9.57.3 -GROUPMAP <groupmapfile>

See -groupmap

9.57.4 -groupmap <groupmapfile>

When this option is set in the initialization string it will override the default path of to the groupmapfile. It is advised to use a absolute path to the groupmapfile to avoid usage of the wrong file(path). When this option is set but without a path to the groupmapfile will fail the initialisation of the plugin and the plugin will not run untill it has been disposed and reloaded.

9.57.5 -GROUPMAPDIR <groupmapdir>

See -groupmapdir

9.57.6 -groupmapdir <groupmapdir>

Here you can override the default directory path to the 'groupmapdir'. This directory is just like the gridmapdir and holds all the poolgroup mappings that has/will be made by linking filenames to a i-node indicating a mapping between a VO-GROUP-ROLE combination and a (system) group or GID.

9.57.7 -mapall

If this parameter is set the plugin is forced to map all voms data entries to (system) groups and find there GID. If not all voms data (VO-GROUP-ROLE) entries on the certificate match with rows in the groupmap-file the plugin will fail. There is no communication between different plugins (like the localgroup plugin) about the failures. A log entry will state the VO-GROUP-ROLE combination what made the plugin fail.

9.58 RETURN VALUES

- LCMAPS_MOD_SUCCESS : Success
- LCMAPS_MOD_FAIL : Failure

9.59 ERRORS

See bugzilla for known errors (`http://marianne.in2p3.fr/datagrid/bugzilla/`)

9.60 SEE ALSO

lcmaps_ldap_enf.mod, **lcmaps_poolaccount.mod**, **lcmaps_posix_enf.mod**, **lcmaps_voms.mod**

Index