logmanager library

Generated by Doxygen 1.7.1

Thu Jul 15 2010 11:32:05

Contents

Chapter 1

Main Page

A PDF version of this document is available at http://managelogs.tekwire.net/doc/api.pdf

Chapter 2

Todo List

Global LMGR_IGNORE_ENOSPC Should be able to send an alarm via an external system on ENOSPC ignored errors.

Chapter 3

Data Structure Index

3.1 Data Structures

Here are the data structures with brief descriptions:

Chapter 4

File Index

4.1 File List

Here is a list of all files with brief descriptions:

Chapter 5

Data Structure Documentation

5.1 LOGMANAGER Struct Reference

The structure identifying a log manager.

`#include <logmanager.h>`

Data Fields

- int dummy

5.1.1 Detailed Description

The structure identifying a log manager. This is an opaque structure whose elements are not directly accessible from the calling program

Examples:

example.c.

5.1.2 Field Documentation

5.1.2.1 int LOGMANAGER::dummy

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

- logmanager.h

5.2 LOGMANAGER_OPTIONS Struct Reference

Log manager configuration provided at creation time as an argument to new_logmanager().

`#include <logmanager.h>`

Data Fields

- unsigned int api_version

  Must be set to LOGMANAGER_API_VERSION.

- char * base_path

  The base path.

- unsigned int flags

  An OR-ed combination of LMGR_* flags.

- struct {
  char * type
  Compression type : 'gz' or 'bz2'.
  char * level
  Compression level as a string.
  } compress

- apr_off_t file_maxsize

  Max log file individual size.

- apr_off_t global_maxsize

  Global size limit (sum).

- unsigned int keep_count

  Maximum number of log files to keep.

- apr_fileperms_t create_mode

  File creation mode.

- char * debug_file

  Path to debug file.

- int debug_level

  Debug level (>= 0).

- char * rotate_cmd

  Command to run on each rotation.

- TIMESTAMP rotate_delay

  Max delay between rotations (in seconds).

- TIMESTAMP purge_delay

  Max log file age (in seconds).

- char * log_path

  If set, overrides base path to compute log file paths.

5.2.1 Detailed Description

Log manager configuration provided at creation time as an argument to new_logmanager().

Examples:

example.c.

5.2.2 Field Documentation

5.2.2.1 unsigned int LOGMANAGER_OPTIONS::api_version

Must be set to LOGMANAGER_API_VERSION.

Examples:

example.c.

5.2.2.2 char* LOGMANAGER_OPTIONS::base_path

The base path.

This is the prefix used to compute link, status, and pid file paths. If log path not set, also used to compute log file paths

Examples:

example.c.

5.2.2.3 unsigned int LOGMANAGER_OPTIONS::flags

An OR-ed combination of LMGR_* flags.

Examples:

example.c.

5.2.2.4 char* LOGMANAGER_OPTIONS::type

Compression type : 'gz' or 'bz2'.

5.2.2.5 char* LOGMANAGER_OPTIONS::level

Compression level as a string.

Possible values depend on the compression type

5.2.2.6 struct { ... } LOGMANAGER_OPTIONS::compress

5.2.2.7 apr_off_t LOGMANAGER_OPTIONS::file_maxsize

Max log file individual size.

When returning from new_logmanager(), the value can be modified if it does not fit internal constraints. Default = 0 = no limit

Examples:

example.c.

5.2.2.8 apr_off_t LOGMANAGER_OPTIONS::global_maxsize

Global size limit (sum).

Default = 0 = no limit

Examples:

example.c.

5.2.2.9 unsigned int LOGMANAGER_OPTIONS::keep_count

Maximum number of log files to keep.

Default = 0 = no limit

5.2.2.10 apr_fileperms_t LOGMANAGER_OPTIONS::create_mode

File creation mode.

Note

mode is a unix-style permission expressed as an hexadecimal number (example: 0x644). Default = 0 => 0x644

5.2.2.11 char* LOGMANAGER_OPTIONS::debug_file

Path to debug file.

Default = NULL = no debug file

5.2.2.12 int LOGMANAGER_OPTIONS::debug_level

Debug level (>= 0).

Default = 0 = no debug output

5.2.2.13 char* LOGMANAGER_OPTIONS::rotate_cmd

Command to run on each rotation.

5.2.2.14 TIMESTAMP LOGMANAGER_OPTIONS::rotate_delay

Max delay between rotations (in seconds).

Default = 0 = not set

5.2.2.15 TIMESTAMP LOGMANAGER_OPTIONS::purge_delay

Max log file age (in seconds).

Default = 0 = not set

5.2.2.16 char* LOGMANAGER_OPTIONS::log_path

If set, overrides base path to compute log file paths.

Default (NULL) => use base path

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

- logmanager.h

Chapter 6

File Documentation

6.1 logmanager.h File Reference

```
#include <apr.h>
#include <apr_file_io.h>
#include <apr_time.h>
```

Data Structures

- struct LOGMANAGER_OPTIONS

  Log manager configuration provided at creation time as an argument to new_logmanager().

- struct LOGMANAGER

  The structure identifying a log manager.

Defines

- #define NOW (TIMESTAMP)0

  Use the system current time.

- #define LMGR_ACTIVE_LINK 0x01

  LOGMANAGER_OPTIONS flag - Maintain a link to the active log file.

- #define LMGR_BACKUP_LINKS 0x02

  LOGMANAGER_OPTIONS flag - Maintain links to backup logs.

- #define LMGR_HARD_LINKS 0x04

  LOGMANAGER_OPTIONS flag - Create hard links instead of symbolic links.

- #define LMGR_IGNORE_EOL 0x08

  LOGMANAGER_OPTIONS flag - Don't rotate on eol only.

- #define LMGR_IGNORE_ENOSPC 0x10

LOGMANAGER_OPTIONS flag - ignore 'no more space' (ENOSPC) system errors.

- #define LMGR_PID_FILE 0x20

  LOGMANAGER_OPTIONS flag - Maintain a PID file as 'base_path'.pid.

- #define LOGMANAGER_API_VERSION 3

  The API version.

Typedefs

- typedef apr_time_t TIMESTAMP

  Timestamp (apr_time_t value or 'NOW').

Functions

- LOGMANAGER * new_logmanager (LOGMANAGER_OPTIONS *opts)

  Creates a log manager.

- void logmanager_destroy (LOGMANAGER *mp)

  Deletes a log manager.

- void logmanager_open (LOGMANAGER *mp, TIMESTAMP t)

  Opens a log manager.

- void logmanager_close (LOGMANAGER *mp)

  Closes a log manager.

- void logmanager_write (LOGMANAGER *mp, const char *buf, apr_off_t size, TIMESTAMP t)

  Sends data to a log manager.

- void logmanager_flush (LOGMANAGER *mp)

  Forces a flush to the log file.

- void logmanager_rotate (LOGMANAGER *mp, TIMESTAMP t)

  Triggers a rotation.

- char * logmanager_compression_list (void)

  Returns the list of supported compression schemes.

- char * logmanager_version (void)

  Returns the version of the library.

- void logmanager_display_stats (LOGMANAGER *mp)

  Displays internal stats.

6.1.1 Define Documentation

6.1.1.1 #define NOW (TIMESTAMP)0

Use the system current time.

When a function receives 'NOW' as timestamp argument, it converts it into the current system time.

Examples:

example.c.

6.1.1.2 #define LMGR_ACTIVE_LINK 0x01

LOGMANAGER_OPTIONS flag - Maintain a link to the active log file.

6.1.1.3 #define LMGR_BACKUP_LINKS 0x02

LOGMANAGER_OPTIONS flag - Maintain links to backup logs.

6.1.1.4 #define LMGR_HARD_LINKS 0x04

LOGMANAGER_OPTIONS flag - Create hard links instead of symbolic links.

6.1.1.5 #define LMGR_IGNORE_EOL 0x08

LOGMANAGER_OPTIONS flag - Don't rotate on eol only.

This flag inhibits the default buffering mechanism ensuring that rotations occur on end of lines only (no truncated last line in log files)

6.1.1.6 #define LMGR_IGNORE_ENOSPC 0x10

LOGMANAGER_OPTIONS flag - ignore 'no more space' (ENOSPC) system errors.

When this flag is set, the log manager ignores 'no more space/file system full' errors, silently discarding new data when no more space remains in the log file system.

Before setting this flag, you must be sure that you prefer discarding data than crashing the service. Can be used as a 'last chance' protection against DOS attacks.

Todo

Should be able to send an alarm via an external system on ENOSPC ignored errors.

6.1.1.7 #define LMGR_PID_FILE 0x20

LOGMANAGER_OPTIONS flag - Maintain a PID file as 'base_path'.pid.

Examples:

example.c.

6.1.1.8 #define LOGMANAGER_API_VERSION 3

The API version.

6.1.2 Typedef Documentation

6.1.2.1 typedef apr_time_t TIMESTAMP

Timestamp (apr_time_t value or 'NOW').

The logmanager library functions receive their time information from the calling program, instead of getting it from the system clock. This way, the calling program controls the way time sent to a log manager varies, independantly from the 'real' (system) time. Different log managers can receive different timestamps as they are independant from each other.

This allows, for instance, to process some old data with the time information extracted from each data line.

Note

Once a timestamp is provided in a function call, a subsequent call cannot provide a smaller timestamp value (no step back in time). If a timestamp value smaller than the previous one is provided, the previous one is used to ensure that the rotation/purge logic remains functional.

6.1.3 Function Documentation

6.1.3.1 LOGMANAGER* new_logmanager (LOGMANAGER_OPTIONS * *opts*)

Creates a log manager.

This function creates a log manager object from a set of options. The options define its logic and the paths it will maintain.

Note

This function does not write anything to the file system. It only allocates and initializes a LOGMANAGER struct in memory. Before sending data to the newly-created log manager, you must open it via a call to logmanager_open().

Parameters

`[in]` ***opts*** A pointer to the log manager's options. This struct and the strings/pointers it may contain should be freed on return (every data needed by the log manager for internal use are transparently allocated/freed by the log manager).

Returns

LOGMANAGER* A pointer to a newly allocated LOGMANAGER struct. This is the pointer to use in subsequent function calls.

Examples:

example.c.

6.1.3.2 void logmanager_destroy (LOGMANAGER * *mp*)

Deletes a log manager.

Also frees the data allocated by new_logmanager()

Parameters

[in] ***mp*** Pointer to a LOGMANAGER struct previously returned by new_logmanager().

Examples:

example.c.

6.1.3.3 void logmanager_open (LOGMANAGER * *mp*, TIMESTAMP *t*)

Opens a log manager.

Called after new_logmanager() before writing to a log manager

Parameters

[in] ***mp*** Pointer to a LOGMANAGER struct previously returned by new_logmanager().

[in] ***t*** Timestamp (apr_time_t value or 'NOW')

Examples:

example.c.

6.1.3.4 void logmanager_close (LOGMANAGER * *mp*)

Closes a log manager.

The log manager can then be reopened via logmanager_open() or destroyed via logmanager_destroy().

Parameters

[in] ***mp*** Pointer to a LOGMANAGER struct previously returned by new_logmanager().

Examples:

example.c.

6.1.3.5 void logmanager_write (LOGMANAGER * *mp*, const char * *buf*, apr_off_t *size*, TIMESTAMP *t*)

Sends data to a log manager.

Parameters

[in] ***mp*** Pointer to a LOGMANAGER struct previously returned by new_logmanager().

[in] ***buf*** The data to write.

[in] ***size*** The size in bytes of the data buffer to write

[in] ***t*** Timestamp (apr_time_t value or 'NOW')

Examples:

example.c.

6.1.3.6 void logmanager_flush (LOGMANAGER * *mp*)

Forces a flush to the log file.

Flushes the remaining data to the file system. Mostly interesting to flush compressed streams so that they can be uncompressed by an external mechanism.

Note

The log manager remains open.

Parameters

[in] ***mp*** Pointer to a LOGMANAGER struct previously returned by new_logmanager().

6.1.3.7 void logmanager_rotate (LOGMANAGER * *mp*, TIMESTAMP *t*)

Triggers a rotation.

Triggers an immediate log rotation. Can also trigger a purge if the rotation causes the purge constraints to be exceeded.

Parameters

[in] ***mp*** Pointer to a LOGMANAGER struct previously returned by new_logmanager().

[in] ***t*** Timestamp (apr_time_t value or 'NOW')

6.1.3.8 char* logmanager_compression_list (void)

Returns the list of supported compression schemes.

Returns

A dynamically allocated string containing the list of supported compression types, separated with ',' characters.

The caller has the responsibility to free the returned string.

6.1.3.9 char* logmanager_version (void)

Returns the version of the library.

Returns

the version of the managelogs package the library belongs to.

The caller has the responsibility to free the returned string.

6.1.3.10 void logmanager_display_stats (LOGMANAGER * *mp*)

Displays internal stats.

Utility function displaying some internal stats and counters. Used for debugging and tests only.

Parameters

`[in]` ***mp*** Pointer to a LOGMANAGER struct previously returned by new_logmanager().

Chapter 7

Example Documentation

7.1 example.c

```
/*=========================================================================
*
* This is a very basic example showing how to create a log manager, send it
* some data read from standard input, and then close and destroy it.
*
*=========================================================================
*/

#include <stdio.h>
#include "logmanager.h"

/*---------*/

int main()
{
LOGMANAGER *mp;
LOGMANAGER_OPTIONS opts;
char *buf;
int nread;

/* First, initialize the opts struct */

memset(&opts,0,sizeof(opts));
opts.api_version=LOGMANAGER_API_VERSION; /* Mandatory */

/* Example : set the log manager to maintain a pid file a symbolic to the
* active log, limit individual file size to 1 Mb and global size to 10 Mb,
* and write the log files in '/some/dir' with a filename starting with
* 'prefix' */

opts.flags=LMGR_PID_FILE|LMGR_ACTIVE_LINK;
opts.file_maxsize=1024*1024;
opts.global_maxsize=10*opts.file_maxsize;
opts.base_path="/some/dir/prefix";

/* Create and open the log manager */

mp=new_logmanager(&opts);
logmanager_open(mp,NOW);

/* Here, in a real program, we would free the mem allocated for the opts struct *
      /

/* Write loop */
```

```
while((nread=read(0,buf,sizeof(buf))) > 0)
    {
    logmanager_write(mp,buf,nread,NOW);
    }

/* The end: Close and destroy the log manager */

logmanager_close(mp);
logmanager_destroy(mp);

return 0;
}
```

Index