

# edg-lcmaps Reference Manual

Generated by Doxygen 1.2.8.1

Thu Mar 4 17:39:02 2004



---

# Contents

<b>1</b>	<b>LCMAPS - Local Credential MAPping Service</b>	<b>1</b>
1.1	Introduction . . . . .	1
1.2	the LCMAPS Interfaces . . . . .	1
1.3	The LCMAPS plugins . . . . .	1
<b>2</b>	<b>edg-lcmaps Module Index</b>	<b>3</b>
2.1	edg-lcmaps Modules . . . . .	3
<b>3</b>	<b>edg-lcmaps Data Structure Index</b>	<b>5</b>
3.1	edg-lcmaps Data Structures . . . . .	5
<b>4</b>	<b>edg-lcmaps File Index</b>	<b>7</b>
4.1	edg-lcmaps File List . . . . .	7
<b>5</b>	<b>edg-lcmaps Page Index</b>	<b>9</b>
5.1	edg-lcmaps Related Pages . . . . .	9
<b>6</b>	<b>edg-lcmaps Module Documentation</b>	<b>11</b>
6.1	Interface to LCMAPS (library) . . . . .	11
6.2	The API to be used by the LCMAPS plugins . . . . .	12
6.3	The interface to the LCMAPS plugins . . . . .	13
<b>7</b>	<b>edg-lcmaps Class Documentation</b>	<b>15</b>
7.1	cred_data_s Struct Reference . . . . .	15
7.2	lcmaps_argument_s Struct Reference . . . . .	18
7.3	lcmaps_cred_id_s Struct Reference . . . . .	19
7.4	lcmaps_db_entry_s Struct Reference . . . . .	20
7.5	lcmaps_plugin_dl_s Struct Reference . . . . .	21
7.6	lcmaps_vo_data_s Struct Reference . . . . .	23
7.7	lcmaps_vo_mapping_s Struct Reference . . . . .	25

---

7.8	<code>plugin_s</code> Struct Reference	26
7.9	<code>policy_s</code> Struct Reference	27
7.10	<code>record_s</code> Struct Reference	28
7.11	<code>rule_s</code> Struct Reference	29
7.12	<code>var_s</code> Struct Reference	30
<b>8</b>	<b>edg-lcmapi File Documentation</b>	<b>31</b>
8.1	<code>_lcmapi_cred_data.h</code> File Reference	31
8.2	<code>_lcmapi_db_read.h</code> File Reference	33
8.3	<code>_lcmapi_defines.h</code> File Reference	36
8.4	<code>_lcmapi_log.h</code> File Reference	38
8.5	<code>_lcmapi_pluginmanager.h</code> File Reference	41
8.6	<code>_lcmapi_runvars.h</code> File Reference	44
8.7	<code>_lcmapi_utils.h</code> File Reference	47
8.8	<code>evaluationmanager.c</code> File Reference	50
8.9	<code>evaluationmanager.h</code> File Reference	53
8.10	<code>lcmapi.c</code> File Reference	56
8.11	<code>lcmapi.h</code> File Reference	58
8.12	<code>lcmapi_arguments.c</code> File Reference	61
8.13	<code>lcmapi_arguments.h</code> File Reference	62
8.14	<code>lcmapi_cred_data.c</code> File Reference	66
8.15	<code>lcmapi_cred_data.h</code> File Reference	68
8.16	<code>lcmapi_db_read.c</code> File Reference	70
8.17	<code>lcmapi_db_read.h</code> File Reference	75
8.18	<code>lcmapi_defines.h</code> File Reference	77
8.19	<code>lcmapi_gss_assist_gridmap.c</code> File Reference	80
8.20	<code>lcmapi_jobrep.c</code> File Reference	81
8.21	<code>lcmapi_ldap.c</code> File Reference	85
8.22	<code>lcmapi_localaccount.c</code> File Reference	89
8.23	<code>lcmapi_log.c</code> File Reference	90
8.24	<code>lcmapi_log.h</code> File Reference	92
8.25	<code>lcmapi_modules.h</code> File Reference	95
8.26	<code>lcmapi_plugin_example.c</code> File Reference	97
8.27	<code>lcmapi_pluginmanager.c</code> File Reference	98
8.28	<code>lcmapi_poolaccount.c</code> File Reference	104
8.29	<code>lcmapi_posix.c</code> File Reference	105
8.30	<code>lcmapi_runvars.c</code> File Reference	106

8.31	lcmmaps_test.c File Reference	108
8.32	lcmmaps_types.h File Reference	109
8.33	lcmmaps_utils.c File Reference	111
8.34	lcmmaps_utils.h File Reference	113
8.35	lcmmaps_vo_data.c File Reference	117
8.36	lcmmaps_vo_data.h File Reference	118
8.37	lcmmaps_voms.c File Reference	124
8.38	lcmmaps_voms_localgroup.c File Reference	126
8.39	lcmmaps_voms_poolaccount.c File Reference	127
8.40	lcmmaps_voms_poolgroup.c File Reference	128
8.41	pdl.h File Reference	129
8.42	pdl_main.c File Reference	135
8.43	pdl_policy.c File Reference	141
8.44	pdl_policy.h File Reference	145
8.45	pdl_rule.c File Reference	149
8.46	pdl_rule.h File Reference	155
8.47	pdl_variable.c File Reference	160
8.48	pdl_variable.h File Reference	164
<b>9</b>	<b>edg-lcmmaps Page Documentation</b>	<b>167</b>
9.1	Job Repository plugin	167
9.2	SYNOPSIS	167
9.3	DESCRIPTION	167
9.4	OPTIONS	167
9.5	afs plugin	169
9.6	SYNOPSIS	169
9.7	DESCRIPTION	169
9.8	OPTIONS	169
9.9	RETURN VALUES	170
9.10	ERRORS	170
9.11	SEE ALSO	170
9.12	example plugin	171
9.13	beschrijving	171
9.14	ldap enforcement plugin	172
9.15	SYNOPSIS	172
9.16	DESCRIPTION	172
9.17	OPTIONS	172

---

9.18 RETURN VALUE . . . . .	173
9.19 ERRORS . . . . .	173
9.20 SEE ALSO . . . . .	173
9.21 localaccount plugin . . . . .	174
9.22 SYNOPSIS . . . . .	174
9.23 DESCRIPTION . . . . .	174
9.24 OPTIONS . . . . .	174
9.25 RETURN VALUES . . . . .	174
9.26 ERRORS . . . . .	175
9.27 SEE ALSO . . . . .	175
9.28 poolaccount plugin . . . . .	176
9.29 SYNOPSIS . . . . .	176
9.30 DESCRIPTION . . . . .	176
9.31 OPTIONS . . . . .	176
9.32 RETURN VALUES . . . . .	177
9.33 ERRORS . . . . .	177
9.34 SEE ALSO . . . . .	177
9.35 posix enforcement plugin . . . . .	178
9.36 SYNOPSIS . . . . .	178
9.37 DESCRIPTION . . . . .	178
9.38 OPTIONS . . . . .	178
9.39 RETURN VALUES . . . . .	179
9.40 ERRORS . . . . .	179
9.41 SEE ALSO . . . . .	179
9.42 voms plugin . . . . .	180
9.43 SYNOPSIS . . . . .	180
9.44 DESCRIPTION . . . . .	180
9.45 OPTIONS . . . . .	180
9.46 RETURN VALUES . . . . .	180
9.47 ERRORS . . . . .	180
9.48 SEE ALSO . . . . .	181
9.49 voms localgroup plugin . . . . .	182
9.50 SYNOPSIS . . . . .	182
9.51 DESCRIPTION . . . . .	182
9.52 OPTIONS . . . . .	182
9.53 RETURN VALUES . . . . .	183

---

9.54 ERRORS	183
9.55 SEE ALSO	183
9.56 voms poolaccount plugin	184
9.57 SYNOPSIS	184
9.58 DESCRIPTION	184
9.59 NOTE 1	184
9.60 NOTE 2	184
9.61 OPTIONS	185
9.62 RETURN VALUES	186
9.63 ERRORS	186
9.64 SEE ALSO	186
9.65 voms poolgroup plugin	187
9.66 SYNOPSIS	187
9.67 DESCRIPTION	187
9.68 OPTIONS	188
9.69 RETURN VALUES	189
9.70 ERRORS	189
9.71 SEE ALSO	189





---

## 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](#)

... miscellaneous:

1. [afs plugin](#)

2. [Job Repository plugin](#)

---

## Chapter 2

# edg-lcmaps Module Index

### 2.1 edg-lcmaps Modules

Here is a list of all modules:

Interface to LCMAPS (library) . . . . .	11
The API to be used by the LCMAPS plugins . . . . .	12
The interface to the LCMAPS plugins . . . . .	13



---

## Chapter 3

# edg-lcmaps Data Structure Index

### 3.1 edg-lcmaps Data Structures

Here are the data structures with brief descriptions:

<b>cred_data_s</b> (Structure that contains the gathered (local) credentials en VOMS info) . . . . .	15
<b>lcmaps_argument_s</b> (Structure representing an LCMAPS plugin run argument) . . . . .	18
<b>lcmaps_cred_id_s</b> (Structure representing an LCMAPS credential) . . . . .	19
<b>lcmaps_db_entry_s</b> (LCMAPS data base element structure) . . . . .	20
<b>lcmaps_plugindl_s</b> (The lcmaps plugin module structure) . . . . .	21
<b>lcmaps_vo_data_s</b> (Structure that contains the VO information found in the user's gss credential)	23
<b>lcmaps_vo_mapping_s</b> (Structure that contains the VO information string (or FQAN) and the local Gid it is mapped to) . . . . .	25
<b>plugin_s</b> (Structure holds a plugin name and its arguments, as well as the line number the plugin is first mentioned) . . . . .	26
<b>policy_s</b> (Keeping track of found policies) . . . . .	27
<b>record_s</b> (Structure is used to keep track of strings and the line they appear on) . . . . .	28
<b>rule_s</b> (Structure keeps track of the state and the true/false braches) . . . . .	29
<b>var_s</b> (Structure keeps track of the variables, their value and the line number they are defined on)	30

---



---

## Chapter 4

# edg-lcmaps File Index

### 4.1 edg-lcmaps File List

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

<b>_lcmaps_cred_data.h</b> (Internal header file of LCMAPS credential data) . . . . .	31
<b>_lcmaps_db_read.h</b> (Internal header file of LCMAPS database reader) . . . . .	33
<b>_lcmaps_defines.h</b> (Internal header file with some common defines for LCMAPS) . . . . .	36
<b>_lcmaps_log.h</b> (Internal header file for LCMAPS logging routines) . . . . .	38
<b>_lcmaps_pluginmanager.h</b> (API of the PluginManager) . . . . .	41
<b>_lcmaps_runvars.h</b> (API of runvars structure) . . . . .	44
<b>_lcmaps_utils.h</b> (Internal header for the LCMAPS utilities) . . . . .	47
<b>evaluationmanager.c</b> (Implementation of the evaluation manager interface) . . . . .	50
<b>evaluationmanager.h</b> (Evaluation Manager interface definition) . . . . .	53
<b>lcmaps.c</b> (The LCMAPS module - the local credential mapping service) . . . . .	56
<b>lcmaps.h</b> (API of the LCMAPS library) . . . . .	58
<b>lcmaps_arguments.c</b> (LCMAPS module for creating and passing introspect/run argument lists)	61
<b>lcmaps_arguments.h</b> (Public header file to be used by plugins) . . . . .	62
<b>lcmaps_cred_data.c</b> (Routines to handle lcmaps credential data) . . . . .	66
<b>lcmaps_cred_data.h</b> (Public header file to be used by plugins) . . . . .	68
<b>lcmaps_db_read.c</b> (The LCMAPS database reader) . . . . .	70
<b>lcmaps_db_read.h</b> (Header file for LCMAPS database structure) . . . . .	75
<b>lcmaps_defines.h</b> (Public header file with common definitions for the LCMAPS (authorization modules)) . . . . .	77
<b>lcmaps_gss_assist_gridmap.c</b> (Legacy interface for LCMAPS) . . . . .	80
<b>lcmaps_jobrep.c</b> (Plugin to get data for the Job Repository Database) . . . . .	81
<b>lcmaps_ldap.c</b> (Interface to the LCMAPS plugins) . . . . .	85
<b>lcmaps_localaccount.c</b> (Interface to the LCMAPS plugins) . . . . .	89
<b>lcmaps_log.c</b> (Logging routines for LCMAPS) . . . . .	90
<b>lcmaps_log.h</b> (Logging API for the LCMAPS plugins and LCMAPS itself) . . . . .	92
<b>lcmaps_modules.h</b> (The LCMAPS authorization plugins/modules should "include" this file) . .	95
<b>lcmaps_plugin_example.c</b> (Interface to the LCMAPS plugins) . . . . .	97
<b>lcmaps_pluginmanager.c</b> (The plugin manager for LCMAPS) . . . . .	98
<b>lcmaps_poolaccount.c</b> (Interface to the LCMAPS plugins) . . . . .	104
<b>lcmaps_posix.c</b> (Interface to the LCMAPS plugins) . . . . .	105
<b>lcmaps_runvars.c</b> (Extract variables that will be used by the plugins) . . . . .	106
<b>lcmaps_test.c</b> (Program to test the LCMAPS and its plugins) . . . . .	108
<b>lcmaps_types.h</b> (Public header file with typedefs for LCMAPS) . . . . .	109

---

<b>lcmaps_utils.c</b> (The utilities for the LCMAPS) . . . . .	111
<b>lcmaps_utils.h</b> (API for the utilities for the LCMAPS) . . . . .	113
<b>lcmaps_vo_data.c</b> (LCMAPS utilities for creating and accessing VO data structures) . . . . .	117
<b>lcmaps_vo_data.h</b> (LCMAPS module for creating and accessing VO data structures) . . . . .	118
<b>lcmaps_voms.c</b> (Interface to the LCMAPS plugins) . . . . .	124
<b>lcmaps_voms_localgroup.c</b> (Interface to the LCMAPS plugins) . . . . .	126
<b>lcmaps_voms_poolaccount.c</b> (Interface to the LCMAPS plugins) . . . . .	127
<b>lcmaps_voms_poolgroup.c</b> (Interface to the LCMAPS plugins) . . . . .	128
<b>pdl.h</b> (General include file) . . . . .	129
<b>pdl_main.c</b> (All functions that do not fit elsewhere can be found here) . . . . .	135
<b>pdl_policy.c</b> (Implementation of the pdl policies) . . . . .	141
<b>pdl_policy.h</b> (Include file for using the pdl policies) . . . . .	145
<b>pdl_rule.c</b> (Implementation of the pdl rules) . . . . .	149
<b>pdl_rule.h</b> (Include file for using the pdl rules) . . . . .	155
<b>pdl_variable.c</b> (Implementation of the pdl variables) . . . . .	160
<b>pdl_variable.h</b> (Include file for using the pdl variables) . . . . .	164



---

## Chapter 5

# edg-lcmaps Page Index

### 5.1 edg-lcmaps Related Pages

Here is a list of all related documentation pages:

Job Repository plugin . . . . .	167
afs plugin . . . . .	169
example plugin . . . . .	171
ldap enforcement plugin . . . . .	172
localaccount plugin . . . . .	174
poolaccount plugin . . . . .	176
posix enforcement plugin . . . . .	178
voms plugin . . . . .	180
voms localgroup plugin . . . . .	182
voms poolaccount plugin . . . . .	184
voms poolgroup plugin . . . . .	187



---

## 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

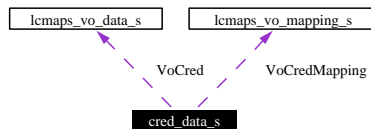
# edg-lcmaps Class Documentation

### 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](#)
- lcmaps\_vo\_mapping\_t\* [VoCredMapping](#)
- int [cntUid](#)
- int [cntPriGid](#)
- int [cntSecGid](#)
- int [cntVoCred](#)
- int [cntVoCredString](#)
- int [cntVoCredMapping](#)

#### 7.1.1 Detailed Description

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

Definition at line 56 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 62 of file `lcmaps_cred_data.h`.

### 7.1.2.2 `lcmaps_vo_mapping_t * cred_data_s::VoCredMapping`

list of VO mapping structures

Definition at line 64 of file `lcmaps_cred_data.h`.

### 7.1.2.3 `char ** cred_data_s::VoCredString`

list of VO data strings

Definition at line 63 of file `lcmaps_cred_data.h`.

### 7.1.2.4 `int cred_data_s::cntPriGid`

number of primary groupIDs (in principle only one)

Definition at line 66 of file `lcmaps_cred_data.h`.

### 7.1.2.5 `int cred_data_s::cntSecGid`

number of secondary groupIDs (could be any number)

Definition at line 67 of file `lcmaps_cred_data.h`.

### 7.1.2.6 `int cred_data_s::cntUid`

number of userIDs

Definition at line 65 of file `lcmaps_cred_data.h`.

### 7.1.2.7 `int cred_data_s::cntVoCred`

number of VO data structures

Definition at line 68 of file `lcmaps_cred_data.h`.

### 7.1.2.8 `int cred_data_s::cntVoCredMapping`

number of VO mapping structures

Definition at line 70 of file `lcmaps_cred_data.h`.



**7.1.2.9 int cred\_data\_s::cntVoCredString**

number of VO data strings

Definition at line 69 of file lcmmaps\_cred\_data.h.

**7.1.2.10 char \* cred\_data\_s::dn**

user globus DN

Definition at line 58 of file lcmmaps\_cred\_data.h.

**7.1.2.11 gid\_t \* cred\_data\_s::priGid**

list of primary groupIDs

Definition at line 60 of file lcmmaps\_cred\_data.h.

**7.1.2.12 gid\_t \* cred\_data\_s::secGid**

list of secondary groupIDs

Definition at line 61 of file lcmmaps\_cred\_data.h.

**7.1.2.13 uid\_t \* cred\_data\_s::uid**

list of userIDs

Definition at line 59 of file lcmmaps\_cred\_data.h.

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

- [lcmmaps\\_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:

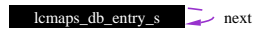
- `lcmaps_types.h`

## 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.

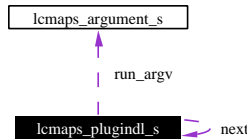
The documentation for this struct was generated from the following file:

- [lcmaps\\_db\\_read.h](#)

## 7.5 lcms\_pluginl\_s Struct Reference

the lcms plugin module structure.

Collaboration diagram for lcms\_pluginl\_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\\_pluginl\\_s](#)\* [next](#)

#### 7.5.1 Detailed Description

the lcms plugin module structure.

For internal use only.

Definition at line 103 of file `lcmaps_pluginmanager.c`.

#### 7.5.2 Field Documentation

##### 7.5.2.1 void \* lcms\_pluginl\_s::handle

dlopen handle to plugin module

Definition at line 105 of file `lcmaps_pluginmanager.c`.

##### 7.5.2.2 int lcms\_pluginl\_s::init\_argc

number of arguments for the initialization function

Definition at line 109 of file `lcmaps_pluginmanager.c`.

##### 7.5.2.3 char \* lcms\_pluginl\_s::init\_argv

list of arguments for the initialization function

Definition at line 110 of file `lcmaps_pluginmanager.c`.

**7.5.2.4 struct lcmaps\_pluginidl\_s \* lcmaps\_pluginidl\_s::next**

pointer to the next plugin in the plugin list

Definition at line 113 of file lcmaps\_pluginmanager.c.

**7.5.2.5 char lcmaps\_pluginidl\_s::pluginargs**

argument string

Definition at line 108 of file lcmaps\_pluginmanager.c.

**7.5.2.6 char lcmaps\_pluginidl\_s::pluginname**

name of plugin

Definition at line 107 of file lcmaps\_pluginmanager.c.

**7.5.2.7 lcmaps\_proc\_t lcmaps\_pluginidl\_s::procs**

list of handles to interface functions of plugin

Definition at line 106 of file lcmaps\_pluginmanager.c.

**7.5.2.8 int lcmaps\_pluginidl\_s::run\_argc**

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

Definition at line 111 of file lcmaps\_pluginmanager.c.

**7.5.2.9 lcmaps\_argument\_t \* lcmaps\_pluginidl\_s::run\_argv**

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

Definition at line 112 of file lcmaps\_pluginmanager.c.

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

- [lcmaps\\_pluginmanager.c](#)

## 7.6 lcmsaps\_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 57 of file lcmsaps\_vo\_data.h.

### 7.6.2 Field Documentation

#### 7.6.2.1 char \* lcmsaps\_vo\_data\_s::capability

the user's capability

Definition at line 63 of file lcmsaps\_vo\_data.h.

#### 7.6.2.2 char \* lcmsaps\_vo\_data\_s::group

group within the VO

Definition at line 60 of file lcmsaps\_vo\_data.h.

#### 7.6.2.3 char \* lcmsaps\_vo\_data\_s::role

the user's role

Definition at line 62 of file lcmsaps\_vo\_data.h.

#### 7.6.2.4 char \* lcmsaps\_vo\_data\_s::subgroup

subgroup name

Definition at line 61 of file lcmsaps\_vo\_data.h.

#### 7.6.2.5 char \* lcmsaps\_vo\_data\_s::vo

name of the VO to which the user belongs

Definition at line 59 of file lcmsaps\_vo\_data.h.

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

- [lcmaps\\_vo\\_data.h](#)



## 7.7 lcmaps\_vo\_mapping\_s Struct Reference

structure that contains the VO information string (or FQAN) and the local Gid it is mapped to.

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

### Data Fields

- char\* [vostring](#)
- char\* [groupname](#)
- gid\_t [gid](#)

#### 7.7.1 Detailed Description

structure that contains the VO information string (or FQAN) and the local Gid it is mapped to.

Definition at line 75 of file lcmaps\_vo\_data.h.

#### 7.7.2 Field Documentation

##### 7.7.2.1 gid\_t lcmaps\_vo\_mapping\_s::gid

GID the VO information string should be mapped to (according to the groupmapfile)

Definition at line 80 of file lcmaps\_vo\_data.h.

##### 7.7.2.2 char \* lcmaps\_vo\_mapping\_s::groupname

groupname the VO information string should be mapped to (according to the groupmapfile)

Definition at line 78 of file lcmaps\_vo\_data.h.

##### 7.7.2.3 char \* lcmaps\_vo\_mapping\_s::vostring

VO information string

Definition at line 77 of file lcmaps\_vo\_data.h.

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

- [lcmaps\\_vo\\_data.h](#)

## 7.8 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.*

#### 7.8.1 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:

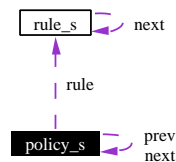
- [pdl.h](#)

## 7.9 `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.*

### 7.9.1 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`

## 7.10 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.*

### 7.10.1 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 matching 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 occurred. 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](#)

## 7.11 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.*

### 7.11.1 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](#)

## 7.12 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.*
- BOOL [okay](#)  
*TRUE if substitution can be done without further checking.*
- unsigned int [lineno](#)  
*Line number the variable appears on.*
- struct var\_s\* [next](#)  
*Next variable, or 0 if none.*

### 7.12.1 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

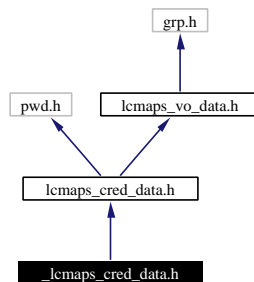
# edg-lcmaps File Documentation

### 8.1 `_lcmaps_cred_data.h` File Reference

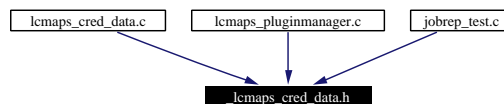
Internal header file of LCMAPS credential data.

```
#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 Steenbakkens 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 265 of file lcmaps\_cred\_data.c.

Referenced by resetCredentialData(), and stopPluginManager().

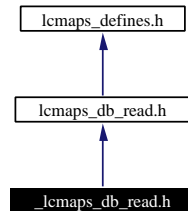


## 8.2 `_lcmaps_db_read.h` File Reference

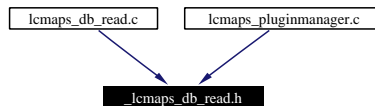
Internal header file of LCMAPS database reader.

```
#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 585 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 555 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.

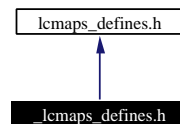
Definition at line 89 of file `lcmaps_db_read.c`.

## 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 Steenbakkens 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.

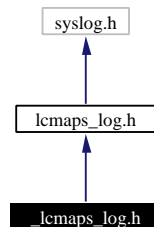
Definition at line 29 of file `_lcmaps_defines.h`.

## 8.4 `_lcmaps_log.h` File Reference

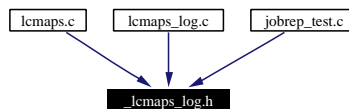
Internal header file for LCMAPS logging routines.

```
#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 317 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. It opens the logfile and tries to set the debugging level in the following order:

1. Try if `DEBUG_LEVEL > 0`
2. Try if environment variable `LCMAPS_DEBUG_LEVEL` is set and if it is an integer `> 0`
3. Otherwise set `debug_level = 0`;

#### Parameters:

*path* path of logfile.

*fp* file pointer to already opened file (or NULL)

*logtype* `DO_USRLOG`, `DO_SYSLOG`

#### Return values:

`0` succes.

*I* failure.

For internal use only.

Definition at line 81 of file lcmapi\_log.c.



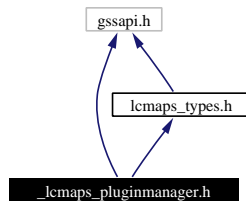
## 8.5 `lcmaps_pluginmanager.h` File Reference

API of the PluginManager.

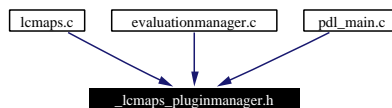
```
#include <gssapi.h>
```

```
#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.*
- `int resetCredentialData ()`  
*Reset the Credential data.*

### 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)
5. [resetCredentialData\(\)](#): reset the Credential Data. (used by Evaluation Manager)

Definition in file [\\_lcmapi\\_pluginmanager.h](#).

## 8.5.2 Function Documentation

### 8.5.2.1 `int resetCredentialData ()`

Reset the Credential data.

Wrapper around [cleanCredentialData\(\)](#) to be used by the Evaluation Manager, before a new policy is evaluated

**Returns:**

the return value of [cleanCredentialData\(\)](#)

Definition at line 957 of file `lcmapi_pluginmanager.c`.

Referenced by `pdl_next_plugin()`.

### 8.5.2.2 `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 989 of file `lcmapi_pluginmanager.c`.

### 8.5.2.3 `int runPluginManager (lcmapi_request_t request, lcmapi_cred_id_t lcmapi_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`.

#### 8.5.2.4 `int startPluginManager ()`

start the PluginManager.

start the PluginManager → load plugins, start evaluation manager

**Return values:**

*0* succes

*1* failure

Definition at line 155 of file `lcmaps_pluginmanager.c`.

Referenced by `lcmaps_init()`.

#### 8.5.2.5 `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 1045 of file `lcmaps_pluginmanager.c`.

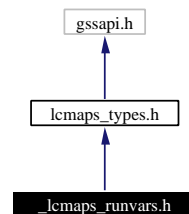
Referenced by `lcmaps_term()`.

## 8.6 `_lcmaps_runvars.h` File Reference

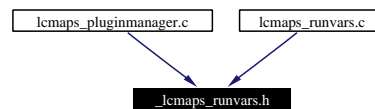
API of runvars structure.

```
#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. `lcmmaps_setRunVars()`: adds run variables to a list
3. `lcmmaps_getRunVars()`: gets run variables from list

Definition in file `lcmmaps_runvars.h`.

## 8.6.2 Function Documentation

### 8.6.2.1 `int lcmmaps_extractRunVars (lcmmaps_request_t request, lcmmaps_cred_id_t lcmmaps_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)
- lcmmaps\_cred* the credential presented by the user

#### Return values:

- 0* succes.
  - 1* failure.
- For internal use only.

Definition at line 97 of file `lcmmaps_runvars.c`.

### 8.6.2.2 `void * lcmmaps_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 `lcmmaps_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 192 of file `lcmmaps_runvars.c`.

### 8.6.2.3 `int lcmmaps_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 `lcmmaps_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.

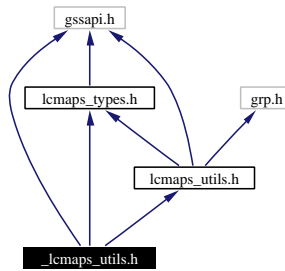
Definition at line 233 of file lcmapi\_runvars.c.

## 8.7 `_lcmaps_utils.h` File Reference

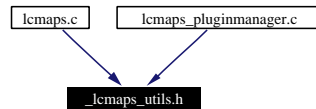
Internal header for the LCMAPS utilities.

```
#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 distinguished 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 credential from distinguished 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.

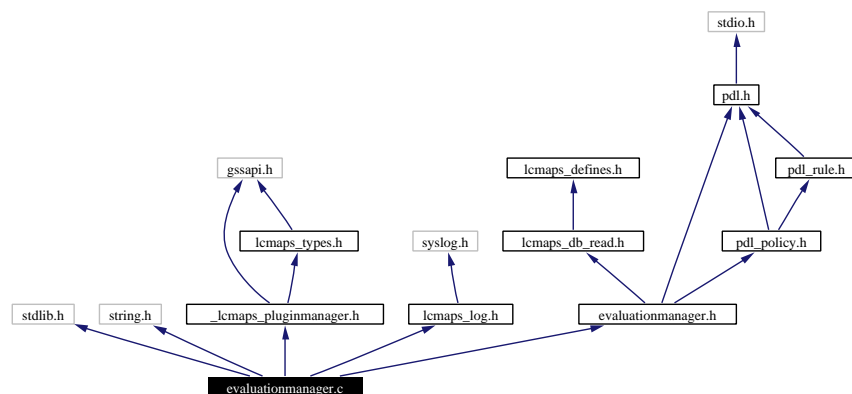
Definition at line 462 of file `lcmaps_utils.c`.

## 8.8 evaluationmanager.c File Reference

Implementation of the evaluation manager interface.

```
#include <stdlib.h>
#include <string.h>
#include "_lcmapi_pluginmanager.h"
#include "lcmapi_log.h"
#include "evaluationmanager.h"
```

Include dependency graph for evaluationmanager.c:



### Functions

- [int free\\_lcmapi\\_db\\_entry \(\)](#)
- [int startEvaluationManager \(const char \\*name\)](#)
- [int getPluginNameAndArgs \(lcmapi\\_db\\_entry\\_t \\*\\*plugins\)](#)
- [int runEvaluationManager \(void\)](#)
- [int stopEvaluationManager \(void\)](#)

### Variables

- [lcmapi\\_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](mailto:venekamp@nikhef.nl))

**Version:**

**Revision:**

1.21

**Date:**

**Date:**

2003/08/15 12:55:40

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 necessary and safe to free the resources, call this function

**Return values:**

- `0` when the call is successful,
- `1` otherwise.

Definition at line 261 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 allocated is freed during the `stopEvaluationManager()` call.

**Parameters:**

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

**Return values:**

- `0` when the call is successful,
- `1` otherwise.

Definition at line 110 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 205 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 63 of file evaluationmanager.c.

#### 8.8.2.5 `int stopEvaluationManager (void)`

Stop the evaluation manager after it has run successfully. Strictly speaking, the evaluation 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 240 of file evaluationmanager.c.

Referenced by `startEvaluationManager()`, and `stopPluginManager()`.

### 8.8.3 Variable Documentation

#### 8.8.3.1 `lcmapi_db_entry_t * global_plugin_list = NULL` [static]

When the `getPluginNameAndArgs()` function has been called, the `global_plugin_list` variable gets 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.

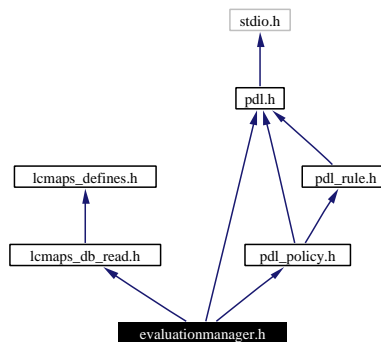
Definition at line 50 of file evaluationmanager.c.

## 8.9 evaluationmanager.h File Reference

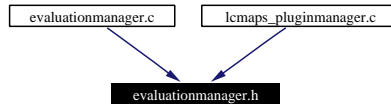
Evaluation Manager interface definition.

```
#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 delegates the necessary work to the Policy Language Description module (PDL).

#### Author:

G.M. Venekamp ([venekamp@nikhef.nl](mailto: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 (lcmapi_db_entry_t ** plugin)`

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

**Parameters:**

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

**Return values:**

*0* when the call is successful,

*1* otherwise.

Definition at line 110 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 [startEvaluationManager](#) first.

**Return values:**

*0* when the call is successful,

*1* otherwise.

Definition at line 205 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 63 of file [evaluationmanager.c](#).Referenced by [startPluginManager\(\)](#).

**8.9.2.4 int stopEvaluationManager (void)**

Stop the evaluation manager after it has run successfully. Strictly speaking, the evaluation 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.

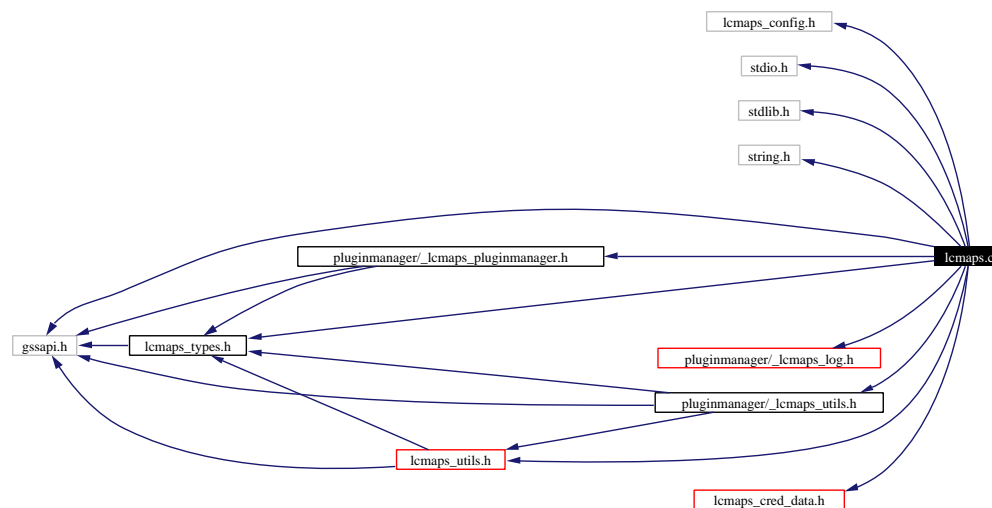
Definition at line 240 of file evaluationmanager.c.

## 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 "pluginmanager/_lcmaps_utils.h"
#include "lcmaps_cred_data.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 74 of file `lcmaps.c`.

### 8.10.2.2 `int lcmaps_initialized = 0` [static]

For internal use only.

Definition at line 75 of file `lcmaps.c`.

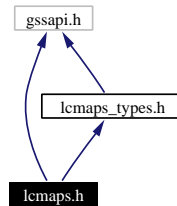
## 8.11 lcmapi.h File Reference

API of the LCMAPS library.

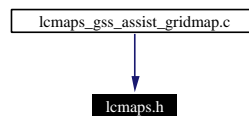
```
#include <gssapi.h>
```

```
#include "lcmapi_types.h"
```

Include dependency graph for lcmapi.h:



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



### Functions

- `int lcmapi_init (FILE *fp)`  
*Initialize the LCMAPS module.*
- `int lcmapi_term ()`  
*Terminate the LCMAPS module.*
- `int lcmapi_run (gss_cred_id_t user_cred, lcmapi_request_t request)`  
*let LCMAPS handle the user mapping.*
- `int lcmapi_run_and_return_username (gss_cred_id_t user_cred, lcmapi_request_t request, char **usernamep)`  
*let LCMAPS handle the user mapping and return user name.*
- `int lcmapi_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 104 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 174 of file `lcmaps.c`.

### 8.11.2.3 `int lcmaps_run_and_return_username (gss_cred_id_t user_cred, lcmaps_request_t request, char ** usernamep)`

let LCMAPS handle the user mapping and return user name.

do the user mapping and return user name (needed for e.g. GridFTP)

#### Parameters:

*request* authorization request in RSL (later JDL)

*user\_cred* GLOBUS user credential

*usernamep* pointer to user name

#### Return values:

*0* mapping succeeded.

*1* mapping failed.

Definition at line 259 of file lcmaps.c.

### 8.11.2.4 `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 337 of file lcmaps.c.

### 8.11.2.5 `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.

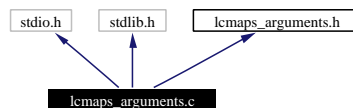
Definition at line 406 of file lcmaps.c.

## 8.12 lcmargs\_arguments.c File Reference

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

```
#include <stdio.h>
#include <stdlib.h>
#include "lcmargs_arguments.h"
```

Include dependency graph for lcmargs\_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:

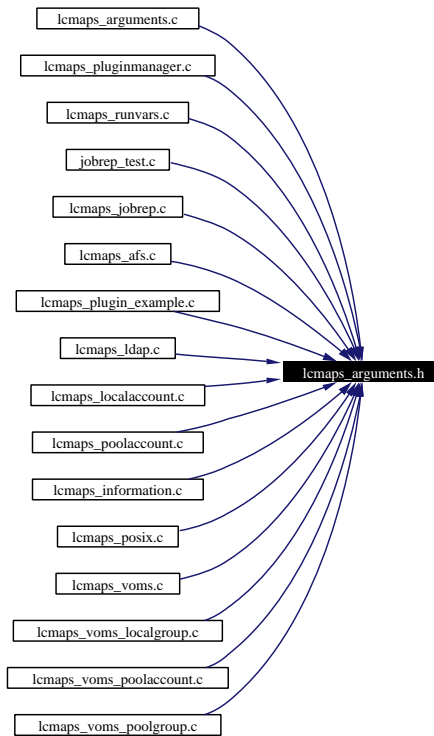
1. [lcmargs\\_setArgValue\(\)](#): Set the value of argument with name `argName` of `argType` to value
2. [lcmargs\\_getArgValue\(\)](#): Get the value of argument with name `argName` of `argType`
3. [lcmargs\\_findArgName\(\)](#): Get index of argument with name `argName`
4. [lcmargs\\_cntArgs\(\)](#): Count the number of arguments

Definition in file [lcmargs\\_arguments.c](#).

## 8.13 lcmapi\_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 [lcmapi\\_argument\\_s](#)  
*structure representing an LCMAPS plugin run argument.*

### Typedefs

- typedef struct [lcmapi\\_argument\\_s](#) [lcmapi\\_argument\\_t](#)  
*Type of LCMAPS plugin run argument (to be passed to the plugin by [plugin\\_run\(\)](#)).*

### Functions

- int [lcmapi\\_setArgValue](#) (char \*argName, char \*argType, void \*value, int argcx, [lcmapi\\_argument\\_t](#) \*\*argvx)  
*Set the value of argument with name argName of argType to value.*
- void\* [lcmapi\\_getArgValue](#) (char \*argName, char \*argType, int argcx, [lcmapi\\_argument\\_t](#) \*argvx)  
*Get the value of argument with name argName of argType.*

- `int lcmargs_findArgName` (`char *argName`, `int argcx`, `lcmargs_argument_t *argvx`)  
*Get index of argument with name argName.*
- `int lcmargs_findArgNameAndType` (`char *argName`, `char *argType`, `int argcx`, `lcmargs_argument_t *argvx`)  
*Get index of argument with name argName.*
- `int lcmargs_cntArgs` (`lcmargs_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. `lcmargs_setArgValue`(): Set the value of argument with name `argName` of `argType` to value
2. `lcmargs_getArgValue`(): Get the value of argument with name `argName` of `argType`
3. `lcmargs_findArgName`(): Get index of argument with name `argName`
4. `lcmargs_cntArgs`(): Count the number of arguments

Definition in file `lcmargs_arguments.h`.

### 8.13.2 Function Documentation

#### 8.13.2.1 `int lcmargs_cntArgs (lcmargs_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 `lcmargs_arguments.c`.

**8.13.2.2 int lcmapi\_findArgName (char \* argName, int argc, lcmapi\_argument\_t \* argv)**

Get index of argument with name argName.

Search for argName in the arguments list. Returns the index to lcmapi\_argument\_t element.

**Parameters:**

*argName* name of argument  
*argc* number of arguments  
*argv* array of arguments structures

**Returns:**

index to lcmapi\_argument\_t element

Definition at line 178 of file lcmapi\_arguments.c.

**8.13.2.3 int lcmapi\_findArgNameAndType (char \* argName, char \* argType, int argc, lcmapi\_argument\_t \* argv)**

Get index of argument with name argName.

Search for argName in the arguments list. Returns the index to lcmapi\_argument\_t element.

**Parameters:**

*argName* name of argument  
*argType* type of argument  
*argc* number of arguments  
*argv* array of arguments structures

**Returns:**

index to lcmapi\_argument\_t element

Definition at line 229 of file lcmapi\_arguments.c.

**8.13.2.4 void \* lcmapi\_getArgValue (char \* argName, char \* argType, int argc, lcmapi\_argument\_t \* argv)**

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  
*argc* number of arguments  
*argv* array of arguments structures

**Returns:**

void pointer to the value or NULL

Definition at line 130 of file lcmapi\_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

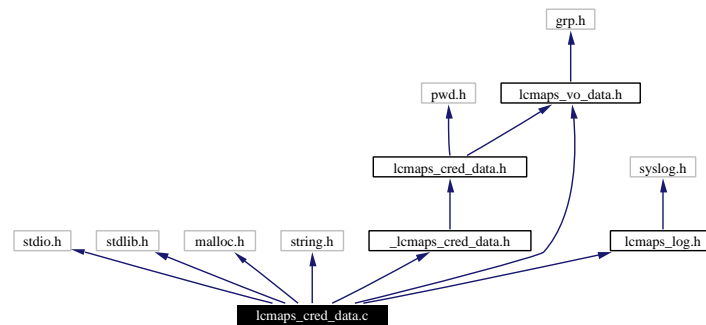
Definition at line 71 of file lcmaps\_arguments.c.

## 8.14 lcmapi\_cred\_data.c File Reference

Routines to handle lcmapi credential data.

```
#include <stdio.h>
#include <stdlib.h>
#include <malloc.h>
#include <string.h>
#include "_lcmapi_cred_data.h"
#include "lcmapi_log.h"
#include "lcmapi_vo_data.h"
```

Include dependency graph for lcmapi\_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 lcmapi credential data.

##### Author:

Oscar Koeroo and Martijn Steenbakkens for the EU DataGrid.

Definition in file [lcmapi\\_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 320 of file lcms\_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,
    (lcmaps_vo_mapping_t *) NULL,
    0,
    0,
    0,
    0,
    0,
    0,
    0,
}
```

Definition at line 41 of file lcms\_cred\_data.c.

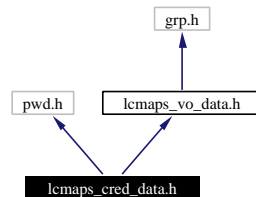
## 8.15 lcmaps\_cred\_data.h File Reference

Public header file to be used by plugins.

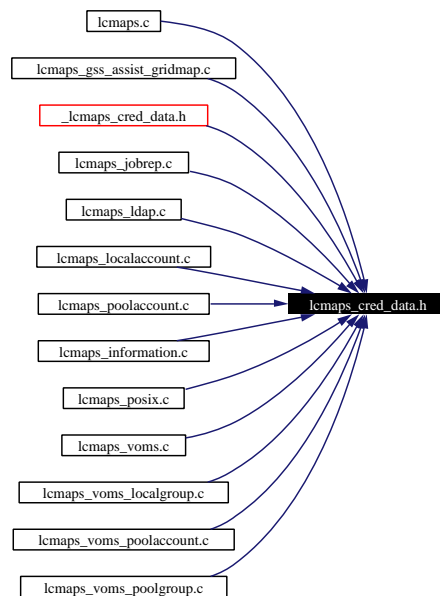
```
#include <pwd.h>
```

```
#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 84 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

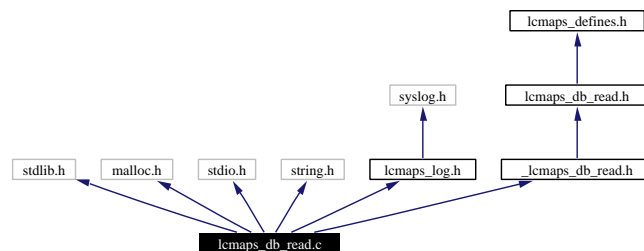
Definition at line 211 of file [lcmaps\\_cred\\_data.c](#).

## 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 list 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 Steenbakkens 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 lcmapi database file

For internal use only.

Definition at line 40 of file lcmapi\_db\_read.c.

#### 8.16.2.7 **#define PAIR\_TERMINATOR\_CHARS PAIR\_SEP\_CHARS WHITESPACE\_CHARS**

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

For internal use only.

Definition at line 52 of file lcmapi\_db\_read.c.

#### 8.16.2.8 **#define QUOTING\_CHARS ”\””**

For internal use only.

Definition at line 35 of file lcmapi\_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 lcmapi\_db\_read.c.

#### 8.16.2.10 **#define VARVAL\_TERMINATOR\_CHARS VARVAL\_SEP\_CHARS WHITESPACE\_CHARS**

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

For internal use only.

Definition at line 57 of file lcmapi\_db\_read.c.

#### 8.16.2.11 **#define WHITESPACE\_CHARS ”\t\n”**

For internal use only.

Definition at line 34 of file lcmapi\_db\_read.c.

### 8.16.3 Function Documentation

#### 8.16.3.1 **int lcmapi\_db\_parse\_line (char \* *line*, lcmapi\_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 lcms\_db\_read.c.

Referenced by lcms\_db\_read\_entries().

### 8.16.3.2 int lcms\_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 397 of file lcms\_db\_read.c.

Referenced by lcms\_db\_parse\_line().

### 8.16.3.3 int lcms\_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 494 of file lcms\_db\_read.c.

Referenced by lcms\_db\_parse\_pair().

### 8.16.3.4 int lcms\_db\_read\_entries (FILE \* *dbstream*) [static]

Read db entries from stream and fill a list 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 lcmapi\_db\_read.c.

Referenced by lcmapi\_db\_read().

## 8.16.4 Variable Documentation

### 8.16.4.1 `lcmapi_db_entry_t * lcmapi_db_list = NULL` [static]

list of database entries

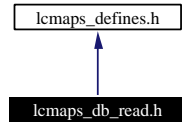
Definition at line 74 of file lcmapi\_db\_read.c.

## 8.17 lcmaps\_db\_read.h File Reference

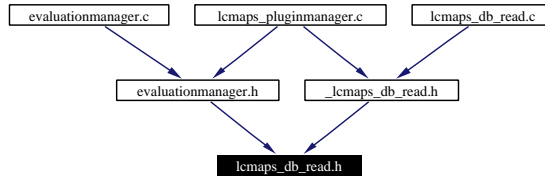
header file for LCMAPS database structure.

```
#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

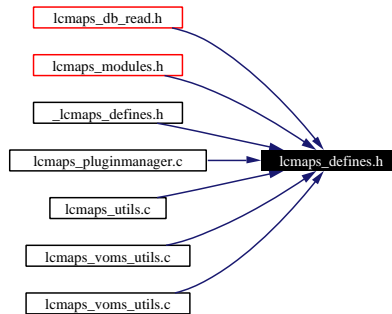
type of LCMAPS data base element.

For internal use only.

## 8.18 lcmmaps\_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](#) 2000
- #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 [lcmmaps\\_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 [lcmmaps\\_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 2000**

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 lcms\_defines.h.

**8.18.2.11 #define LCMAPS\_MOD\_SUCCESS (int)(0)**

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

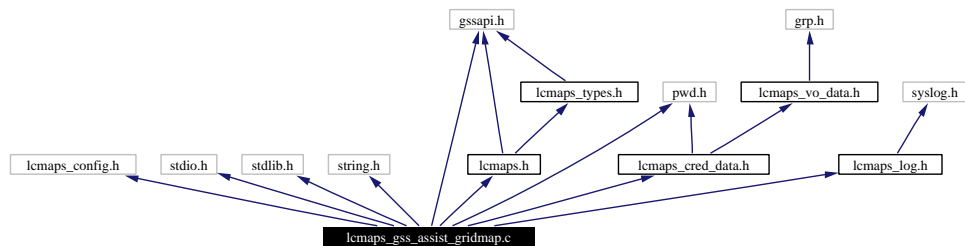
Definition at line 28 of file lcms\_defines.h.

## 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](#).

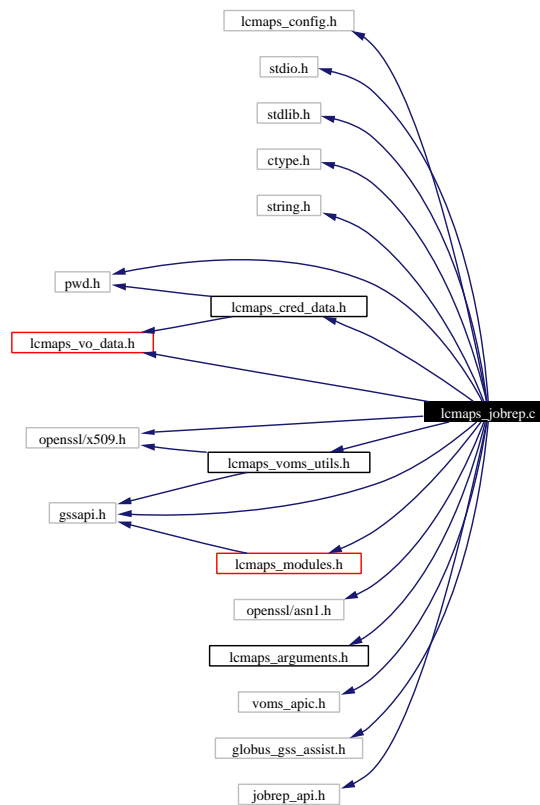


## 8.20 lcmaps\_jobrep.c File Reference

Plugin to get data for the Job Repository Database.

```
#include "lcmaps_config.h"
#include <stdio.h>
#include <stdlib.h>
#include <ctype.h>
#include <string.h>
#include <pwd.h>
#include <openssl/x509.h>
#include <openssl/asn1.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 "jobrep_api.h"
```

Include dependency graph for lcmaps\_jobrep.c:



## Functions

- `int lcmapi_get_jobrep_config (const char *, char **)`  
*Get the Job Repository configuration from file.*
- `int plugin_initialize (int argc, char **argv)`  
*initialize the plugin.*
- `int plugin_introspect (int *argc, lcmapi_argument_t **argv)`  
*Plugin asks for required arguments.*
- `int plugin_run (int argc, lcmapi_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.*

### 8.20.1 Detailed Description

Plugin to get data for the Job Repository Database.

**Author:**

Oscar Koeroo for the EU DataGrid.

Definition in file [lcmaps\\_jobrep.c](#).

## 8.20.2 Function Documentation

### 8.20.2.1 `int lcmaps_get_jobrep_config (const char * path, char ** jobrep_config) [static]`

Get the Job Repository configuration from file.

This function tries to read the Job Repository configuration from the `jobrep_config` file. It also tests if the access bits of the file are correctly set.

#### Parameters:

*path* the path to the `jobrep_config` file containing the config string.

*jobrep\_config* variable to set the configuration string in

#### Return values:

*0* success

*1* other failure

Definition at line 2170 of file `lcmaps_jobrep.c`.

Referenced by `plugin_initialize()`.

### 8.20.2.2 `int plugin_initialize (int argc, char ** argv)`

initialize the plugin.

[plugin\\_initialize](#)(int argc, char \*\*argv)

Everything that is needed to initialize the plugin should be put inside this function. Arguments as read from the 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 418 of file `lcmaps_jobrep.c`.

### 8.20.2.3 `int plugin_introspect (int * argc, lcmaps_argument_t ** argv)`

Plugin asks for required arguments.

[plugin\\_introspect](#)(int \*argc, lcmaps\_argument\_t \*\*argv)

#### 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 529 of file lcmaps\_jobrep.c.

#### 8.20.2.4 int plugin\_run (int argc, lcmaps\_argument\_t \* argv)

Gather credentials for user making use of the ordered arguments.

[plugin\\_run](#)(int argc, lcmaps\_argument\_t \*argv)

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 565 of file lcmaps\_jobrep.c.

#### 8.20.2.5 int plugin\_terminate ()

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

[plugin\\_terminate](#)()

**Return values:**

*LCMAPS\_MOD\_SUCCESS* success

*LCMAPS\_MOD\_FAIL* failure (will result in an authorization failure)

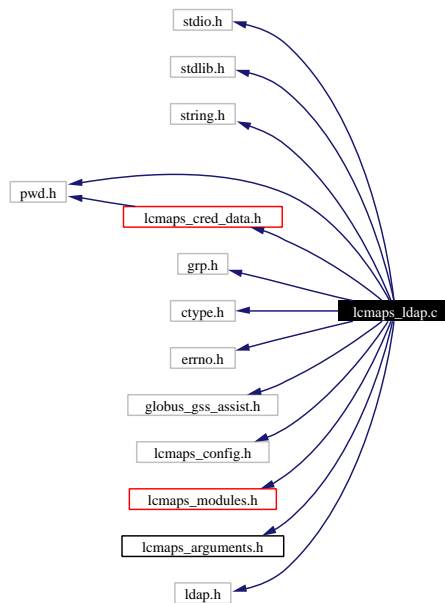
Definition at line 1925 of file lcmaps\_jobrep.c.

## 8.21 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 <errno.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:



### Defines

- #define [MAX\\_LOG\\_BUFFER\\_SIZE](#) 2048

### Functions

- int [lcmaps\\_get\\_ldap\\_pw](#) (const char \*path, char \*\*ldap\_passwd)

*Get the LDAP password from file.*

- `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_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.*

## 8.21.1 Detailed Description

Interface to the LCMAPS plugins.

### Author:

Wim Som de Cerff and Martijn Steenbakkens 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`.

## 8.21.2 Define Documentation

### 8.21.2.1 `#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 125 of file `lcmaps_ldap.c`.

## 8.21.3 Function Documentation

### 8.21.3.1 `int lcmaps_add_username_to_ldapgroup` (const char \* username, const char \* groupname, gid\_t groupnumber, LDAP \* ld\_handle, const char \* searchBase) [static]

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 1016 of file lcmsaps\_ldap.c.

**8.21.3.2 int lcmsaps\_get\_ldap\_pw (const char \* path, char \*\* ldap\_passwd) [static]**

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 1363 of file lcmsaps\_ldap.c.

**8.21.3.3 int lcmsaps\_set\_pgid (const char \* username, const char \* pgroupname, gid\_t pgroupnumber, LDAP \* ld\_handle, const char \* searchBase) [static]**

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 1255 of file lcmsaps\_ldap.c.

## 8.21.4 Variable Documentation

### 8.21.4.1 struct timeval timeout [static]

**Initial value:**

```
{
    (time_t) 0,
    (suseconds_t) 0
}
```

Definition at line 177 of file lcmaps\_ldap.c.

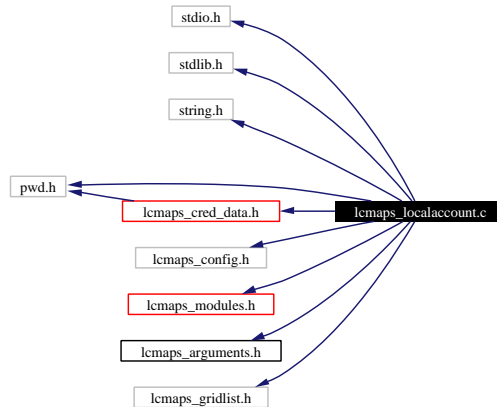


## 8.22 lcmaps\_localaccount.c File Reference

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:



### 8.22.1 Detailed Description

Interface to the LCMAPS plugins.

**Author:**

Martijn Steenbakkens for the EU DataGrid.

This file contains the code for localaccount plugin

1. [plugin\\_initialize\(\)](#)
2. [plugin\\_run\(\)](#)
3. [plugin\\_terminate\(\)](#)
4. [plugin\\_introspect\(\)](#)

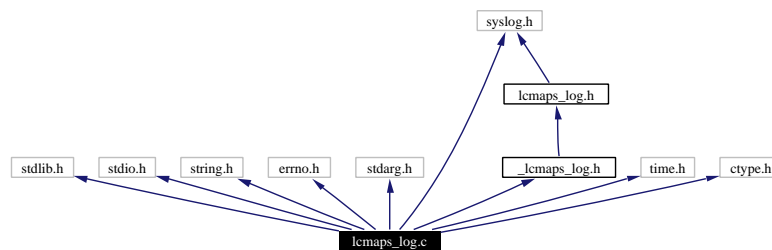
Definition in file [lcmaps\\_localaccount.c](#).

## 8.23 lcmapi\_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 <ctype.h>
#include "_lcmapi_log.h"
```

Include dependency graph for lcmapi\_log.c:



### Defines

- #define `DEBUG_LEVEL` 0

### Variables

- FILE\* `lcmapi_logfp` = NULL
- int `logging_usrlog` = 0
- int `logging_syslog` = 0
- int `debug_level` = 0
- char\* `extra_logstr` = NULL

### 8.23.1 Detailed Description

Logging routines for LCMAPS.

#### Author:

Martijn Steenbakkers for the EU DataGrid.

Definition in file `lcmapi_log.c`.

## 8.23.2 Define Documentation

### 8.23.2.1 #define DEBUG\_LEVEL 0

default debugging level

Definition at line 35 of file lcms\_log.c.

## 8.23.3 Variable Documentation

### 8.23.3.1 int debug\_level = 0 [static]

debugging level

For internal use only.

Definition at line 45 of file lcms\_log.c.

### 8.23.3.2 char \* extra\_logstr = NULL [static]

string to be included in every log statement

For internal use only.

Definition at line 46 of file lcms\_log.c.

### 8.23.3.3 FILE \* lcms\_logfp = NULL [static]

logfile descriptor.

For internal use only.

Definition at line 41 of file lcms\_log.c.

### 8.23.3.4 int logging\_syslog = 0 [static]

flag to use syslog

For internal use only.

Definition at line 43 of file lcms\_log.c.

### 8.23.3.5 int logging\_usrlog = 0 [static]

flag to do user logging

For internal use only.

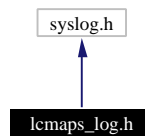
Definition at line 42 of file lcms\_log.c.

## 8.24 lcmaps\_log.h File Reference

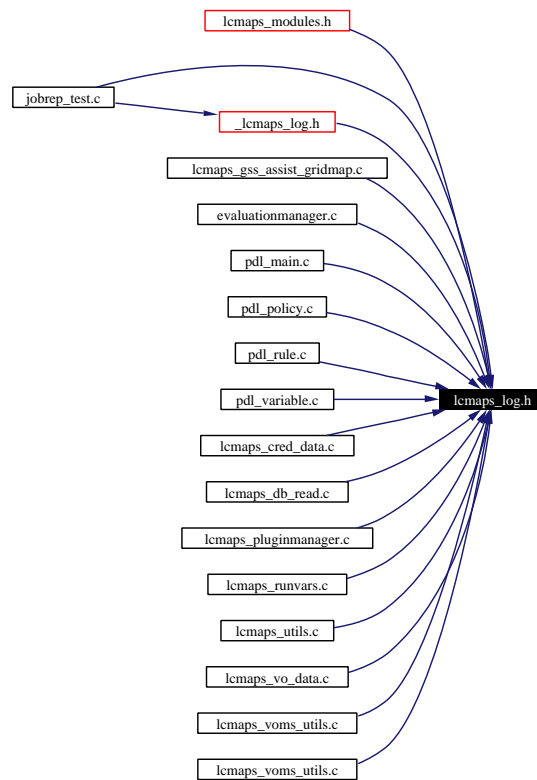
Logging API for the LCMAPS plugins and LCMAPS itself.

```
#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.*

### 8.24.1 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`.

### 8.24.2 Function Documentation

#### 8.24.2.1 `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 210 of file `lcmaps_log.c`.

#### 8.24.2.2 `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 277 of file lcmaps\_log.c.

**8.24.2.3 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.

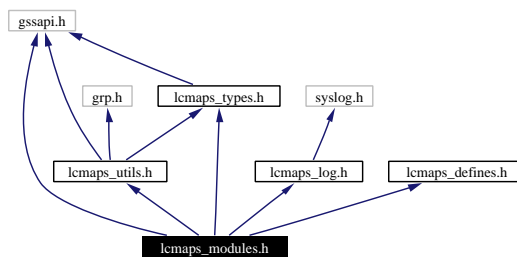
Definition at line 363 of file lcmaps\_log.c.

## 8.25 lcms\_modules.h File Reference

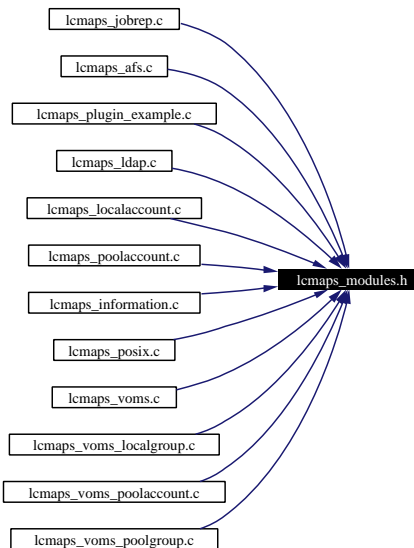
The LCMAPS authorization plugins/modules should "include" this file.

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

Include dependency graph for lcms\_modules.h:



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



### 8.25.1 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 [lcmaphs\\_modules.h](#).

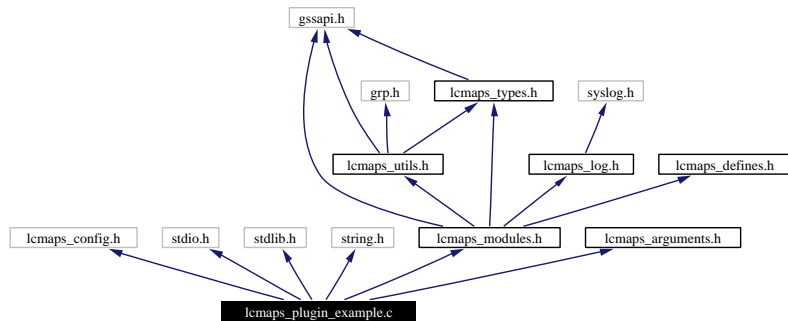


## 8.26 lcmaps\_plugin\_example.c File Reference

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:



### 8.26.1 Detailed Description

Interface to the LCMAPS plugins.

**Author:**

Gerben Venekamp 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\(\)](#)

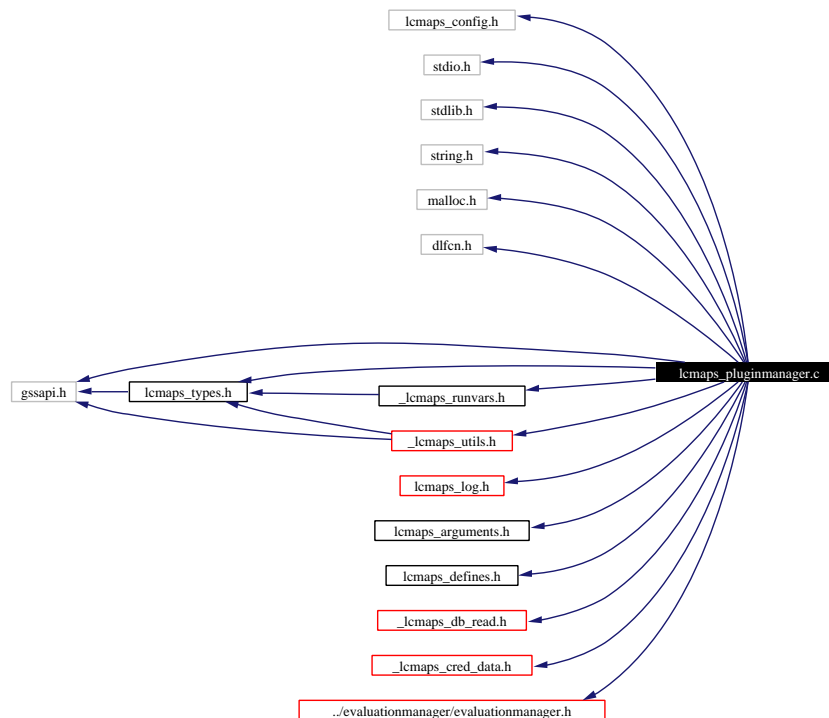
Definition in file [lcmaps\\_plugin\\_example.c](#).

## 8.27 lcmapi\_pluginmanager.c File Reference

the plugin manager for LCMAPS.

```
#include "lcmapi_config.h"
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <malloc.h>
#include <dlfcn.h>
#include <gssapi.h>
#include "lcmapi_types.h"
#include "lcmapi_log.h"
#include "lcmapi_arguments.h"
#include "lcmapi_defines.h"
#include "_lcmapi_utils.h"
#include "_lcmapi_db_read.h"
#include "_lcmapi_runvars.h"
#include "_lcmapi_cred_data.h"
#include "../evaluationmanager/evaluationmanager.h"
```

Include dependency graph for lcmapi\_pluginmanager.c:



## Data Structures

- struct `lcmaps_pluginidl_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_pluginidl_s` `lcmaps_pluginidl_t`  
*the type definition of the lcmaps plugin module structure.*

## Enumerations

- enum `lcmaps_proctype_e` { `INITPROC`, `RUNPROC`, `TERMPROC`, `INTROPROC`, `ENDOFFPROCS` }
- This enumeration type gives the different plugin symbol/function types.*

## Functions

- `lcmaps_pluginidl_t*` `PluginInit` (`lcmaps_db_entry_t *`, `lcmaps_pluginidl_t **`)  
*Initialize the plugin.*
- `lcmaps_proc_t` `get_procsymbol` (`void *`, `char *`)  
*get procedure symbol from dlopen-ed library.*
- int `print_lcmaps_plugin` (`int`, `lcmaps_pluginidl_t *`)  
*print the lcmaps\_pluginidl\_t structure.*
- int `parse_args_plugin` (`const char *`, `const char *`, `char **`, `int *`)  
*convert plugin argument string into xargc, xargv.*
- int `clean_plugin_list` (`lcmaps_pluginidl_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_pluginidl_t*` `plugin_list` = `NULL`

## 8.27.1 Detailed Description

the plugin manager for LCMAPS.

### Author:

Martijn Steenbakkens 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)
6. `resetCredentialData()`: reset the Credential Data. (used by Evaluation Manager)

Definition in file `lcmapi_pluginmanager.c`.

## 8.27.2 Define Documentation

### 8.27.2.1 `#define MAXPROCS 4`

maximum number of interface symbols in plugin modules

For internal use only.

Definition at line 65 of file `lcmapi_pluginmanager.c`.

### 8.27.2.2 `#define NUL '\0'`

NUL character

For internal use only.

Definition at line 61 of file `lcmapi_pluginmanager.c`.

## 8.27.3 Typedef Documentation

### 8.27.3.1 `typedef struct lcmapi_pluginInfo_t lcmapi_pluginInfo_t`

the type definition of the lcmapi plugin module structure.

For internal use only.

### 8.27.3.2 `typedef int(* lcmapi_proc_t)()`

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

For internal use only.

Definition at line 91 of file `lcmapi_pluginmanager.c`.

## 8.27.4 Enumeration Type Documentation

### 8.27.4.1 enum lcms\_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 77 of file lcms\_pluginmanager.c.

## 8.27.5 Function Documentation

### 8.27.5.1 `lcmaps_plugin_t * PluginInit (lcmaps_db_entry_t * db_handle, lcmaps_plugin_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 lcms\_pluginmanager.c.

Referenced by startPluginManager().

### 8.27.5.2 `int clean_plugin_list (lcmaps_plugin_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 freed.

#### Return values:

- 0 succes.

*I* failure.

For internal use only.

Definition at line 781 of file lcmaps\_pluginmanager.c.

Referenced by startPluginManager(), and stopPluginManager().

### 8.27.5.3 `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().

### 8.27.5.4 `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.

*I* failure.

For internal use only.

Definition at line 578 of file lcmaps\_pluginmanager.c.

Referenced by PluginInit().

### 8.27.5.5 `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 lcms\_pluginmanager.c.

Referenced by runPluginManager(), and startPluginManager().

## 8.27.6 Variable Documentation

**8.27.6.1** `char * lcms_db_file_default = NULL` [static]

For internal use only.

Definition at line 129 of file lcms\_pluginmanager.c.

**8.27.6.2** `char * lcms_dir = NULL` [static]

For internal use only.

Definition at line 130 of file lcms\_pluginmanager.c.

**8.27.6.3** `lcms_plugin_t * plugin_list = NULL` [static]

For internal use only.

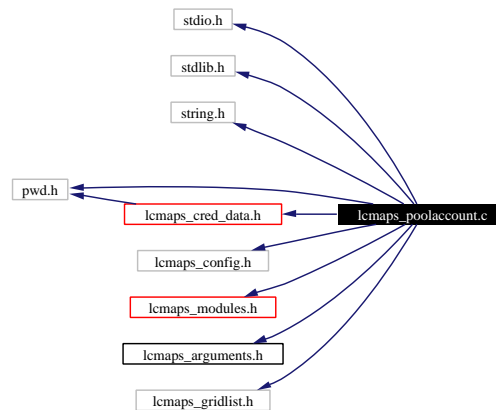
Definition at line 131 of file lcms\_pluginmanager.c.

## 8.28 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:



### 8.28.1 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](#).

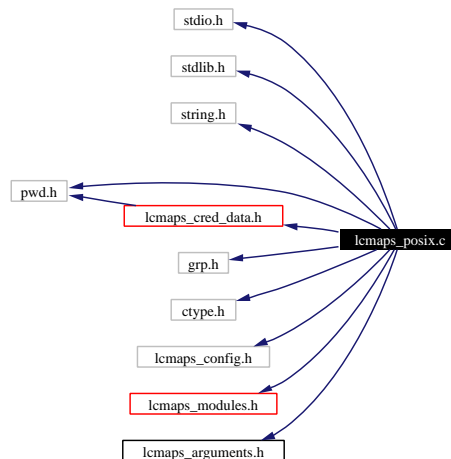


## 8.29 lcmaps\_posix.c File Reference

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:



### 8.29.1 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\(\)](#)

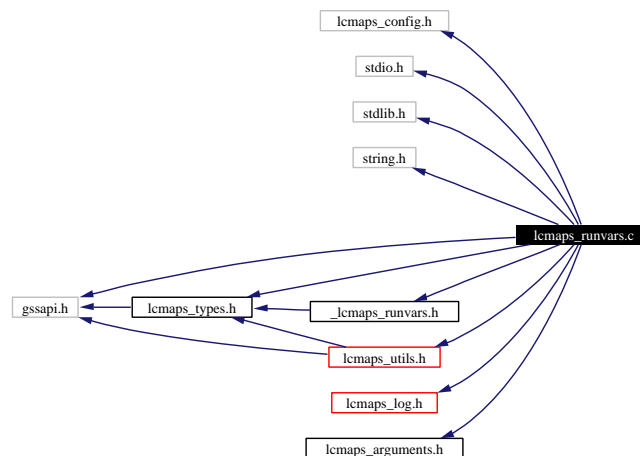
Definition in file [lcmaps\\_posix.c](#).

## 8.30 lcmapi\_runvars.c File Reference

Extract variables that will be used by the plugins.

```
#include "lcmapi_config.h"
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <gssapi.h>
#include "lcmapi_log.h"
#include "lcmapi_types.h"
#include "lcmapi_utils.h"
#include "lcmapi_arguments.h"
#include "_lcmapi_runvars.h"
```

Include dependency graph for lcmapi\_runvars.c:



### Variables

- [lcmapi\\_argument\\_t runvars\\_list](#) []

### 8.30.1 Detailed Description

Extract variables that will be used by the plugins.

#### Author:

Martijn Steenbakkens 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. `lcm_extractRunVars()`: takes the global credential and Job request and extracts run variables from them
2. `lcm_setRunVars()`: adds run variables to a list
3. `lcm_getRunVars()`: gets run variables from list  
For internal use only.

Definition in file `lcm_runvars.c`.

## 8.30.2 Variable Documentation

### 8.30.2.1 `lcm_argument_t runvars_list` [static]

**Initial value:**

```
{
  { "user_dn"      , "char *"      , 0, NULL},
  { "user_cred"   , "gss_cred_id_t" , 0, NULL},
  { "lcm_cred"    , "lcm_cred_id_t" , 0, NULL},
  { "job_request" , "lcm_request_t" , 0, NULL},
  { "job_request" , "char *"      , 0, NULL},
  { NULL         , NULL         , -1, NULL}
}
```

For internal use only.

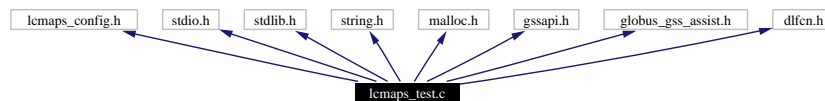
Definition at line 58 of file `lcm_runvars.c`.

## 8.31 lcmapi\_test.c File Reference

Program to test the LCMAPS and its plugins.

```
#include "lcmapi_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 lcmapi\_test.c:



### 8.31.1 Detailed Description

Program to test the LCMAPS and its plugins.

**Author:**

Martijn Steenbakkens 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 `./lcmapi-test` It is not possible (yet) to feed a user credential (proxy) to the program.

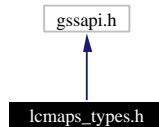
Definition in file [lcmapi\\_test.c](#).

## 8.32 lcms\_types.h File Reference

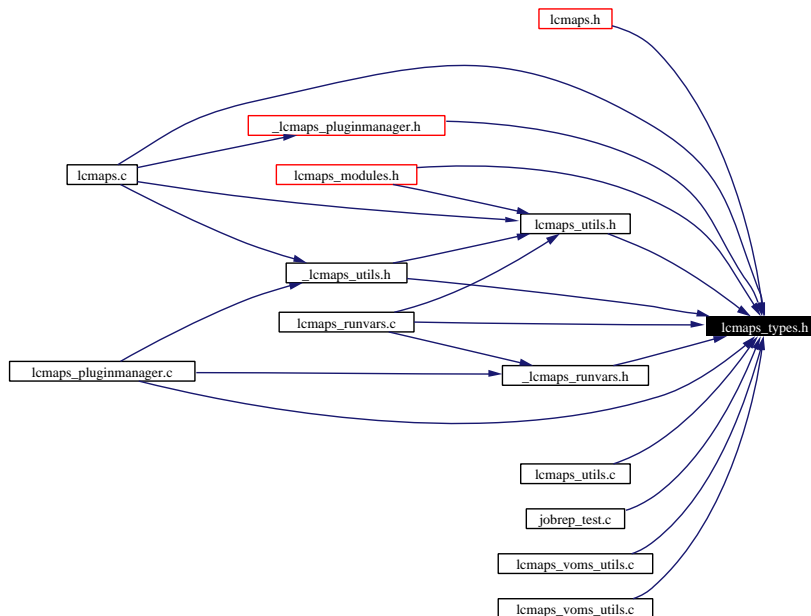
Public header file with typedefs for LCMAPS.

```
#include <gssapi.h>
```

Include dependency graph for lcms\_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.*

### 8.32.1 Detailed Description

Public header file with typedefs for LCMAPS.

**Author:**

Martijn Steenbakkers for the EU DataGrid.

Definition in file [lcmaps\\_types.h](#).

### 8.32.2 Typedef Documentation

#### 8.32.2.1 typedef char \* lcmaps\_request\_t

Type of the LCMAPS request expressed in RSL/JDL.

(Internal) just a string.

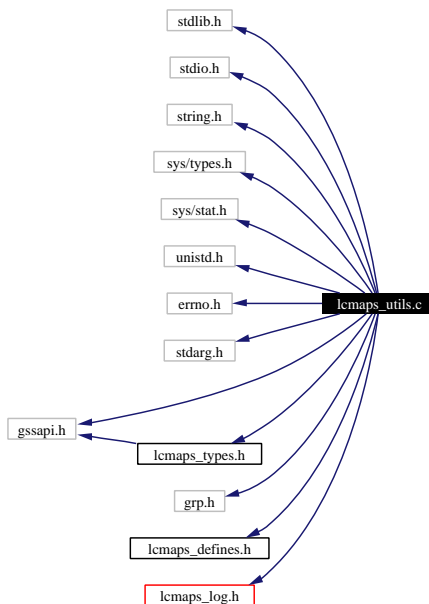
Definition at line 37 of file lcmaps\_types.h.

## 8.33 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>.*

### 8.33.1 Detailed Description

the utilities for the LCMAPS.

**Author:**

Martijn Steenbakkens for the EU DataGrid.

Definition in file [lcmaps\\_utils.c](#).

### 8.33.2 Function Documentation

#### 8.33.2.1 `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()`.

#### 8.33.2.2 `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()`.

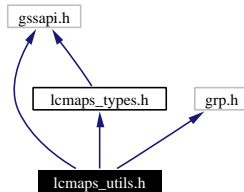


## 8.34 lcms\_apis.h File Reference

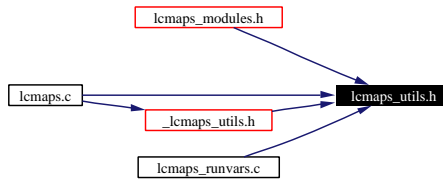
API for the utilities for the LCMAPS.

```
#include <gssapi.h>
#include "lcmapi_types.h"
#include <grp.h>
```

Include dependency graph for lcmapi\_apis.h:



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



## CREDENTIAL FUNCTIONS

- char\* [lcmapi\\_get\\_dn](#) (lcmapi\_cred\_id\_t lcmapi\_credential)  
*Retrieve user DN from (LCMAPS) credential.*

## FILENAME FUNCTIONS

- char\* [lcmapi\\_genfilename](#) (char \*prefix, char \*path, char \*suffix)  
*Generate an absolute file name.*
- char\* [lcmapi\\_getfexist](#) (int n,...)  
*Picks the first existing file in argument list.*
- char\* [lcmapi\\_findfile](#) (char \*name)  
*Checks for file in standard directories.*

## Functions

- int [lcmapi\\_get\\_gidlist](#) (const char \*username, int \*ngroups, gid\_t \*\*group\_list)  
*Finds the list of gids for user in the group file (/etc/group).*

### 8.34.1 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](#).

### 8.34.2 Function Documentation

#### 8.34.2.1 `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 389 of file [lcmaps\\_utils.c](#).

#### 8.34.2.2 `char * lcmaps_genfilename (char * prefix, 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 lcms\_apis.c.

### 8.34.2.3 char \* lcms\_get\_dn (lcms\_cred\_id\_t lcms\_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:**

*lcms\_cred* the LCMAPS credential

**Returns:**

a string containing the user DN

Definition at line 151 of file lcms\_apis.c.

### 8.34.2.4 int lcms\_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 577 of file lcms\_apis.c.

### 8.34.2.5 char \* lcms\_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

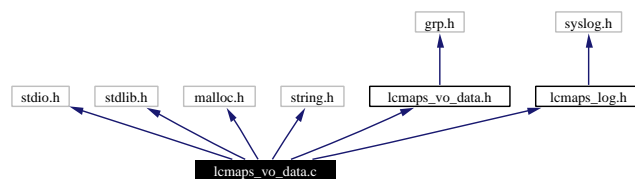
Definition at line 347 of file lcmapi\_utils.c.

## 8.35 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:



### 8.35.1 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

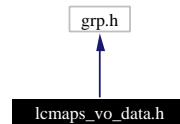
Definition in file `lcmaps_vo_data.c`.

## 8.36 lcmaphs\_vo\_data.h File Reference

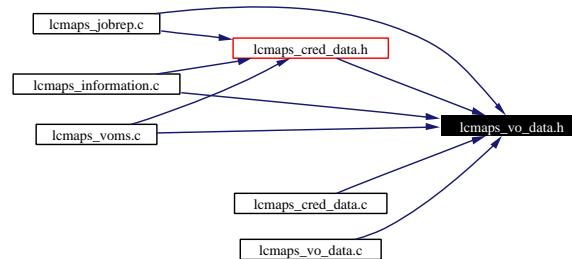
LCMAPS module for creating and accessing VO data structures.

```
#include <grp.h>
```

Include dependency graph for lcmaphs\_vo\_data.h:



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



### Data Structures

- struct [lcmaphs\\_vo\\_data\\_s](#)  
*structure that contains the VO information found in the user's gss credential.*
- struct [lcmaphs\\_vo\\_mapping\\_s](#)  
*structure that contains the VO information string (or FQAN) and the local Gid it is mapped to.*

### Defines

- #define [LCMAPS\\_NO\\_GID](#) (gid\_t)(-1)

### Typedefs

- typedef struct [lcmaphs\\_vo\\_data\\_s](#) [lcmaphs\\_vo\\_data\\_t](#)  
*Type of VO information structure.*
- typedef struct [lcmaphs\\_vo\\_mapping\\_s](#) [lcmaphs\\_vo\\_mapping\\_t](#)  
*Type of VO mapping structure.*

## 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.*
- [lcmaps\\_vo\\_mapping\\_t\\*](#) [lcmaps\\_createVoMapping](#) (const char \*vo\_data\_string, const char \*groupname, const gid\_t gid)  
*Create a VoMapping structure.*
- int [lcmaps\\_deleteVoMapping](#) ([lcmaps\\_vo\\_mapping\\_t](#) \*\*vo\_mapping)  
*Delete a VoMapping structure.*
- int [lcmaps\\_cleanVoMapping](#) ([lcmaps\\_vo\\_mapping\\_t](#) \*vo\_mapping)  
*Clean a VoMapping structure.*
- int [lcmaps\\_copyVoMapping](#) ([lcmaps\\_vo\\_mapping\\_t](#) \*dst\_vo\_mapping, const [lcmaps\\_vo\\_mapping\\_t](#) \*src\_vo\_mapping)  
*Copy a VoMapping structure into an empty VoMapping structure.*
- int [lcmaps\\_printVoMapping](#) (int debug\_level, const [lcmaps\\_vo\\_mapping\\_t](#) \*vo\_mapping)  
*Print the contents of a VoMapping structure.*

### 8.36.1 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\\_cleanVoData\(\)](#): clean a VoData structure
4. [lcmaps\\_copyVoData\(\)](#): copy (the contents of) a VoData structure
5. [lcmaps\\_printVoData\(\)](#): print the contents of a VoData structure
6. [lcmaps\\_stringVoData\(\)](#): cast a VoData structure into a string
7. [lcmaps\\_createVoMapping\(\)](#): create a VoMapping structure
8. [lcmaps\\_deleteVoMapping\(\)](#): delete a VoMapping structure
9. [lcmaps\\_cleanVoMapping\(\)](#): clean a VoMapping structure
10. [lcmaps\\_copyVoMapping\(\)](#): copy (the contents of) a VoMapping structure
11. [lcmaps\\_printVoMapping\(\)](#): print the contents of a VoMapping structure

Definition in file [lcmaps\\_vo\\_data.h](#).

## 8.36.2 Define Documentation

### 8.36.2.1 `#define LCMAPS_NO_GID (gid_t)(-1)`

Invalid GID

Definition at line 37 of file [lcmaps\\_vo\\_data.h](#).

## 8.36.3 Function Documentation

### 8.36.3.1 `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 195 of file [lcmaps\\_vo\\_data.c](#).

### 8.36.3.2 `int lcmaps_cleanVoMapping (lcmaps_vo_mapping_t * vo_mapping)`

Clean a VoMapping structure.

Clean a VoMapping structure that was previously filled with [lcmaps\\_copyVoMapping\(\)](#). The contents are freed and set to zero.

#### Parameters:

*vo\_mapping* a pointer to a VoMapping structure



**Return values:**

- 0* in case of success
- 1* in case of failure

Definition at line 682 of file lcmsaps\_vo\_data.c.

**8.36.3.3 int lcmsaps\_copyVoData (lcmaps\_vo\_data\_t \* dst\_vo\_data, const lcmsaps\_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 263 of file lcmsaps\_vo\_data.c.

**8.36.3.4 int lcmsaps\_copyVoMapping (lcmaps\_vo\_mapping\_t \* dst\_vo\_mapping, const lcmsaps\_vo\_mapping\_t \* src\_vo\_mapping)**

Copy a VoMapping structure into an empty VoMapping structure.

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

**Parameters:**

- dst\_vo\_mapping* pointer to a empty VoMapping structure that should be filled
- src\_vo\_mapping* pointer to the VoMapping structure that should be copied

**Return values:**

- 0* success
- 1* failure (either src\_vo\_mapping or dst\_vo\_mapping was empty)

Definition at line 736 of file lcmsaps\_vo\_data.c.

**8.36.3.5 lcmsaps\_vo\_data\_t \* lcmsaps\_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 81 of file lcmaps\_vo\_data.c.

**8.36.3.6** `lcmaps_vo_mapping_t * lcmaps_createVoMapping (const char * vo_data_string, const char * groupname, const gid_t gid)`

Create a VoMapping structure.

Create a VoMapping structure store the VO information string (or FQAN) in combination with the corresponding gid. Allocate the memory. To be freed with `lcmaps_deleteVoMapping()`.

**Parameters:**

*vo\_data\_string* the VO information string (or FQAN)

*groupname* the (POSIX) group name

*gid* the (POSIX) GID

**Returns:**

pointer to the VoMapping structure or NULL

Definition at line 577 of file lcmaps\_vo\_data.c.

**8.36.3.7** `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 141 of file lcmaps\_vo\_data.c.

**8.36.3.8** `int lcmaps_deleteVoMapping (lcmaps_vo_mapping_t ** vo_mapping)`

Delete a VoMapping structure.

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

**Parameters:**

*vo\_mapping* pointer to a pointer to a VoMapping structure

**Return values:**

- 0* in case of success
- 1* in case of failure

Definition at line 631 of file lcmaps\_vo\_data.c.

**8.36.3.9 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 324 of file lcmaps\_vo\_data.c.

**8.36.3.10 int lcmaps\_printVoMapping (int *debug\_level*, const lcmaps\_vo\_mapping\_t \* *vo\_mapping*)**

Print the contents of a VoMapping structure.

**Parameters:**

- vo\_mapping* pointer to a VoMapping structure
- debug\_level* debug\_level for which the contents will be printed

**Returns:**

- 0 (always)

Definition at line 787 of file lcmaps\_vo\_data.c.

**8.36.3.11 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

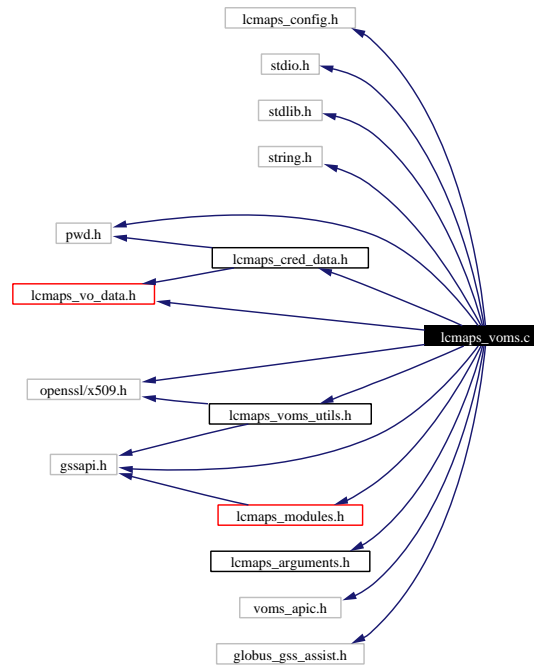
Definition at line 392 of file lcmaps\_vo\_data.c.

## 8.37 lcmapi\_voms.c File Reference

Interface to the LCMAPS plugins.

```
#include "lcmapi_config.h"
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <pwd.h>
#include <openssl/x509.h>
#include "gssapi.h"
#include "lcmapi_modules.h"
#include "lcmapi_arguments.h"
#include "lcmapi_cred_data.h"
#include "lcmapi_voms_utils.h"
#include "lcmapi_vo_data.h"
#include "voms_apic.h"
#include "globus_gss_assist.h"
```

Include dependency graph for lcmapi\_voms.c:



### 8.37.1 Detailed Description

Interface to the LCMAPS plugins.

**Author:**

Martijn Steenbakkens 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\(\)](#)

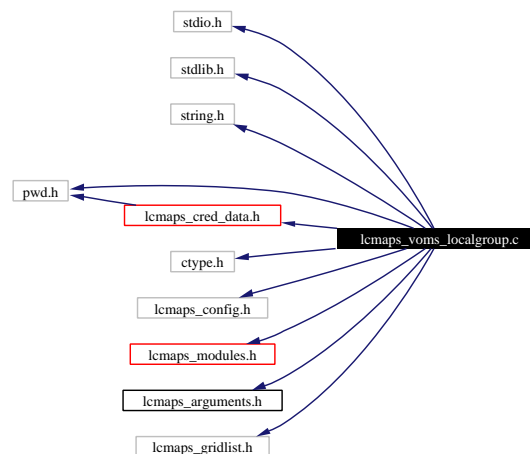
Definition in file [lcmaps\\_voms.c](#).

## 8.38 lcmaps\_voms\_localgroup.c File Reference

Interface to the LCMAPS plugins.

```
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <pwd.h>
#include <ctype.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:



### 8.38.1 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\(\)](#)

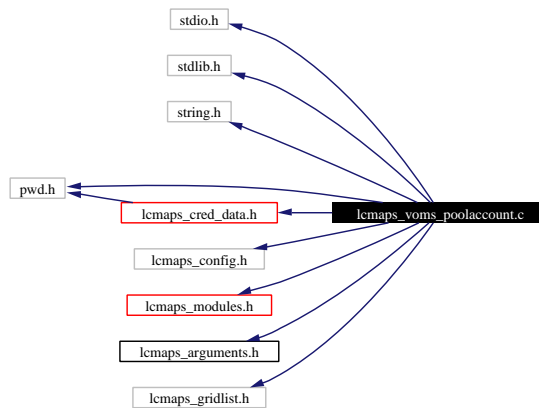
Definition in file [lcmaps\\_voms\\_localgroup.c](#).

## 8.39 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:



### 8.39.1 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\(\)](#)

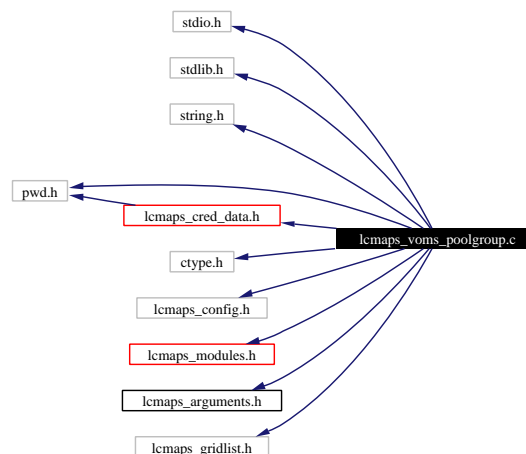
Definition in file [lcmaps\\_voms\\_poolaccount.c](#).

## 8.40 lcmaps\_voms\_poolgroup.c File Reference

Interface to the LCMAPS plugins.

```
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <pwd.h>
#include <ctype.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:



### 8.40.1 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](#).



## 8.41 pdl.h File Reference

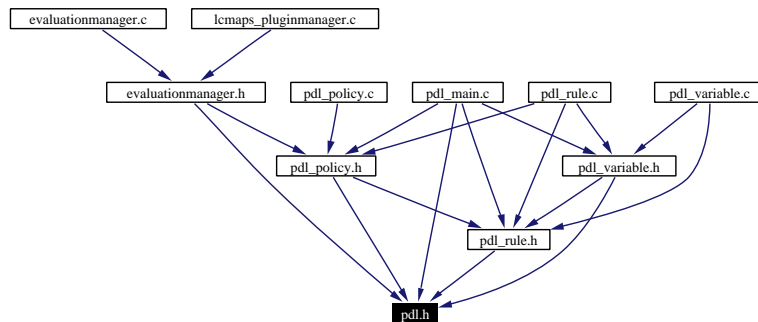
General include file.

```
#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 \*)
- void `set_path` (record\_t \*\_path)
- record\_t\* `concat_strings` (record\_t \*s1, record\_t \*s2)
- record\_t\* `concat_strings_with_space` (record\_t \*s1, record\_t \*s2)
- const plugin\_t\* `get_plugins` (void)
- void `warning` (pdl\_error\_t error, const char \*s,...)
- void `free_resources` (void)
- const char\* `pdl_next_plugin` (plugin\_status\_t status)

## Variables

- unsigned int `lineno` = 1  
*The first line of a configuration script is labeled 1.*

### 8.41.1 Detailed Description

General include file.

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

**Author:**

G.M. Venekamp ([venekamp@nikhef.nl](mailto:venekamp@nikhef.nl))

**Version:**

**Revision:**

1.14

**Date:**

**Date:**

2003/09/11 11:20:28

Definition in file `pdl.h`.

## 8.41.2 Define Documentation

### 8.41.2.1 #define TRUE 1

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

Definition at line 44 of file pdl.h.

## 8.41.3 Typedef Documentation

### 8.41.3.1 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.

### 8.41.3.2 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 matching 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 occurred. This allows for easier debugging of the configuration file.

## 8.41.4 Enumeration Type Documentation

### 8.41.4.1 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.

### 8.41.4.2 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 unsuccessful.

Definition at line 65 of file pdl.h.

## 8.41.5 Function Documentation

### 8.41.5.1 `record_t * concat_strings (record_t * s1, record_t * s2)`

Concatenate two strings. The original two strings are freed. When the concatenation fails, the original 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 393 of file `pdl_main.c`.

### 8.41.5.2 `record_t * concat_strings_with_space (record_t * s1, record_t * s2)`

Concatenate two strings. The original two strings are freed. When the concatenation fails, the original 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 448 of file `pdl_main.c`.

### 8.41.5.3 `void free_resources (void)`

Free the resources.

Definition at line 612 of file `pdl_main.c`.

Referenced by `stopEvaluationManager()`.

### 8.41.5.4 `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 134 of file `pdl_main.c`.

Referenced by `getPluginNameAndArgs()`.

#### 8.41.5.5 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 74 of file pdl\_main.c.

#### 8.41.5.6 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 498 of file pdl\_main.c.

#### 8.41.5.7 const char \* pdl\_path (void)

Get the path.

**Returns:**

Path.

Definition at line 306 of file pdl\_main.c.

Referenced by getPluginNameAndArgs(), and pdl\_next\_plugin().

#### 8.41.5.8 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 336 of file pdl\_main.c.

**8.41.5.9 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 648 of file pdl\_main.c.

**8.41.5.10 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 319 of file pdl\_main.c.

**8.41.5.11 int yyparse\_errors (void)**

Tell if there were errors/warning during parsing.

**Returns:**

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

Definition at line 122 of file pdl\_main.c.

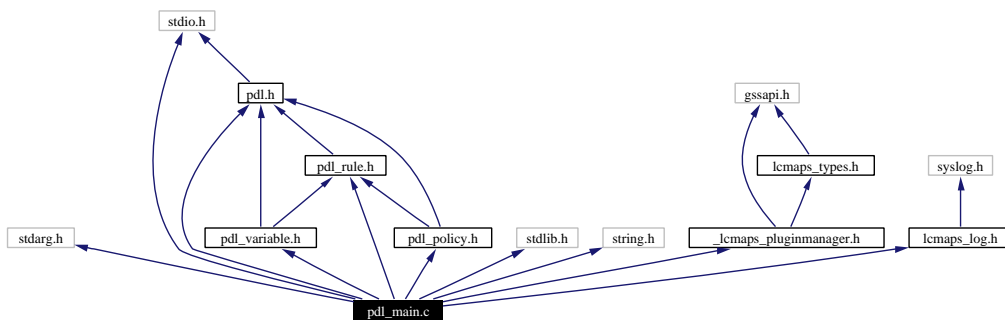
Referenced by `startEvaluationManager()`.

## 8.42 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_pluginmanager.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, const char \*extra)
- void [reduce\\_policies](#) (void)
- BOOL [plugin\\_exists](#) (const char \*string)
- int [find\\_first\\_space](#) (const char \*string)
- int [pdl\\_init](#) (const char \*name)
- int [yyparse\\_errors](#) (void)
- const [plugin\\_t\\*](#) [get\\_plugins](#) (void)
- 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)
- [record\\_t\\*](#) [concat\\_strings\\_with\\_space](#) ([record\\_t](#) \*s1, [record\\_t](#) \*s2)
- const char\* [pdl\\_next\\_plugin](#) ([plugin\\_status\\_t](#) status)
- 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 scipt is labeled 1.*

### 8.42.1 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](mailto:venekamp@nikhef.nl))

**Version:**

**Revision:**

1.34

**Date:**



**Date:**

2003/10/07 11:21:17

Definition in file [pdl\\_main.c](#).

## 8.42.2 Function Documentation

### 8.42.2.1 `record_t * _concat_strings (const record_t * s1, const record_t * s2, const char * extra)`

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 414 of file [pdl\\_main.c](#).Referenced by [concat\\_strings\(\)](#), and [concat\\_strings\\_with\\_space\(\)](#).

### 8.42.2.2 `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 353 of file [pdl\\_main.c](#).Referenced by [set\\_path\(\)](#).

### 8.42.2.3 `record_t* concat_strings (record_t * s1, record_t * s2)`

Concatenate two strings. The original two strings are freed. When the concatenation fails, the original 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 393 of file [pdl\\_main.c](#).

**8.42.2.4 record\_t\* concat\_strings\_with\_space (record\_t \* s1, record\_t \* s2)**

Concatenate two strings. The original two strings are freed. When the concatenation fails, the original 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 448 of file `pdl_main.c`.

**8.42.2.5 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 289 of file `pdl_main.c`.

Referenced by `plugin_exists()`.

**8.42.2.6 void free\_path (void)**

Free the string allocated to hold the path

Definition at line 372 of file `pdl_main.c`.

Referenced by `free_resources()`.

**8.42.2.7 void free\_resources (void)**

Free the resources.

Definition at line 612 of file `pdl_main.c`.

**8.42.2.8 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 134 of file `pdl_main.c`.

### 8.42.2.9 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 74 of file pdl\_main.c.

Referenced by startEvaluationManager().

### 8.42.2.10 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 498 of file pdl\_main.c.

Referenced by runEvaluationManager().

### 8.42.2.11 const char\* pdl\_path (void)

Get the path.

**Returns:**

Path.

Definition at line 306 of file pdl\_main.c.

### 8.42.2.12 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 187 of file pdl\_main.c.

#### 8.42.2.13 void reduce\_policies (void)

Reduce\_policies to its elementary form, i.e. each policy has a list of rules which need to be reduced.

Definition at line 217 of file pdl\_policy.c.

Referenced by startEvaluationManager().

#### 8.42.2.14 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 336 of file pdl\_main.c.

#### 8.42.2.15 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 648 of file pdl\_main.c.

Referenced by `_add_policy()`, `_add_rule()`, `_add_variable()`, `_concat_strings()`, `_set_path()`, `check_rule_for_recursion()`, `has_recursion()`, `pdl_init()`, `pdl_next_plugin()`, `reduce_to_var()`, and `yyerror()`.

#### 8.42.2.16 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 `warning()`;

##### Parameters:

*s* error string.

Definition at line 319 of file pdl\_main.c.

#### 8.42.2.17 int yyparse\_errors (void)

Tell if there were errors/warning during parsing.

##### Returns:

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

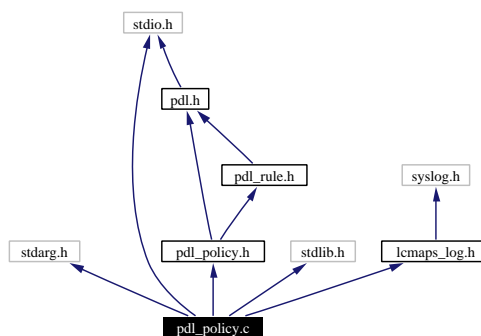
Definition at line 122 of file pdl\_main.c.

## 8.43 pdl\_policy.c File Reference

Implementation of the pdl policies.

```
#include <stdarg.h>
#include <stdio.h>
#include <stdlib.h>
#include "lcmaps_log.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)
- `BOOL check_policies_for_recursion` (void)
- 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.*

#### 8.43.1 Detailed Description

Implementation of the pdl policies.

**Author:**

G.M. Venekamp ([venekamp@nikhef.nl](mailto:venekamp@nikhef.nl))

**Version:****Revision:**

1.13

**Date:****Date:**

2003/08/06 08:15:36

Definition in file [pdl\\_policy.c](#).

## 8.43.2 Function Documentation

### 8.43.2.1 **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 whether 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 118 of file `pdl_policy.c`.

Referenced by `add_policy()`.

### 8.43.2.2 **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 84 of file `pdl_policy.c`.

### 8.43.2.3 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 allowed, FALSE otherwise.

Definition at line 66 of file pdl\_policy.c.

### 8.43.2.4 BOOL check\_policies\_for\_recursion (void)

Check for recursions in the policy rules.

**Returns:**

TRUE if at least one recursion has been found, FALSE otherwise.

Definition at line 190 of file pdl\_policy.c.

Referenced by startEvaluationManager().

### 8.43.2.5 policy\_t \* current\_policy (void)

Return the current policy.

**Returns:**

Current policy.

Definition at line 50 of file pdl\_policy.c.

### 8.43.2.6 policy\_t \* find\_policy (const char \* name)

Find a policy based.

**Parameters:**

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

Definition at line 171 of file pdl\_policy.c.

### 8.43.2.7 void free\_policies (void)

Free all policies and their allocated resources.

Definition at line 271 of file pdl\_policy.c.

Referenced by free\_resources().

**8.43.2.8 policy\_t \* get\_policies (void)**

Get the list of policies.

**Returns:**

First policy in the list.

Definition at line 244 of file pdl\_policy.c.

Referenced by check\_policies\_for\_recursion(), get\_plugins(), pdl\_next\_plugin(), and reduce\_policies().

**8.43.2.9 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 293 of file pdl\_policy.c.

Referenced by get\_plugins().

**8.43.2.10 void reduce\_policies (void)**

Reduce\_policies to its elementary form, i.e. each policy has a list of rules which need to be reduced.

Definition at line 217 of file pdl\_policy.c.

**8.43.2.11 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 157 of file pdl\_policy.c.

**8.43.2.12 void show\_policies (void)**

Display the policies and the rules associated with the policy.

Definition at line 254 of file pdl\_policy.c.



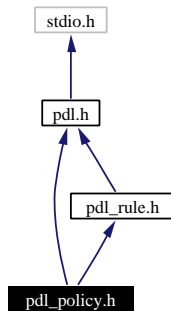
## 8.44 pdl\_policy.h File Reference

Include file for using the pdl policies.

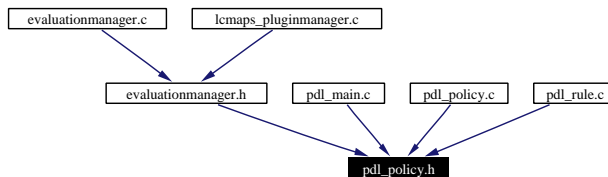
```
#include "pdl.h"
```

```
#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)

- BOOL [check\\_policies\\_for\\_recursion](#) (void)
- void [reduce\\_policies](#) (void)
- BOOL [policies\\_have\\_been\\_reduced](#) (void)
- [policy\\_t\\*](#) [find\\_policy](#) (const char \*name)
- [policy\\_t\\*](#) [current\\_policy](#) (void)
- [policy\\_t\\*](#) [get\\_policies](#) (void)

### 8.44.1 Detailed Description

Include file for using the pdl policies.

**Author:**

G.M. Venekamp ([venekamp@nikhef.nl](mailto:venekamp@nikhef.nl))

**Version:**

**Revision:**

1.8

**Date:**

**Date:**

2003/07/30 14:37:08

Definition in file [pdl\\_policy.h](#).

### 8.44.2 Typedef Documentation

#### 8.44.2.1 typedef struct [policy\\_s](#) [policy\\_t](#)

Keeping track of found policies.

### 8.44.3 Function Documentation

#### 8.44.3.1 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 84 of file `pdl_policy.c`.

### 8.44.3.2 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 allowed, FALSE otherwise.

Definition at line 66 of file pdl\_policy.c.

Referenced by `_add_policy()`.

### 8.44.3.3 BOOL check\_policies\_for\_recursion (void)

Check for recursions in the policy rules.

**Returns:**

TRUE if at least one recursion has been found, FALSE otherwise.

Definition at line 190 of file pdl\_policy.c.

### 8.44.3.4 policy\_t\* current\_policy (void)

Return the current policy.

**Returns:**

Current policy.

Definition at line 50 of file pdl\_policy.c.

### 8.44.3.5 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 171 of file pdl\_policy.c.

Referenced by `_add_policy()`, and `_add_rule()`.

### 8.44.3.6 void free\_policies (void)

Free all policies and their allocated resources.

Definition at line 271 of file pdl\_policy.c.

**8.44.3.7 policy\_t\* get\_policies (void)**

Get the list of policies.

**Returns:**

First policy in the list.

Definition at line 244 of file pdl\_policy.c.

**8.44.3.8 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 293 of file pdl\_policy.c.

**8.44.3.9 void reduce\_policies (void)**

Reduce\_policies to its elementary form, i.e. each policy has a list of rules which need to be reduced.

Definition at line 217 of file pdl\_policy.c.

**8.44.3.10 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 157 of file pdl\_policy.c.

**8.44.3.11 void show\_policies (void)**

Display the policies and the rules associated with the policy.

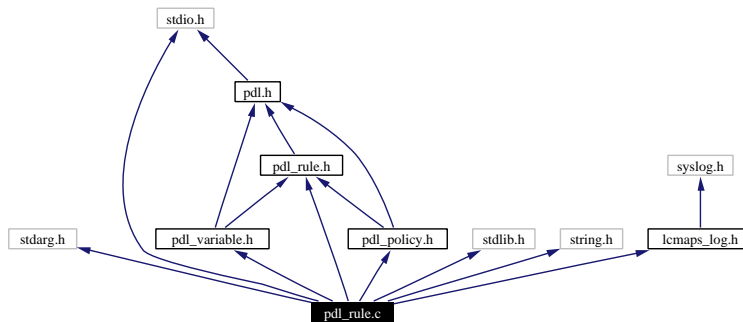
Definition at line 254 of file pdl\_policy.c.

## 8.45 pdl\_rule.c File Reference

Implementation of the pdl rules.

```
#include <stdarg.h>
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include "lcmaps_log.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)
- int `find_insert_position` (const int \*list, const int rule\_number, unsigned int high)
- unsigned int `rule_number` (const `rule_t` \*rule)
- BOOL `make_list` (int \*new\_list, const int \*list, const int rule\_number, const unsigned int depth)
- unsigned int `count_rules` (const `rule_t` \*rule)
- void `update_list` (unsigned int \*rules, unsigned int rule)
- const `rule_t*` `get_rule_number` (unsigned int rule\_num)
- 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)
- BOOL `check_rule_for_recursion` (const `rule_t` \*rule)
- `recursion_t` `has_recursion` (const `rule_t` \*rule, int \*list, unsigned int depth, unsigned int \*seen\_rules)
- void `reduce_rule` (`rule_t` \*rule)
- void `show_rules` (const `rule_t` \*rule)
- void `free_rules` (`rule_t` \*rule)
- const `rule_t*` `get_top_rule` (void)
- void `set_top_rule` (const `rule_t` \*rule)

### 8.45.1 Detailed Description

Implementation of the pdl rules.

**Author:**

G.M. Venekamp ([venekamp@nikhef.nl](mailto:venekamp@nikhef.nl))

**Version:**

**Revision:**

1.19

**Date:**

**Date:**

2003/09/04 13:47:35

Definition in file [pdl\\_rule.c](#).

### 8.45.2 Function Documentation

#### 8.45.2.1 `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 third form has only the false transit state. When either the true or false transit state for a rule does not exist, 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 144 of file `pdl_rule.c`.

Referenced by `add_rule()`.

**8.45.2.2** `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 86 of file pdl\_rule.c.

**8.45.2.3** `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 71 of file pdl\_rule.c.

**8.45.2.4** `BOOL check_rule_for_recursion (const rule_t * rule)`

Check the rule for occurrences of recursion.

**Returns:**

TRUE if a recursion have been found, FALSE otherwise.

Definition at line 220 of file pdl\_rule.c.

**8.45.2.5** `unsigned int count_rules (const rule_t * rule)`

Count the number of rules that follow 'rule' inclusive.

**Parameters:**

*rule* The rule to start count from.

**Returns:**

Number of counted rules.

Definition at line 257 of file pdl\_rule.c.

Referenced by `check_rule_for_recursion()`.

**8.45.2.6** `int find_insert_position (const int * list, const int rule_number, unsigned int high)`

Based on a sorted list, find the position where to insert a new element without disturbing the ordering in the list. The search is a binary search.

**Parameters:**

*list* List of sorted numbers.

*rule\_number* Number to be inserted.

*high* Element number of last element in the list.

**Returns:**

Position of insertion.

Definition at line 488 of file pdl\_rule.c.

Referenced by make\_list(), and update\_list().

**8.45.2.7 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 202 of file pdl\_rule.c.

**8.45.2.8 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 637 of file pdl\_rule.c.

**8.45.2.9 const rule\_t \* get\_rule\_number (unsigned int rule\_num)**

Give the position of the rule in the policy, return that rule.

**Parameters:**

*rule\_num* Position of the rule in the current policy.

**Returns:**

Rule that is associated with the rule\_num, NULL if the rule cannot be found.

Definition at line 278 of file pdl\_rule.c.

Referenced by check\_rule\_for\_recursion().

**8.45.2.10 const rule\_t \* get\_top\_rule (void)**

Get the top rule.

**Returns:**

Top rule.

Definition at line 658 of file pdl\_rule.c.



**8.45.2.11** `recursion_t` `has_recursion` (`const rule_t * rule`, `int * list`, `unsigned int depth`, `unsigned int * seen_rules`)

Check the a rule for recursion. This is done in a recursive manner. From the top rule, all possible paths are considered. Each path becomes a top of its own and from their all possible paths are traveled. Each time the tree is searched at a greater depth, a list is kept to tell which states have been seen for the current path. In this list of states no duplicates should be present. If a seen state state already appears in the list, the path taken is recursive. This information is propagated back up the traveled tree.

At the same time another list is maintained. In this list all visited states are remembered. Duplicates are not added. When all possible paths have been traveled, the list tells all visited rules. When a particular rule is not part of the tree, it is also not listed in the list. This way one can check for disconnected rules.

**Parameters:**

- rule* Rule to check for recursion.
- list* List to keep track of each traveled path.
- depth* Current depth of the tree.
- seen\_rules* Rules that have been visited and hence are part of the path.

**Returns:**

Whether or not recursion has been detected and also if it has been reported.

Definition at line 318 of file pdl\_rule.c.

Referenced by `check_rule_for_recursion()`.

**8.45.2.12** `BOOL` `make_list` (`int * new_list`, `const int * list`, `const int rule_number`, `const unsigned int depth`)

Make a new sorted list based on the current list and the element to be inserted. The element will only be added to the list if it is not already present.

**Parameters:**

- new\_list* New list after sorted insertion of new element.
- list* Old list.
- rule\_number* Number to be inserted into the list.
- depth* Current depth of the tree. It is used to detmine the number of elements of the list.

**Returns:**

TRUE if element has been added, FALSE otherwise

Definition at line 526 of file pdl\_rule.c.

Referenced by `has_recursion()`.

**8.45.2.13** `void` `reduce_rule` (`rule_t * rule`)

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

**Parameters:**

- rule* Rule to reduce.

Definition at line 558 of file pdl\_rule.c.

**8.45.2.14 unsigned int rule\_number (const rule\_t \* rule)**

Given a rule, find the corresponding position in the policy.

**Parameters:**

*rule* Rule of which the position must be found.

**Returns:**

Position of the rule in the current policy.

Definition at line 425 of file pdl\_rule.c.

Referenced by has\_recursion().

**8.45.2.15 void set\_top\_rule (const rule\_t \* rule)**

Set the top rule to a new value.

**Parameters:**

*rule* New value of top rule.

Definition at line 670 of file pdl\_rule.c.

**8.45.2.16 void show\_rules (const rule\_t \* rule)**

Show a rule and its descendants.

**Parameters:**

*rule* Rule to display.

Definition at line 615 of file pdl\_rule.c.

**8.45.2.17 void start\_new\_rules (void)**

Start a new list of rules.

Definition at line 58 of file pdl\_rule.c.

Referenced by add\_policy().

**8.45.2.18 void update\_list (unsigned int \* rules, unsigned int rule)**

Update the list that hold the visited rules. This is a sorted list for easy insertion and look-up. Duplicate rules are not inserted. The first element of the list tells the total number of elements that follow.

**Note:**

The list expects rules to be numbered starting from 1. This is because 0 denotes empty cells. The [find\\_insert\\_position\(\)](#) returns numbers starting from 0. This is corrected for in this function.

**Parameters:**

*rules* List of visited rules.

*rule* rule to insert.

Definition at line 453 of file pdl\_rule.c.

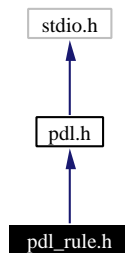
Referenced by has\_recursion().

## 8.46 pdl\_rule.h File Reference

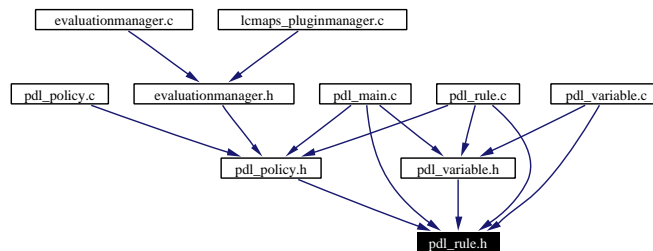
Include file for using the pdl rules.

```
#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 branches.*

### Typedefs

- typedef struct [rule\\_s](#) [rule\\_t](#)

*Structure keeps track of the state and the true/false branches.*

### Enumerations

- enum [rule\\_type\\_t](#) { [STATE](#), [TRUE\\_BRANCH](#), [FALSE\\_BRANCH](#) }

*Which type is the current rule.*

- enum [recursion\\_t](#) { [NO\\_RECURSION](#) = 0x00, [RECURSION](#) = 0x01, [RECURSION\\_HANDLED](#) = 0x02 }

*Tell something about recursion in rules.*

- enum `side_t` { `left_side`, `right_side` }

*Given a rule, which side of the rule are we working on.*

## 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)
- const `rule_t*` `get_top_rule` (void)
- void `allow_new_rules` (BOOL allow)
- void `set_top_rule` (const `rule_t` \*rule)
- BOOL `check_rule_for_recursion` (const `rule_t` \*rule)
- void `reduce_rule` (`rule_t` \*rule)

### 8.46.1 Detailed Description

Include file for using the pdl rules.

**Author:**

G.M. Venekamp ([venekamp@nikhef.nl](mailto:venekamp@nikhef.nl))

**Version:**

**Revision:**

1.11

**Date:**

**Date:**

2003/07/31 10:33:00

Definition in file [pdl\\_rule.h](#).

### 8.46.2 Typedef Documentation

#### 8.46.2.1 typedef struct `rule_s` `rule_t`

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

## 8.46.3 Enumeration Type Documentation

### 8.46.3.1 enum recursion\_t

Tell something about recursion in rules.

**Enumeration values:**

*NO\_RECURSION* There is no known recursion.

*RECURSION* Recursion has been found.

*RECURSION\_HANDLED* Recursion has been found and handled/reported.

Definition at line 62 of file pdl\_rule.h.

### 8.46.3.2 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 52 of file pdl\_rule.h.

### 8.46.3.3 enum side\_t

Given a rule, which side of the rule are we working on.

**Enumeration values:**

*left\_side* left side, i.e. state part of the rule.

*right\_side* right side, i.e. either true or false branch.

Definition at line 72 of file pdl\_rule.h.

## 8.46.4 Function Documentation

### 8.46.4.1 `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 86 of file pdl\_rule.c.

**8.46.4.2 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 71 of file pdl\_rule.c.

Referenced by allow\_rules().

**8.46.4.3 BOOL check\_rule\_for\_recursion (const [rule\\_t](#) \* *rule*)**

Check the rule for occurrences of recursion.

**Returns:**

TRUE if a recursion have been found, FALSE otherwise.

Definition at line 220 of file pdl\_rule.c.

Referenced by check\_policies\_for\_recursion().

**8.46.4.4 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 637 of file pdl\_rule.c.

Referenced by add\_policy(), and free\_policies().

**8.46.4.5 const [rule\\_t](#)\* get\_top\_rule (void)**

Get the top rule.

**Returns:**

Top rule.

Definition at line 658 of file pdl\_rule.c.

**8.46.4.6 void reduce\_rule ([rule\\_t](#) \* *rule*)**

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

**Parameters:**

*rule* Rule to reduce.

Definition at line 558 of file pdl\_rule.c.

Referenced by reduce\_policies().

**8.46.4.7 void set\_top\_rule (const rule\_t \* rule)**

Set the top rule to a new value.

**Parameters:**

*rule* New value of top rule.

Definition at line 670 of file pdl\_rule.c.

Referenced by reduce\_policies().

**8.46.4.8 void show\_rules (const rule\_t \* rule)**

Show a rule and its descendants.

**Parameters:**

*rule* Rule to display.

Definition at line 615 of file pdl\_rule.c.

Referenced by show\_policies().

**8.46.4.9 void start\_new\_rules (void)**

Start a new list of rules.

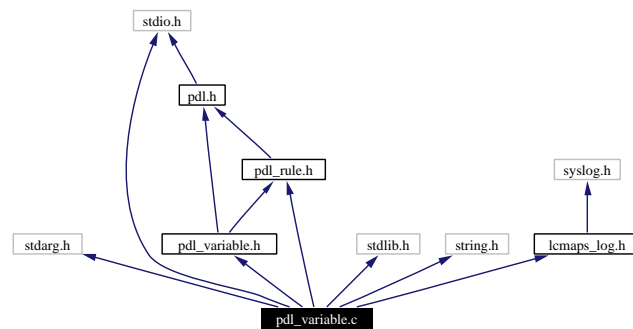
Definition at line 58 of file pdl\_rule.c.

## 8.47 pdl\_variable.c File Reference

Implementation of the pdl variables.

```
#include <stdarg.h>
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include "lcmaps_log.h"
#include "pdl_variable.h"
#include "pdl_rule.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)
- **void** `reduce_to_var` (const char \*\*name, `rule_type_t` rule\_type)
- `var_t*` `get_variables` (void)
- **void** `show_variables` (void)

### 8.47.1 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](mailto:venekamp@nikhef.nl))

#### Version:



**Revision:**

1.9

**Date:****Date:**

2003/07/30 14:37:09

Definition in file [pdl\\_variable.c](#).

## 8.47.2 Function Documentation

### 8.47.2.1 **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 already 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 89 of file `pdl_variable.c`.Referenced by `add_variable()`.

### 8.47.2.2 **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 structures is freed as well.

**Parameters:**

*name* Name of the variable.

*value* Value of the variable.

Definition at line 64 of file `pdl_variable.c`.

### 8.47.2.3 **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 193 of file pdl\_variable.c.

Referenced by \_add\_variable().

**8.47.2.4 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 168 of file pdl\_variable.c.

**8.47.2.5 void free\_variables (void)**

Free the resources allocated for the variables.

Definition at line 142 of file pdl\_variable.c.

Referenced by free\_resources().

**8.47.2.6 var.t \* get\_variables (void)**

Get a list of all variables in the configure file.

**Returns:**

First variable of the list.

Definition at line 269 of file pdl\_variable.c.

**8.47.2.7 void reduce\_to\_var (const char \*\* name, rule\_type\_t rule\_type)**

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 referring variable.

**Parameters:**

*name* Name of the variable to be reduced.

**Returns:**

Real value of the reduced variable.

Definition at line 239 of file pdl\_variable.c.

**8.47.2.8 void show\_variables (void)**

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

Definition at line 280 of file pdl\_variable.c.

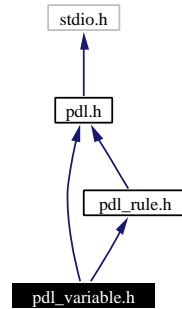
## 8.48 pdl\_variable.h File Reference

Include file for using the pdl variables.

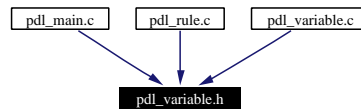
```
#include "pdl.h"
```

```
#include "pdl_rule.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)
- void [reduce\\_to\\_var](#) (const char \*\*name, [rule\\_type\\_t](#) rule\_type)
- void [show\\_variables](#) (void)
- void [free\\_variables](#) (void)
- [var\\_t](#)\* [get\\_variables](#) (void)

## 8.48.1 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](mailto:venekamp@nikhef.nl))

**Version:**

**Revision:**

1.6

**Date:**

**Date:**

2003/07/30 14:37:09

Definition in file [pdl\\_variable.h](#).

## 8.48.2 Typedef Documentation

### 8.48.2.1 typedef struct [var\\_s](#) [var\\_t](#)

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

## 8.48.3 Function Documentation

### 8.48.3.1 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 structures is freed as well.

**Parameters:**

*name* Name of the variable.

*value* Value of the variable.

Definition at line 64 of file [pdl\\_variable.c](#).

### 8.48.3.2 void [free\\_variables](#) (void)

Free the resources allocated for the variables.

Definition at line 142 of file [pdl\\_variable.c](#).

**8.48.3.3** `var.t*` `get_variables` (void)

Get a list of all variables in the configure file.

**Returns:**

First variable of the list.

Definition at line 269 of file `pdl_variable.c`.

**8.48.3.4** void `reduce_to_var` (const char \*\* *name*, `rule_type.t` *rule\_type*)

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 referring variable.

**Parameters:**

*name* Name of the variable to be reduced.

**Returns:**

Real value of the reduced variable.

Definition at line 239 of file `pdl_variable.c`.

Referenced by `reduce_rule()`.

**8.48.3.5** void `show_variables` (void)

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

Definition at line 280 of file `pdl_variable.c`.

---

## Chapter 9

# edg-lcmaps Page Documentation

### 9.1 Job Repository plugin

### 9.2 SYNOPSIS

```
lcmaps_jobrep.mod -vommdir <path> -certdir <path> [server=<MySQL hostname/dns-name>]  
[dsn=<DSN-name>] [port=<port nr.>] [driver=<path/file>] [database=<database-name>]  
[username=<database username>] [password=<database password>] [-jr_config <path/file>]
```

### 9.3 DESCRIPTION

The Job Repository module collects the user, job and user-mapping information when a job has been assigned to a fabric and handled by LCMAPS. This information is stored into a SQL database. It is essential that this information is kept secure due to the nature of the data it stores.

### 9.4 OPTIONS

#### 9.4.1 dsn=<DSN-name>

This will select the Data Source Name (DSN) that has been set in a odbc.ini file. In the odbc.ini file are one or more Database sources described with the needed settings to connect to a database. (Note: also partial connection settings are supported (3rd party support))

#### 9.4.2 server=<dns-name>

specifies the host that has the database to connect to.

#### 9.4.3 port=<port nr.>

specifies the port number on the host to override the default portnumber

---

#### 9.4.4 **driver=<path/file>**

specifies the path and filename of the shared object distributed by the MyODBC package. ODBC needs a driver file to know what kind of database it will be connecting to.

#### 9.4.5 **database=<database-name>**

specifies the name of the database that has to be connected to. One machine can host several databases. The default name to fill in is: "JobRepository".

#### 9.4.6 **username=<database username>**

specifies the database username that the LCMAPS module must use to authorize itself with.

#### 9.4.7 **password=<database password>**

specifies the database password that the LCMAPS module must use to authorize itself with. (Note: Disabled to use this in the lcmaps.db file because it is publicly accessible)

#### 9.4.8 **-jr\_config <path/file>**

The path and file specification leads to the root-only-readable file that should contain the database password (or more things like the username) to prevent that a simple user could compromise the information that is in the database. There is a specific format for this file. All these database parameters must be semi-colon delimited. The same parameters as states above are used. This means that in the lcmaps.db all these parameters could be in this file. Also that is not really advised due to practical reasons (shifting of servers for example) then you'll just have to edit one file. The lcmaps.db file in this case. This means that you can mix these parameters as you like between the files. All these parameters are concatenated after each other. When parameters are more than once defined the last one in the concatenated string will be effectively used. (default: /opt/edg/etc/lcmaps/jobrep\_config)

#### 9.4.9 **-vommdir <path>**

This is the same directory as the LCMAPS VOMS Extract module looks at. This where the VO certificates are stored. (default: /etc/grid-security/vommdir/). Note: this parameter can not be used in the jobrep\_config file.

#### 9.4.10 **-certdir <path>**

This is the same directory as the LCMAPS VOMS Extract module looks at. This where the user certificates end up (temporarily) (default: /etc/grid-security/certificates/). Note: this parameter can not be used in the jobrep\_config file.



## 9.5 afs plugin

## 9.6 SYNOPSIS

```
lcmaps_afs.mod [ -lifetime <hh[:mm[:ss]]> | -server <server> | -port <port> | -cell <cell> | -setpag | -gsiklog ]
```

## 9.7 DESCRIPTION

This plugin is an Acquisition and Enforcement Plugin and provides the LCMAPS with afs access. Users must have an accessible afs file system somewhere. The plugin can only function when gssklog and gssklogd are installed. Underneath it uses the client program gssklog to get an afs token using the user's proxy. The gssklog client depends on the daemon gssklogd to get the afs token. The daemon must be installed on one of the afs servers and hence there is no necessity for it to be installed on the same machine as the gatekeeper or the client side of gssklog. The afs machine must be reachable from the client machine. The gssklogd daemon uses the file /etc/grid-security/afsgrid-mapfile for the DN-to-AFSaccount mapping.

## 9.8 OPTIONS

### 9.8.1 -lifetime <ticket lifetime in hh[:mm[:ss]]>

Use this option to specify the maximum lifetime of afs token. gssklog allows a maximum of 720 hours. Defaults to 12 hours.

### 9.8.2 -server <afs server>

Name of the server where gssklogd runs.

### 9.8.3 -port <port>

Port number at which the gssklogd daemon runs. Defaults to 750.

### 9.8.4 -cell <cell>

Cell name.

### 9.8.5 -setpag

Setpag.

### 9.8.6 -gsiklog

Use GSI interface instead of GSS interface

## 9.9 RETURN VALUES

- LCMAPS\_MOD\_SUCCESS : Success
- LCMAPS\_MOD\_FAIL : Failure

## 9.10 ERRORS

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

## 9.11 SEE ALSO

[lcmaps\\_localaccount.mod](#), [lcmaps\\_poolaccount.mod](#), [lcmaps\\_posix\\_enf.mod](#), [lcmaps\\_ldap\\_enf.mod](#),  
[lcmaps\\_voms.mod](#),

## 9.12 example plugin

### 9.13 beschrijving

beschrijf beschrijf ...

## 9.14 ldap enforcement plugin

### 9.15 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> -timeout <timeout value>
```

### 9.16 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 compliant. (see documentation)

### 9.17 OPTIONS

#### 9.17.1 -maxuid <maxuid>

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

#### 9.17.2 -maxpgid <maxpgid>

Maximum number of primary gids to be used.

#### 9.17.3 -maxsgid <maxsgid>

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

#### 9.17.4 -hostname <hostname>

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

#### 9.17.5 -port <port>

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

#### 9.17.6 -require\_all\_groups [yes|no]

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

#### 9.17.7 -dn\_manager <DN>

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

**9.17.8 -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.

**9.17.9 -sb\_groups <searchbase>**

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

**9.17.10 -sb\_user <searchbase>**

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

**9.17.11 -timeout <timeout value>**

timeout (in seconds) that will be applied to the ldap binding

**9.18 RETURN VALUE**

- LCMAPS\_MOD\_SUCCESS : succes
- LCMAPS\_MOD\_FAIL : failure

**9.19 ERRORS**

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

**9.20 SEE ALSO**

[lcmaps\\_localaccount.mod](#), [lcmaps\\_poolaccount.mod](#), [lcmaps\\_posix\\_enf.mod](#), [lcmaps\\_voms.mod](#), [lcmaps\\_voms\\_poolaccount.mod](#), [lcmaps\\_voms\\_poolgroup.mod](#), [lcmaps\\_voms\\_localgroup.mod](#)

## 9.21 localaccount plugin

## 9.22 SYNOPSIS

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

## 9.23 DESCRIPTION

This plugin is an Acquisition Plugin and will provide the LCMAPS system with Local Account credential information. To do this it needs to look up the Distinguished Name (DN) from a user's certificate in the gridmapfile. If this DN is found in the gridmapfile 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 secondary GIDs. When this 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 its 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 an Enforcement plugin that can apply these gathered credentials in a way that is appropriate to a system administration's needs.

## 9.24 OPTIONS

### 9.24.1 -GRIDMAPFILE <gridmapfile>

See [-gridmap](#)

### 9.24.2 -gridmapfile <gridmapfile>

See [-gridmap](#)

### 9.24.3 -GRIDMAP <gridmapfile>

See [-gridmap](#)

### 9.24.4 -gridmap <gridmapfile>

When this option is set it will override the default path of the gridmapfile. It is advised to use an absolute path to the gridmapfile to avoid usage of the wrong file(path).

## 9.25 RETURN VALUES

- LCMAPS\_MOD\_SUCCESS : Success
- LCMAPS\_MOD\_FAIL : Failure

## 9.26 ERRORS

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

## 9.27 SEE ALSO

[lcmaps\\_poolaccount.mod](#), [lcmaps\\_posix\\_enf.mod](#), [lcmaps\\_ldap\\_enf.mod](#), [lcmaps\\_voms.mod](#), [lcmaps\\_voms\\_poolaccount.mod](#), [lcmaps\\_voms\\_poolgroup.mod](#), [lcmaps\\_voms\\_localgroup.mod](#)

## 9.28 poolaccount plugin

### 9.29 SYNOPSIS

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

### 9.30 DESCRIPTION

This plugin is a Acquisition Plugin and will provide the LCMAPS system with Pool Account information. To do this it needs to look up the Distinguished Name (DN) from a user's certificate in the gridmapfile. If this DN is found in the gridmapfile the plugin now knows to which pool of local system accounts the user will be mapped. The poolname (starting with a dot '.') instead of an alphanumeric character) will be converted into the an account from a list of local accounts. This list is located in the *gridmapdir* and is made out of filenames. These filenames correspond to the system poolaccount names. (E.g. if a DN corresponds to *.test* in the gridmapfile, it will be mapped to *test001*, *test002*, etc., which names can be found in the *gridmapdir*)

If there is no pool account assigned to the user yet, the plugin will get a directory listing of the *gridmapdir*. This list will contain usernames corresponding to system accounts specially designated for pool accounting. If the plugin resolved the mapping of a certain pool name, let's say *.test*, the plugin will look into the directory list and will find the first available file in the list corresponding with *'test'* (e.g. *'test001'*) by checking the number of links to its *i*-node. If this number is 1, this account is still available. To lease this account a second hard link is created, named after the URL-encoded, decapitalized DN.

When a user returns to this site the plugin will look for the DN of the user (URL encoded) in this directory. If found, the corresponding poolaccount will be assigned to the user.

The plugin will resolve the UID, GID and all the secondary GIDs belonging to the poolaccount. When this 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 its 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 an Enforcement plugin that can apply these gathered credentials in a way that is appropriate to a system administration's needs.

### 9.31 OPTIONS

#### 9.31.1 -GRIDMAPFILE <gridmapfile>

See [-gridmap](#)

#### 9.31.2 -gridmapfile <gridmapfile>

See [-gridmap](#)

#### 9.31.3 -GRIDMAP <gridmapfile>

See [-gridmap](#)



### 9.31.4 **-gridmap** <gridmapfile>

If this option is set, it will override the default path of the gridmapfile. It is advised to use an absolute path to the gridmapfile to avoid usage of the wrong file(path).

### 9.31.5 **-GRIDMAPDIR** <gridmapdir>

See [-gridmapdir](#)

### 9.31.6 **-gridmapdir** <gridmapdir>

If this option is set, it will override the default path to the gridmapdir. It is advised to use an absolute path to the gridmapdir to avoid usage of the wrong path.

### 9.31.7 **-OVERRIDE\_INCONSISTENCY**

See [-override\\_inconsistency](#)

### 9.31.8 **-override\_inconsistency**

Moving a user from one pool to another (because of a VO change) should only be done by changing the gridmapfile indicating the new pool for this user. If a user has already been mapped previously to a poolaccount, there is a link present between this poolaccount and his DN. In the good old days prior to LCMAPS, a 'pool change' would still result in a mapping to the old pool account, neglecting the administrative changes in the gridmapfile. LCMAPS corrects this behaviour: By default the poolaccount plugin will *fail* if the pool designated by the gridmapfile doesn't match the previously mapped poolaccount leasename. If the site doesn't want a failure on this inconsistency it can turn on this parameter. When the inconsistency is detected the plugin will automatically unlink the previous mapping and will proceed by making a *new* lease from the new pool.

## 9.32 RETURN VALUES

- LCMAPS\_MOD\_SUCCESS : Success
- LCMAPS\_MOD\_FAIL : Failure

## 9.33 ERRORS

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

## 9.34 SEE ALSO

[lcmaps\\_localaccount.mod](#), [lcmaps\\_posix\\_enf.mod](#), [lcmaps\\_ldap\\_enf.mod](#), [lcmaps\\_voms.mod](#), [lcmaps\\_voms\\_poolaccount.mod](#), [lcmaps\\_voms\\_poolgroup.mod](#), [lcmaps\\_voms\\_localgroup.mod](#)

## 9.35 posix enforcement plugin

## 9.36 SYNOPSIS

**lcmaps\_posix\_enf.mod** [-maxuid|-MAXUID <number of uids>] [-maxpgid|-MAXPGID <number of primary gids>] [-maxsgid|-MAXSGID <number of secondary gids>]

## 9.37 DESCRIPTION

The Posix Enforcement plugin will enforce (apply) the gathered credentials that are stacked in the data-structure of the Plugin Manager. The plugin will get the credential information that is gathered by one or more Acquisition plugins. This implies that at least one Acquisition should have been run 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 its user groups.

The Posix Enforcent plugin does not validate the secondary 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.

The (BSD/POSIX) functions setreuid(), setregid() and setgroups() are used to change the privileges of the process from root to that of a local user.

## 9.38 OPTIONS

### 9.38.1 -MAXUID <number of uids>

See [-maxuid](#)

### 9.38.2 -maxuid <number of uids>

In principle, this will set the maximum number of allowed UIDs that this plugin will handle, but at the moment only the first UID found will be enforced; the others will discarded. 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.

### 9.38.3 -MAXPGID <number of primary gids>

See [-maxpgid](#)

### 9.38.4 -maxpgid <number of primary gids>

This will set the maximum number of allowed Primary GIDs that this plugin will handle, similar to [-maxuid](#). Also here only the first primary GID found will be taken into account.

### 9.38.5 -MAXSGID <number of secondary gids>

See [-maxsgid](#)

### 9.38.6 -maxsgid <number of secondary gids>

This will set the maximum allowed Secondary GIDs that this plugin will handle. This number is limited by the system (NGROUPS) and is usually 32. If the plugin cannot determine the system value, it limits itself to 32.

### 9.38.7 -set\_only\_euid [yes|no]

The result of setting this option to 'yes' is that only the effective uid is set. In other words, it is still possible to regain root (uid) privileges for the process. This is definitely undesirable if this module is used from a process like the gatekeeper, since it would be possible for user jobs to get root privileges ! **THIS IS A DANGEROUS OPTION, PLEASE KNOW WHAT YOU'RE DOING !** By default, this option is set to 'no'. Possibly this option should be set if the module is used by gridFTP, since this service does not spawn user jobs and has to regain root privileges at the end.

### 9.38.8 -set\_only\_egid [yes|no]

Analogue to the previous option the result of setting this option to 'yes' is that only the effective (primary) gid is set. In other words, it is still possible to regain root (gid) privileges for the process. This is definitely undesirable if this module is used from a process like the gatekeeper, since it would be possible for user jobs to get root privileges ! **THIS IS A DANGEROUS OPTION, PLEASE KNOW WHAT YOU'RE DOING !** By default, this option is set to 'no'. Possibly this option should be set if the module is used by gridFTP, since this service does not spawn user jobs and has to regain root privileges at the end.

## 9.39 RETURN VALUES

- LCMAPS\_MOD\_SUCCESS : Success
- LCMAPS\_MOD\_FAIL : Failure

## 9.40 ERRORS

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

## 9.41 SEE ALSO

[lcmaps\\_localaccount.mod](#), [lcmaps\\_poolaccount.mod](#), [lcmaps\\_ldap\\_enf.mod](#), [lcmaps\\_voms.mod](#), [lcmaps\\_voms\\_poolaccount.mod](#), [lcmaps\\_voms\\_poolgroup.mod](#), [lcmaps\\_voms\\_localgroup.mod](#)

## 9.42 voms plugin

## 9.43 SYNOPSIS

**lcmaps\_voms.mod** -vommdir <vommdir> -certdir <certdir>

## 9.44 DESCRIPTION

This plugin forms the link between the VOMS data found in the user grid credential (X509 certificate) and the lcmaps system. It will retrieve the VOMS data by using the VOMS API. The plugin stores the VOMS data in the LCMAPS process space, where it is accessible by other 'VOMS-aware' plugins, and should, therefore, be evaluated before the other plugins, that actually gather the local credentials based on the VOMS information (e.g. [lcmaps\\_voms\\_poolaccount.mod](#), [lcmaps\\_voms\\_poolgroup.mod](#) and [lcmaps\\_voms\\_localgroup.mod](#)).

## 9.45 OPTIONS

### 9.45.1 -VOMSDIR <vommdir>

See [-vommdir](#)

### 9.45.2 -vommdir <vommdir>

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

### 9.45.3 -CERTDIR <certdir>

See [-certdir](#)

### 9.45.4 -certdir <certdir>

This is the directory which contains the CA certificates

## 9.46 RETURN VALUES

- LCMAPS\_MOD\_SUCCESS : Success
- LCMAPS\_MOD\_FAIL : Failure

## 9.47 ERRORS

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

## 9.48 SEE ALSO

[lcmads\\_voms\\_poolaccount.mod](#), [lcmads\\_voms\\_poolgroup.mod](#), [lcmads\\_voms\\_localgroup.mod](#) [lcmads\\_-localaccount.mod](#), [lcmads\\_poolaccount.mod](#), [lcmads\\_posix\\_enf.mod](#), [lcmads\\_ldap\\_enf.mod](#),

## 9.49 voms localgroup plugin

### 9.50 SYNOPSIS

```
lcmaps_voms_localgroup.mod -GROUPMAPFILE|-groupmapfile|-GROUPMAP|-groupmap
<groupmapfile> [-mapall] [-mapmin <group count>]
```

### 9.51 DESCRIPTION

The localgroup acquisition plugin is a 'VOMS-aware' plugin. It uses the VOMS information (acquired by the plugin [lcmaps\\_voms.mod](#)) to gather primary and secondary GIDs. This is accomplished by matching VO-GROUP-ROLE(-CAPABILITY) combinations in the so-called *groupmapfile* (gridmapfile style) and by finding the corresponding local GID. Wildcards can be used in the groupmapfile to match VO-GROUP-ROLE combinations.

EXAMPLE 'groupmapfile':

```
"/VO=atlas/GROUP=mcprod" atmcpod
```

```
"/VO=atlas/GROUP=*" atlasgrps
```

A VO-GROUP combination /VO=atlas/GROUP=mcprod matches "/VO=atlas/GROUP=mcprod", resulting in a mapping to the GID of the 'atmcpod' group. All the other groups within the 'atlas' VO will be mapped to 'atlasgrps'. A user with /VO=cms/GROUP=user will not be mapped to any local system group, unless there will be an extra row in the groupmapfile like "/VO=\*" allothers' resulting in a mapping from any other VO-GROUP-ROLE combination to 'allothers'. The mapping is based on the first match found for a VO-GROUP-ROLE combination, implying that the most significant row must be on top.

The poolgroup plugin will try to match each VO-GROUP-ROLE combination that was found by the plugin [lcmaps\\_voms.mod](#). The first VO-GROUP-ROLE combination will become the primary group, the others secondary groups. As the primary GID may be used for auditing and accounting purposes it is important that the user uses the correct ordering of VO-GROUP-ROLE combinations in his grid credential (X509 certificate).

### 9.52 OPTIONS

#### 9.52.1 -GROUPMAPFILE <groupmapfile>

See [-groupmap](#)

#### 9.52.2 -groupmapfile <groupmapfile>

See [-groupmap](#)

#### 9.52.3 -GROUPMAP <groupmapfile>

See [-groupmap](#)

### 9.52.4 -groupmap <groupmapfile>

If this option is set, it will override the default path to the groupmapfile. It is advised to use an absolute path to the groupmapfile to avoid usage of the wrong file(path).

### 9.52.5 -mapall

If this parameter is set, the plugin only succeeds if it manages to map all voms data entries to (system) groups and find their GID. There is no communication between different plugins (like the voms\_poolgroup plugin) about the failures. A log entry will state the VO-GROUP-ROLE combination that made the plugin fail.

### 9.52.6 -mapmin <group count>

This option will set a minimum amount of groups that have to be resolved for later mapping. If the minimum is not set then the minimum amount is set to '0' by default. If the plugin is not able to the required number of local groups it will fail. Note: if the minimum is set to zero or the minimum is not set the plugin will return a success if no other errors occur, even if no local groups were found.

## 9.53 RETURN VALUES

- LCMAPS\_MOD\_SUCCESS : Success
- LCMAPS\_MOD\_FAIL : Failure

## 9.54 ERRORS

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

## 9.55 SEE ALSO

[lcmaps\\_voms.mod](#), [lcmaps\\_voms\\_poolaccount.mod](#), [lcmaps\\_voms\\_poolgroup.mod](#), [lcmaps\\_-localaccount.mod](#), [lcmaps\\_poolaccount.mod](#), [lcmaps\\_posix\\_enf.mod](#), [lcmaps\\_ldap\\_enf.mod](#),

## 9.56 voms poolaccount plugin

### 9.57 SYNOPSIS

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

### 9.58 DESCRIPTION

This poolaccount acquisition plugin is a 'VOMS-aware' modification of the 'poolaccount' plugin. The plugin tries to find a poolaccount (more specifically a UID) based on the VOMS information that has been retrieved by the plugin `lcmaps_voms.mod` from the user's grid credential. It will try to match a VO-GROUP-ROLE combination from the user's grid credential with an entry in a gridmapfile (most likely the traditional gridmapfile, used by the localaccount and poolaccount plugins) In this file VO-GROUP-ROLE combinations are listed with a poolaccount entry, as shown in the following example.

EXAMPLE:

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

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

If the first matching VO-GROUP-ROLE combination is `"/VO=wilma/GROUP=*` the plugin will get a poolaccount from the `.test` pool. This could result in `wilma001` 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 for the `'poolaccount'` plugin (the `gridmapdir`), otherwise it gives rise to inconsistencies when both are used on a site. The actual account assigned to the user is based on his VO information matched in the gridmapfile, the user's DN and the primary (and secondary) GIDs gathered so far. In the `gridmapdir` directory this is reflected in the lease name, which consists of the url-encoded DN + a concatenation of the gathered groupnames. So a lease name could look like this:

EXAMPLE DN with pool/localgroups attached: `%2fo%3ddutchgrid%2fo%3dusers%2fo%3dnikhelf%2fcn%3dmar`

If a user changes his VO-GROUP-ROLE combinations (but not his VO), in this case he will be mapped to a different account (UID) within the same pool.

### 9.59 NOTE 1

This plugin should only be used in combination with the `'voms_localgroup'` and/or `'voms_poolgroup'` plugins.

### 9.60 NOTE 2

The options `'-do_not_require_primary_gid'` and `'-do_not_use_secondary_gids'` can not be used together, because at least one GID is needed.



## 9.61 OPTIONS

### 9.61.1 **-GRIDMAPFILE** <gridmapfile>

See [-gridmap](#)

### 9.61.2 **-gridmapfile** <gridmapfile>

See [-gridmap](#)

### 9.61.3 **-GRIDMAP** <gridmapfile>

See [-gridmap](#)

### 9.61.4 **-gridmap** <gridmapfile>

When this option is set it will override the default path to the gridmapfile. It is advised to use an absolute path to the gridmapfile to avoid usage of the wrong file(path).

### 9.61.5 **-GRIDMAPDIR** <gridmapdir>

See [-gridmapdir](#)

### 9.61.6 **-gridmapdir** <gridmapdir>

If this option is set, it will override the default path to the gridmapdir. It is advised to use an absolute path to the gridmapdir to avoid usage of the wrong path.

### 9.61.7 **-do\_not\_use\_secondary\_gids**

The determination of the poolaccount will not be based on the secondary GIDs found, but only on the user's DN, the VOMS info for the user and the primary GID that has been found. Cannot be used with [-do\\_not\\_require\\_primary\\_gid](#).

### 9.61.8 **-do\_not\_require\_primary\_gid**

The determination of the poolaccount will not be based on the primary GID found, but only on the user's DN, the VOMS info for the user and the secondary GIDs found. Normally this option should not be used, but it can be useful for debugging. Cannot be used with [-do\\_not\\_use\\_secondary\\_gids](#).

### 9.61.9 **-OVERRIDE\_INCONSISTENCY**

See [-override\\_inconsistency](#)

### 9.61.10 `-override_inconsistency`

Moving a user from one pool to another (because of a VO change) should only be done by changing the gridmapfile indicating the new pool for this user. If a user has already been mapped previously to a poolaccount, there is a link present between this poolaccount and his DN. In the good old days prior to LCMAPS, a 'pool change' would still result in a mapping to the old pool account, neglecting the administrative changes in the gridmapfile. LCMAPS corrects this behaviour: By default the voms\_poolaccount plugin will *fail* if the pool designated by the gridmapfile doesn't match the previously mapped voms\_poolaccount leasename. If the site doesn't want a failure on this inconsistency it can turn on this parameter. When the inconsistency is detected the plugin will automatically unlink the previous mapping and will proceed by making a *new* lease from the new pool.

## 9.62 RETURN VALUES

- LCMAPS\_MOD\_SUCCESS : Success
- LCMAPS\_MOD\_FAIL : Failure

## 9.63 ERRORS

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

## 9.64 SEE ALSO

[lcmaps\\_voms.mod](#), [lcmaps\\_voms\\_localgroup.mod](#), [lcmaps\\_voms\\_poolgroup.mod](#), [lcmaps\\_-localaccount.mod](#), [lcmaps\\_poolaccount.mod](#), [lcmaps\\_posix\\_enf.mod](#), [lcmaps\\_ldap\\_enf.mod](#),

## 9.65 voms poolgroup plugin

### 9.66 SYNOPSIS

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

### 9.67 DESCRIPTION

The poolgroup acquisition plugin is a 'VOMS-aware' plugin. It uses the VOMS information (acquired by the plugin [lcmaps\\_voms.mod](#)) to gather primary and secondary GIDs. This is accomplished by matching VO-GROUP-ROLE(-CAPABILITY) combinations in the so-called *groupmapfile* (gridmapfile style) and by finding the corresponding 'poolgroup' (similar to the 'poolaccount' procedure, see [lcmaps.-poolaccount.mod](#)) 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
```

The VO-GROUP-ROLE combination `"/VO=atlas/GROUP=mcprod"` starts with an alphanumeric character (not `"/`) and indicates a localgroup entry in the groupmapfile (will be resolved by the [lcmaps\\_voms.-localgroup.mod](#)). The VO-GROUP-ROLE combination `"/VO=atlas/GROUP=*"` 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 *poolaccount* plugin this plugin will link an entry (in this case a VO-GROUP-ROLE combination) to a locally known group (a.k.a. poolgroup) in the *groupmapdir* directory. The difference with the *poolaccount* plugin is that there is not a Distinguished Name but a VO-GROUP-ROLE combination and there is no poolaccount but poolgroup defined in the groupmapfile (similar to the gridmapfile). Instead of the *gridmapdir* the *groupmapdir* directory is used for the registration of the mapping between poolgroups and the VO-GROUP-ROLE combination.

As you can see 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 entry `"/VO=atlas/GROUP=dev"` will also result in a group from this '.atlas' pool, but a different one, e.g. 'atlas002'. Finally, we have random other groups not predefined in the groupmapfile, for example `"/VO=atlas/GROUP=foo"`, which matches `"/VO=atlas/GROUP=*"` in the groupmapfile. This VO-GROUP combination will be mapped to a poolgroup (probably) called 'atlas003'.

The poolgroup plugin will try to match each VO-GROUP-ROLE combination that was found by the plugin [lcmaps\\_voms.mod](#). The first VO-GROUP-ROLE combination will become the primary group, the others secondary groups. As the primary GID may be used for auditing and accounting purposes it is important that the user uses the correct ordering of VO-GROUP-ROLE combinations in his grid credential (X509 certificate).

## 9.68 OPTIONS

### 9.68.1 **-GROUPMAPFILE** <groupmapfile>

See [-groupmap](#)

### 9.68.2 **-groupmapfile** <groupmapfile>

See [-groupmap](#)

### 9.68.3 **-GROUPMAP** <groupmapfile>

See [-groupmap](#)

### 9.68.4 **-groupmap** <groupmapfile>

If this option is set, it will override the default path to the groupmapfile. It is advised to use an absolute path to the groupmapfile to avoid usage of the wrong file(path).

### 9.68.5 **-GROUPMAPDIR** <groupmapdir>

See [-groupmapdir](#)

### 9.68.6 **-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.

### 9.68.7 **-mapall**

If this parameter is set, the plugin only succeeds if it manages to map all voms data entries to (system) groups and find their GID. There is no communication between different plugins (like the voms\_localgroup plugin) about the failures. A log entry will state the VO-GROUP-ROLE combination that made the plugin fail.

### 9.68.8 **-OVERRIDE\_INCONSISTENCY**

See [-override\\_inconsistency](#)

### 9.68.9 **-override\_inconsistency**

Moving a VO group from one pool to another should only be done by changing the groupmapfile indicating the new pool for this VO group. If a VO group has already been mapped previously to a poolaccount, there is a link present between this poolgroup and its VO-GROUP-ROLE combination. By default the voms.-poolgroup plugin will *fail* if the pool designated by the gridmapfile doesn't match the previously mapped

poolgroup leasename. If the site doesn't want a failure on this inconsistency it can turn on this parameter. When the inconsistency is detected the plugin will automatically unlink the previous mapping and will proceed by making a *new* lease from the new pool.

#### 9.68.10 **-mapmin** <group count>

This option will set a minimum amount of groups that have to be resolved for later mapping. If the minimum is not set then the minimum amount is set to '0' by default. If the plugin is not able to the required number of poolgroups it will fail. Note: if the minimum is set to zero or the minimum is not set the plugin will return a success if no other errors occur, even if no poolgroups were found.

## 9.69 RETURN VALUES

- LCMAPS\_MOD\_SUCCESS : Success
- LCMAPS\_MOD\_FAIL : Failure

## 9.70 ERRORS

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

## 9.71 SEE ALSO

[lcmaps\\_voms.mod](#), [lcmaps\\_voms\\_poolaccount.mod](#), [lcmaps\\_voms\\_localgroup.mod](#), [lcmaps-localaccount.mod](#), [lcmaps\\_poolaccount.mod](#), [lcmaps\\_posix\\_enf.mod](#), [lcmaps\\_ldap\\_enf.mod](#),

---

# Index

- `_add_policy`
    - `pdl_policy.c`, 142
  - `_add_rule`
    - `pdl_rule.c`, 150
  - `_add_variable`
    - `pdl_variable.c`, 161
  - `_concat_strings`
    - `pdl_main.c`, 137
  - `_lcmaps_cred_data.h`, 31
    - `cleanCredentialData`, 32
  - `_lcmaps_db_read.h`, 33
    - `lcmaps_db_clean`, 34
    - `lcmaps_db_clean_list`, 34
    - `lcmaps_db_fill_entry`, 34
    - `lcmaps_db_read`, 34
  - `_lcmaps_defines.h`, 36
    - `MAXARGS`, 36
    - `MAXARGSTRING`, 36
    - `MAXPATHLEN`, 36
  - `_lcmaps_log.h`, 38
    - `DO_SYSLOG`, 39
    - `DO_USRLOG`, 39
    - `lcmaps_log_close`, 39
    - `lcmaps_log_open`, 39
    - `MAX_LOG_BUFFER_SIZE`, 39
  - `_lcmaps_pluginmanager.h`, 41
    - `resetCredentialData`, 42
    - `runPlugin`, 42
    - `runPluginManager`, 42
    - `startPluginManager`, 43
    - `stopPluginManager`, 43
  - `_lcmaps_runvars.h`, 44
    - `lcmaps_extractRunVars`, 45
    - `lcmaps_getRunVars`, 45
    - `lcmaps_setRunVars`, 45
  - `_lcmaps_utils.h`, 47
    - `lcmaps_fill_cred`, 48
    - `lcmaps_release_cred`, 48
    - `lcmaps_tokenize`, 48
  - `_set_path`
    - `pdl_main.c`, 137
  - `add_policy`
    - `pdl_policy.c`, 142
    - `pdl_policy.h`, 146
  - `add_rule`
    - `pdl_rule.c`, 150
    - `pdl_rule.h`, 157
  - `add_variable`
    - `pdl_variable.c`, 161
    - `pdl_variable.h`, 165
  - `addCredentialData`
    - `lcmaps_cred_data.h`, 69
  - `allow_new_rules`
    - `pdl_rule.c`, 151
    - `pdl_rule.h`, 157
  - `allow_rules`
    - `pdl_policy.c`, 142
    - `pdl_policy.h`, 146
  - `argInOut`
    - `lcmaps_argument_s`, 18
  - `argName`
    - `lcmaps_argument_s`, 18
  - `args`
    - `plugin_s`, 26
  - `argType`
    - `lcmaps_argument_s`, 18
  - `capability`
    - `lcmaps_vo_data_s`, 23
  - `check_policies_for_recursion`
    - `pdl_policy.c`, 143
    - `pdl_policy.h`, 147
  - `check_rule_for_recursion`
    - `pdl_rule.c`, 151
    - `pdl_rule.h`, 158
  - `clean_plugin_list`
    - `lcmaps_pluginmanager.c`, 101
  - `cleanCredentialData`
    - `_lcmaps_cred_data.h`, 32
  - `cntPriGid`
    - `cred_data_s`, 16
  - `cntSecGid`
    - `cred_data_s`, 16
  - `cntUid`
    - `cred_data_s`, 16
  - `cntVoCred`
    - `cred_data_s`, 16
  - `cntVoCredMapping`
    - `cred_data_s`, 16
-

- cntVoCredString
  - cred\_data.s, 16
- COMMENT\_CHARS
  - lcmaps\_db\_read.c, 71
- concat\_strings
  - pdl.h, 132
  - pdl\_main.c, 137
- concat\_strings\_with\_space
  - pdl.h, 132
  - pdl\_main.c, 137
- count\_rules
  - pdl\_rule.c, 151
- cred
  - lcmaps\_cred\_id.s, 19
- cred\_data.s, 15
  - cntPriGid, 16
  - cntSecGid, 16
  - cntUid, 16
  - cntVoCred, 16
  - cntVoCredMapping, 16
  - cntVoCredString, 16
  - dn, 17
  - priGid, 17
  - secGid, 17
  - uid, 17
  - VoCred, 16
  - VoCredMapping, 16
  - VoCredString, 16
- cred\_data\_t
  - lcmaps\_cred\_data.h, 68
- cred\_to\_dn
  - lcmaps\_utils.c, 112
- credData
  - lcmaps\_cred\_data.c, 67
- current\_policy
  - pdl\_policy.c, 143
  - pdl\_policy.h, 147
- d\_path
  - pdl\_main.c, 136
- DEBUG\_LEVEL
  - lcmaps\_log.c, 91
- debug\_level
  - lcmaps\_log.c, 91
- default\_path
  - pdl\_main.c, 136
- detect\_loop
  - pdl\_variable.c, 161
- dn
  - cred\_data.s, 17
  - lcmaps\_cred\_id.s, 19
- DO\_SYSLOG
  - \_lcmaps\_log.h, 39
- DO\_USRLOG
  - \_lcmaps\_log.h, 39
- ESCAPING\_CHARS
  - lcmaps\_db\_read.c, 71
- EVALUATION\_FAILURE
  - pdl.h, 131
- EVALUATION\_START
  - pdl.h, 131
- EVALUATION\_SUCCESS
  - pdl.h, 131
- evaluationmanager.c, 50
  - free\_lcmaps\_db\_entry, 51
  - getPluginNameAndArgs, 51
  - global\_plugin\_list, 52
  - runEvaluationManager, 51
  - startEvaluationManager, 51
  - stopEvaluationManager, 52
- evaluationmanager.h, 53
  - getPluginNameAndArgs, 54
  - runEvaluationManager, 54
  - startEvaluationManager, 54
  - stopEvaluationManager, 54
- extra\_logstr
  - lcmaps\_log.c, 91
- FALSE\_BRANCH
  - pdl\_rule.h, 157
- false\_branch
  - rule.s, 29
- fexist
  - lcmaps\_utils.c, 112
- find\_first\_space
  - pdl\_main.c, 138
- find\_insert\_position
  - pdl\_rule.c, 151
- find\_policy
  - pdl\_policy.c, 143
  - pdl\_policy.h, 147
- find\_state
  - pdl\_rule.c, 152
- find\_variable
  - pdl\_variable.c, 162
- free\_lcmaps\_db\_entry
  - evaluationmanager.c, 51
- free\_path
  - pdl\_main.c, 138
- free\_policies
  - pdl\_policy.c, 143
  - pdl\_policy.h, 147
- free\_resources
  - pdl.h, 132
  - pdl\_main.c, 138
- free\_rules
  - pdl\_rule.c, 152

- pdl\_rule.h, 158
- free\_variables
  - pdl\_variable.c, 162
  - pdl\_variable.h, 165
- get\_plugins
  - pdl.h, 132
  - pdl\_main.c, 138
- get\_policies
  - pdl\_policy.c, 143
  - pdl\_policy.h, 147
- get\_procsymbol
  - lcmaps\_pluginmanager.c, 102
- get\_rule\_number
  - pdl\_rule.c, 152
- get\_top\_rule
  - pdl\_rule.c, 152
  - pdl\_rule.h, 158
- get\_variables
  - pdl\_variable.c, 162
  - pdl\_variable.h, 165
- getCredentialData
  - lcmaps\_cred\_data.h, 69
- getPluginNameAndArgs
  - evaluationmanager.c, 51
  - evaluationmanager.h, 54
- gid
  - lcmaps\_vo\_mapping\_s, 25
- global\_plugin\_list
  - evaluationmanager.c, 52
- group
  - lcmaps\_vo\_data\_s, 23
- groupname
  - lcmaps\_vo\_mapping\_s, 25
- handle
  - lcmaps\_pluginidl\_s, 21
- has\_recursion
  - pdl\_rule.c, 152
- init\_argc
  - lcmaps\_pluginidl\_s, 21
- init\_argv
  - lcmaps\_pluginidl\_s, 21
- INITPROC
  - lcmaps\_pluginmanager.c, 101
- Interface to LCMAPS (library), 11
- INTROPROC
  - lcmaps\_pluginmanager.c, 101
- lcmaps.c, 56
  - lcmaps\_cred, 57
  - lcmaps\_initialized, 57
- lcmaps.h, 58
  - lcmaps\_init, 59
  - lcmaps\_run, 59
  - lcmaps\_run\_and\_return\_username, 59
  - lcmaps\_run\_without\_credentials, 60
  - lcmaps\_term, 60
- lcmaps\_add\_username\_to\_ldapgroup
  - lcmaps\_ldap.c, 86
- lcmaps\_argument\_s, 18
  - argInOut, 18
  - argName, 18
  - argType, 18
  - value, 18
- lcmaps\_argument\_t
  - lcmaps\_arguments.h, 62
- lcmaps\_arguments.c, 61
- lcmaps\_arguments.h, 62
  - lcmaps\_argument\_t, 62
  - lcmaps\_cntArgs, 63
  - lcmaps\_findArgName, 63
  - lcmaps\_findArgNameAndType, 64
  - lcmaps\_getArgValue, 64
  - lcmaps\_setArgValue, 64
- lcmaps\_cleanVoData
  - lcmaps\_vo\_data.h, 120
- lcmaps\_cleanVoMapping
  - lcmaps\_vo\_data.h, 120
- lcmaps\_cntArgs
  - lcmaps\_arguments.h, 63
- lcmaps\_copyVoData
  - lcmaps\_vo\_data.h, 121
- lcmaps\_copyVoMapping
  - lcmaps\_vo\_data.h, 121
- lcmaps\_createVoData
  - lcmaps\_vo\_data.h, 121
- lcmaps\_createVoMapping
  - lcmaps\_vo\_data.h, 122
- lcmaps\_cred
  - lcmaps.c, 57
- lcmaps\_cred\_data.c, 66
  - credData, 67
  - printCredData, 66
- lcmaps\_cred\_data.h, 68
  - addCredentialData, 69
  - cred\_data\_t, 68
  - getCredentialData, 69
- lcmaps\_cred\_id\_s, 19
  - cred, 19
  - dn, 19
- lcmaps\_cred\_id\_t
  - lcmaps\_types.h, 109
- lcmaps\_db\_clean
  - \_lcmaps\_db\_read.h, 34
- lcmaps\_db\_clean\_list
  - \_lcmaps\_db\_read.h, 34



- lcmaps\_db\_entry\_s, 20
  - next, 20
  - pluginargs, 20
  - pluginname, 20
- lcmaps\_db\_entry\_t
  - lcmaps\_db\_read.h, 76
- lcmaps\_db\_file\_default
  - lcmaps\_pluginmanager.c, 103
- lcmaps\_db\_fill\_entry
  - \_lcmaps\_db\_read.h, 34
- lcmaps\_db\_list
  - lcmaps\_db\_read.c, 74
- lcmaps\_db\_parse\_line
  - lcmaps\_db\_read.c, 72
- lcmaps\_db\_parse\_pair
  - lcmaps\_db\_read.c, 73
- lcmaps\_db\_parse\_string
  - lcmaps\_db\_read.c, 73
- lcmaps\_db\_read
  - \_lcmaps\_db\_read.h, 34
- lcmaps\_db\_read.c, 70
  - COMMENT\_CHARS, 71
  - ESCAPING\_CHARS, 71
  - lcmaps\_db\_list, 74
  - lcmaps\_db\_parse\_line, 72
  - lcmaps\_db\_parse\_pair, 73
  - lcmaps\_db\_parse\_string, 73
  - lcmaps\_db\_read\_entries, 73
  - MAXDBENTRIES, 71
  - MAXPAIRS, 71
  - NUL, 71
  - PAIR\_SEP\_CHARS, 71
  - PAIR\_TERMINATOR\_CHARS, 72
  - QUOTING\_CHARS, 72
  - VARVAL\_SEP\_CHARS, 72
  - VARVAL\_TERMINATOR\_CHARS, 72
  - WHITESPACE\_CHARS, 72
- lcmaps\_db\_read.h, 75
  - lcmaps\_db\_entry\_t, 76
- lcmaps\_db\_read\_entries
  - lcmaps\_db\_read.c, 73
- lcmaps\_defines.h, 77
  - LCMAPS\_ETC\_HOME, 77
  - LCMAPS\_LIB\_HOME, 77
  - LCMAPS\_MAXARGS, 78
  - LCMAPS\_MAXARGSTRING, 78
  - LCMAPS\_MAXPATHLEN, 78
  - LCMAPS\_MOD\_ENTRY, 78
  - LCMAPS\_MOD\_FAIL, 78
  - LCMAPS\_MOD\_HOME, 78
  - LCMAPS\_MOD\_NOENTRY, 78
  - LCMAPS\_MOD\_NOFILE, 78
  - LCMAPS\_MOD\_SUCCESS, 79
- lcmaps\_deleteVoData
  - lcmaps\_vo\_data.h, 122
- lcmaps\_deleteVoMapping
  - lcmaps\_vo\_data.h, 122
- lcmaps\_dir
  - lcmaps\_pluginmanager.c, 103
- LCMAPS\_ETC\_HOME
  - lcmaps\_defines.h, 77
- lcmaps\_extractRunVars
  - \_lcmaps\_runvars.h, 45
- lcmaps\_fill\_cred
  - \_lcmaps\_utils.h, 48
- lcmaps\_findArgName
  - lcmaps\_arguments.h, 63
- lcmaps\_findArgNameAndType
  - lcmaps\_arguments.h, 64
- lcmaps\_findfile
  - lcmaps\_utils.h, 114
- lcmaps\_genfilename
  - lcmaps\_utils.h, 114
- lcmaps\_get\_dn
  - lcmaps\_utils.h, 115
- lcmaps\_get\_gidlist
  - lcmaps\_utils.h, 115
- lcmaps\_get\_jobrep\_config
  - lcmaps\_jobrep.c, 83
- lcmaps\_get\_ldap\_pw
  - lcmaps\_ldap.c, 87
- lcmaps\_getArgValue
  - lcmaps\_arguments.h, 64
- lcmaps\_getfexist
  - lcmaps\_utils.h, 115
- lcmaps\_getRunVars
  - \_lcmaps\_runvars.h, 45
- lcmaps\_gss\_assist\_gridmap.c, 80
- lcmaps\_init
  - lcmaps.h, 59
- lcmaps\_initialized
  - lcmaps.c, 57
- lcmaps\_jobrep.c, 81
  - lcmaps\_get\_jobrep\_config, 83
  - plugin\_initialize, 83
  - plugin\_introspect, 83
  - plugin\_run, 84
  - plugin\_terminate, 84
- lcmaps\_ldap.c, 85
  - lcmaps\_add\_username\_to\_ldapgroup, 86
  - lcmaps\_get\_ldap\_pw, 87
  - lcmaps\_set\_pgid, 87
  - MAX\_LOG\_BUFFER\_SIZE, 86
  - timeout, 88
- LCMAPS\_LIB\_HOME
  - lcmaps\_defines.h, 77
- lcmaps\_localaccount.c, 89
- lcmaps\_log

- lcmaps\_log.h, 93
- lcmaps\_log.c, 90
  - DEBUG\_LEVEL, 91
  - debug\_level, 91
  - extra\_logstr, 91
  - lcmaps\_logfp, 91
  - logging\_syslog, 91
  - logging\_usrlog, 91
- lcmaps\_log.h, 92
  - lcmaps\_log, 93
  - lcmaps\_log\_debug, 93
  - lcmaps\_log\_time, 94
- lcmaps\_log\_close
  - \_lcmaps\_log.h, 39
- lcmaps\_log\_debug
  - lcmaps\_log.h, 93
- lcmaps\_log\_open
  - \_lcmaps\_log.h, 39
- lcmaps\_log\_time
  - lcmaps\_log.h, 94
- lcmaps\_logfp
  - lcmaps\_log.c, 91
- LCMAPS\_MAXARGS
  - lcmaps\_defines.h, 78
- LCMAPS\_MAXARGSTRING
  - lcmaps\_defines.h, 78
- LCMAPS\_MAXPATHLEN
  - lcmaps\_defines.h, 78
- LCMAPS\_MOD\_ENTRY
  - lcmaps\_defines.h, 78
- LCMAPS\_MOD\_FAIL
  - lcmaps\_defines.h, 78
- LCMAPS\_MOD\_HOME
  - lcmaps\_defines.h, 78
- LCMAPS\_MOD\_NOENTRY
  - lcmaps\_defines.h, 78
- LCMAPS\_MOD\_NOFILE
  - lcmaps\_defines.h, 78
- LCMAPS\_MOD\_SUCCESS
  - lcmaps\_defines.h, 79
- lcmaps\_modules.h, 95
- LCMAPS\_NO\_GID
  - lcmaps\_vo\_data.h, 120
- lcmaps\_plugin\_example.c, 97
- lcmaps\_pluginindl.s, 21
  - handle, 21
  - init\_argc, 21
  - init\_argv, 21
  - next, 21
  - pluginargs, 22
  - pluginname, 22
  - procs, 22
  - run\_argc, 22
  - run\_argv, 22
- lcmaps\_pluginindl.t
  - lcmaps\_pluginmanager.c, 100
- lcmaps\_pluginmanager.c
  - INITPROC, 101
  - INTROPROC, 101
  - RUNPROC, 101
  - TERMPROC, 101
- lcmaps\_pluginmanager.c, 98
  - clean\_plugin\_list, 101
  - get\_procsymbol, 102
  - lcmaps\_db\_file\_default, 103
  - lcmaps\_dir, 103
  - lcmaps\_pluginindl.t, 100
  - lcmaps\_proc.t, 100
  - lcmaps\_proctype\_e, 101
  - MAXPROCS, 100
  - NUL, 100
  - parse\_args\_plugin, 102
  - plugin\_list, 103
  - PluginInit, 101
  - print\_lcmaps\_plugin, 102
- lcmaps\_poolaccount.c, 104
- lcmaps\_posix.c, 105
- lcmaps\_printVoData
  - lcmaps\_vo\_data.h, 123
- lcmaps\_printVoMapping
  - lcmaps\_vo\_data.h, 123
- lcmaps\_proc.t
  - lcmaps\_pluginmanager.c, 100
- lcmaps\_proctype\_e
  - lcmaps\_pluginmanager.c, 101
- lcmaps\_release\_cred
  - \_lcmaps\_utils.h, 48
- lcmaps\_request.t
  - lcmaps\_types.h, 110
- lcmaps\_run
  - lcmaps.h, 59
- lcmaps\_run\_and\_return\_username
  - lcmaps.h, 59
- lcmaps\_run\_without\_credentials
  - lcmaps.h, 60
- lcmaps\_runvars.c, 106
  - runvars\_list, 107
- lcmaps\_set\_pgid
  - lcmaps\_ldap.c, 87
- lcmaps\_setArgValue
  - lcmaps\_arguments.h, 64
- lcmaps\_setRunVars
  - \_lcmaps\_runvars.h, 45
- lcmaps\_stringVoData
  - lcmaps\_vo\_data.h, 123
- lcmaps\_term
  - lcmaps.h, 60
- lcmaps\_test.c, 108

- lcmaps\_tokenize
  - \_lcmaps\_utils.h, 48
- lcmaps\_types.h, 109
  - lcmaps\_cred\_id\_t, 109
  - lcmaps\_request\_t, 110
- lcmaps\_utils.c, 111
  - cred\_to\_dn, 112
  - fexist, 112
- lcmaps\_utils.h, 113
  - lcmaps\_findfile, 114
  - lcmaps\_genfilename, 114
  - lcmaps\_get\_dn, 115
  - lcmaps\_get\_gidlist, 115
  - lcmaps\_getfexist, 115
- lcmaps\_vo\_data.c, 117
- lcmaps\_vo\_data.h, 118
  - lcmaps\_cleanVoData, 120
  - lcmaps\_cleanVoMapping, 120
  - lcmaps\_copyVoData, 121
  - lcmaps\_copyVoMapping, 121
  - lcmaps\_createVoData, 121
  - lcmaps\_createVoMapping, 122
  - lcmaps\_deleteVoData, 122
  - lcmaps\_deleteVoMapping, 122
  - LCMAPS\_NO\_GID, 120
  - lcmaps\_printVoData, 123
  - lcmaps\_printVoMapping, 123
  - lcmaps\_stringVoData, 123
  - lcmaps\_vo\_data\_t, 118
  - lcmaps\_vo\_mapping\_t, 118
- lcmaps\_vo\_data\_s, 23
  - capability, 23
  - group, 23
  - role, 23
  - subgroup, 23
  - vo, 23
- lcmaps\_vo\_data\_t
  - lcmaps\_vo\_data.h, 118
- lcmaps\_vo\_mapping\_s, 25
  - gid, 25
  - groupname, 25
  - vostring, 25
- lcmaps\_vo\_mapping\_t
  - lcmaps\_vo\_data.h, 118
- lcmaps\_voms.c, 124
- lcmaps\_voms\_localgroup.c, 126
- lcmaps\_voms\_poolaccount.c, 127
- lcmaps\_voms\_poolgroup.c, 128
- left\_side
  - pdl\_rule.h, 157
- level\_str
  - pdl\_main.c, 136
- lineno
  - pdl.h, 130
- pdl\_main.c, 136
- plugin\_s, 26
- policy\_s, 27
- record\_s, 28
- rule\_s, 29
- var\_s, 30
- logging\_syslog
  - lcmaps\_log.c, 91
- logging\_usrlog
  - lcmaps\_log.c, 91
- make\_list
  - pdl\_rule.c, 153
- MAX\_LOG\_BUFFER\_SIZE
  - \_lcmaps\_log.h, 39
  - lcmaps\_ldap.c, 86
- MAXARGS
  - \_lcmaps\_defines.h, 36
- MAXARGSTRING
  - \_lcmaps\_defines.h, 36
- MAXDBENTRIES
  - lcmaps\_db\_read.c, 71
- MAXPAIRS
  - lcmaps\_db\_read.c, 71
- MAXPATHLEN
  - \_lcmaps\_defines.h, 36
- MAXPROCS
  - lcmaps\_pluginmanager.c, 100
- name
  - plugin\_s, 26
  - policy\_s, 27
  - var\_s, 30
- next
  - lcmaps\_db\_entry\_s, 20
  - lcmaps\_plugindl\_s, 21
  - plugin\_s, 26
  - policy\_s, 27
  - rule\_s, 29
  - var\_s, 30
- NO\_RECURSION
  - pdl\_rule.h, 157
- NUL
  - lcmaps\_db\_read.c, 71
  - lcmaps\_pluginmanager.c, 100
- okay
  - var\_s, 30
- PAIR\_SEP\_CHARS
  - lcmaps\_db\_read.c, 71
- PAIR\_TERMINATOR\_CHARS
  - lcmaps\_db\_read.c, 72
- parse\_args\_plugin

- lcmaps\_pluginmanager.c, 102
- parse\_error
  - pdl\_main.c, 136
- path
  - pdl\_main.c, 136
- path\_lineno
  - pdl\_main.c, 136
- pdl.h, 129
  - concat\_strings, 132
  - concat\_strings\_with\_space, 132
  - EVALUATION\_FAILURE, 131
  - EVALUATION\_START, 131
  - EVALUATION\_SUCCESS, 131
  - free\_resources, 132
  - get\_plugins, 132
  - lineno, 130
  - PDL\_ERROR, 131
  - pdl\_error\_t, 131
  - PDL\_INFO, 131
  - pdl\_init, 132
  - pdl\_next\_plugin, 133
  - pdl\_path, 133
  - PDL\_SAME, 131
  - PDL\_UNKNOWN, 131
  - PDL\_WARNING, 131
  - plugin\_status\_t, 131
  - plugin\_t, 131
  - record\_t, 131
  - set\_path, 133
  - TRUE, 131
  - warning, 133
  - yyerror, 134
  - yyvsparse\_errors, 134
- PDL\_ERROR
  - pdl.h, 131
- pdl\_error\_t
  - pdl.h, 131
- PDL\_INFO
  - pdl.h, 131
- pdl\_init
  - pdl.h, 132
  - pdl\_main.c, 138
- pdl\_main.c, 135
  - \_concat\_strings, 137
  - \_set\_path, 137
  - concat\_strings, 137
  - concat\_strings\_with\_space, 137
  - d\_path, 136
  - default\_path, 136
  - find\_first\_space, 138
  - free\_path, 138
  - free\_resources, 138
  - get\_plugins, 138
  - level\_str, 136
  - lineno, 136
  - parse\_error, 136
  - path, 136
  - path\_lineno, 136
  - pdl\_init, 138
  - pdl\_next\_plugin, 139
  - pdl\_path, 139
  - plugin\_exists, 139
  - reduce\_policies, 139
  - script\_name, 136
  - set\_path, 140
  - top\_plugin, 136
  - warning, 140
  - yyerror, 140
  - yyvsparse\_errors, 140
- pdl\_next\_plugin
  - pdl.h, 133
  - pdl\_main.c, 139
- pdl\_path
  - pdl.h, 133
  - pdl\_main.c, 139
- pdl\_policy.c, 141
  - \_add\_policy, 142
  - add\_policy, 142
  - allow\_rules, 142
  - check\_policies\_for\_recursion, 143
  - current\_policy, 143
  - find\_policy, 143
  - free\_policies, 143
  - get\_policies, 143
  - policies\_have\_been\_reduced, 144
  - policies\_reduced, 141
  - reduce\_policies, 144
  - remove\_policy, 144
  - show\_policies, 144
- pdl\_policy.h, 145
  - add\_policy, 146
  - allow\_rules, 146
  - check\_policies\_for\_recursion, 147
  - current\_policy, 147
  - find\_policy, 147
  - free\_policies, 147
  - get\_policies, 147
  - policies\_have\_been\_reduced, 148
  - policy\_t, 146
  - reduce\_policies, 148
  - remove\_policy, 148
  - show\_policies, 148
- pdl\_rule.c, 149
  - \_add\_rule, 150
  - add\_rule, 150
  - allow\_new\_rules, 151
  - check\_rule\_for\_recursion, 151
  - count\_rules, 151

- find\_insert\_position, 151
- find\_state, 152
- free\_rules, 152
- get\_rule\_number, 152
- get\_top\_rule, 152
- has\_recursion, 152
- make\_list, 153
- reduce\_rule, 153
- rule\_number, 153
- set\_top\_rule, 154
- show\_rules, 154
- start\_new\_rules, 154
- update\_list, 154
- pdl\_rule.h
  - FALSE\_BRANCH, 157
  - left\_side, 157
  - NO\_RECURSION, 157
  - RECURSION, 157
  - RECURSION\_HANDLED, 157
  - right\_side, 157
  - STATE, 157
  - TRUE\_BRANCH, 157
- pdl\_rule.h, 155
  - add\_rule, 157
  - allow\_new\_rules, 157
  - check\_rule\_for\_recursion, 158
  - free\_rules, 158
  - get\_top\_rule, 158
  - recursion\_t, 157
  - reduce\_rule, 158
  - rule\_t, 156
  - rule\_type\_t, 157
  - set\_top\_rule, 158
  - show\_rules, 159
  - side\_t, 157
  - start\_new\_rules, 159
- PDL\_SAME
  - pdl.h, 131
- PDL\_UNKNOWN
  - pdl.h, 131
- pdl\_variable.c, 160
  - \_add\_variable, 161
  - add\_variable, 161
  - detect\_loop, 161
  - find\_variable, 162
  - free\_variables, 162
  - get\_variables, 162
  - reduce\_to\_var, 162
  - show\_variables, 162
- pdl\_variable.h, 164
  - add\_variable, 165
  - free\_variables, 165
  - get\_variables, 165
  - reduce\_to\_var, 166
  - show\_variables, 166
  - var\_t, 165
- PDL\_WARNING
  - pdl.h, 131
- plugin\_exists
  - pdl\_main.c, 139
- plugin\_initialize
  - lcmaps\_jobrep.c, 83
- plugin\_introspect
  - lcmaps\_jobrep.c, 83
- plugin\_list
  - lcmaps\_pluginmanager.c, 103
- plugin\_run
  - lcmaps\_jobrep.c, 84
- plugin\_s, 26
  - args, 26
  - lineno, 26
  - name, 26
  - next, 26
- plugin\_status\_t
  - pdl.h, 131
- plugin\_t
  - pdl.h, 131
- plugin\_terminate
  - lcmaps\_jobrep.c, 84
- pluginargs
  - lcmaps\_db\_entry\_s, 20
  - lcmaps\_plugindl\_s, 22
- PluginInit
  - lcmaps\_pluginmanager.c, 101
- pluginname
  - lcmaps\_db\_entry\_s, 20
  - lcmaps\_plugindl\_s, 22
- policies\_have\_been\_reduced
  - pdl\_policy.c, 144
  - pdl\_policy.h, 148
- policies\_reduced
  - pdl\_policy.c, 141
- policy\_s, 27
  - lineno, 27
  - name, 27
  - next, 27
  - prev, 27
  - rule, 27
- policy\_t
  - pdl\_policy.h, 146
- prev
  - policy\_s, 27
- priGid
  - cred\_data\_s, 17
- print\_lcmaps\_plugin
  - lcmaps\_pluginmanager.c, 102
- printCredData
  - lcmaps\_cred\_data.c, 66

- procs
  - lcmaps\_plugindl\_s, 22
- QUOTING\_CHARS
  - lcmaps\_db\_read.c, 72
- record\_s, 28
  - lineno, 28
  - string, 28
- record\_t
  - pdl.h, 131
- RECURSION
  - pdl\_rule.h, 157
- RECURSION\_HANDLED
  - pdl\_rule.h, 157
- recursion\_t
  - pdl\_rule.h, 157
- reduce\_policies
  - pdl\_main.c, 139
  - pdl\_policy.c, 144
  - pdl\_policy.h, 148
- reduce\_rule
  - pdl\_rule.c, 153
  - pdl\_rule.h, 158
- reduce\_to\_var
  - pdl\_variable.c, 162
  - pdl\_variable.h, 166
- remove\_policy
  - pdl\_policy.c, 144
  - pdl\_policy.h, 148
- resetCredentialData
  - \_lcmaps\_pluginmanager.h, 42
- right\_side
  - pdl\_rule.h, 157
- role
  - lcmaps\_vo\_data\_s, 23
- rule
  - policy\_s, 27
- rule\_number
  - pdl\_rule.c, 153
- rule\_s, 29
  - false\_branch, 29
  - lineno, 29
  - next, 29
  - state, 29
  - true\_branch, 29
- rule\_t
  - pdl\_rule.h, 156
- rule\_type\_t
  - pdl\_rule.h, 157
- run\_argc
  - lcmaps\_plugindl\_s, 22
- run\_argv
  - lcmaps\_plugindl\_s, 22
- runEvaluationManager
  - evaluationmanager.c, 51
  - evaluationmanager.h, 54
- runPlugin
  - \_lcmaps\_pluginmanager.h, 42
- runPluginManager
  - \_lcmaps\_pluginmanager.h, 42
- RUNPROC
  - lcmaps\_pluginmanager.c, 101
- runvars\_list
  - lcmaps\_runvars.c, 107
- script\_name
  - pdl\_main.c, 136
- secGid
  - cred\_data\_s, 17
- set\_path
  - pdl.h, 133
  - pdl\_main.c, 140
- set\_top\_rule
  - pdl\_rule.c, 154
  - pdl\_rule.h, 158
- show\_policies
  - pdl\_policy.c, 144
  - pdl\_policy.h, 148
- show\_rules
  - pdl\_rule.c, 154
  - pdl\_rule.h, 159
- show\_variables
  - pdl\_variable.c, 162
  - pdl\_variable.h, 166
- side\_t
  - pdl\_rule.h, 157
- start\_new\_rules
  - pdl\_rule.c, 154
  - pdl\_rule.h, 159
- startEvaluationManager
  - evaluationmanager.c, 51
  - evaluationmanager.h, 54
- startPluginManager
  - \_lcmaps\_pluginmanager.h, 43
- STATE
  - pdl\_rule.h, 157
- state
  - rule\_s, 29
- stopEvaluationManager
  - evaluationmanager.c, 52
  - evaluationmanager.h, 54
- stopPluginManager
  - \_lcmaps\_pluginmanager.h, 43
- string
  - record\_s, 28
- subgroup
  - lcmaps\_vo\_data\_s, 23

- TERMPROC
  - lcmaps\_pluginmanager.c, 101
- The API to be used by the LCMAPS plugins, 12
- The interface to the LCMAPS plugins, 13
- timeout
  - lcmaps\_ldap.c, 88
- top\_plugin
  - pdl\_main.c, 136
- TRUE
  - pdl.h, 131
- TRUE\_BRANCH
  - pdl\_rule.h, 157
- true\_branch
  - rule\_s, 29
  
- uid
  - cred\_data\_s, 17
- update\_list
  - pdl\_rule.c, 154
  
- value
  - lcmaps\_argument\_s, 18
  - var\_s, 30
- var\_s, 30
  - lineno, 30
  - name, 30
  - next, 30
  - okay, 30
  - value, 30
- var\_t
  - pdl\_variable.h, 165
- VARVAL\_SEP\_CHARS
  - lcmaps\_db\_read.c, 72
- VARVAL\_TERMINATOR\_CHARS
  - lcmaps\_db\_read.c, 72
- vo
  - lcmaps\_vo\_data\_s, 23
- VoCred
  - cred\_data\_s, 16
- VoCredMapping
  - cred\_data\_s, 16
- VoCredString
  - cred\_data\_s, 16
- vostring
  - lcmaps\_vo\_mapping\_s, 25
  
- warning
  - pdl.h, 133
  - pdl\_main.c, 140
- WHITESPACE\_CHARS
  - lcmaps\_db\_read.c, 72
  
- yyerror
  - pdl.h, 134
  
- pdl\_main.c, 140
- yyparse\_errors
  - pdl.h, 134
  - pdl\_main.c, 140