cvsntLCP Reference Manual

0.0.1

Generated by Doxygen 1.5.1

Sat Mar 10 10:09:19 2007

Contents

Chapter 1

cvsntLCP Class Index

1.1 cvsntLCP Class List

Here are the classes, structs, unions and interfaces with brief descriptions:

Chapter 2

cvsntLCP File Index

2.1 cvsntLCP File List

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

Chapter 3

cvsntLCP Class Documentation

3.1 Options Class Reference

Options class.

`#include <Options.hpp>`

Public Member Functions

- Options ()
  Constructor.
- ~Options ()
  Destructor.
- int readOptions (void)
  Read the options from file.
- int writeOptions (void)
  Write the current options to file.
- char * getInetdName (void)
  Return the currently used name of inetd.
- void setInetdName (char *)
  Set new name of inetd.
- char * getCfgFilename (void)
  Return name of used CVSNT configuration file.
- void setCfgFilename (char *)
  Set name for used CVSNT configuration file.
- bool getBackupFlag (void)
  Return backup flag.

- void setBackupFlag (bool)

  Set backup flag.

- char * getBackupExt (void)

  Return file extension for backup file.

- void setBackupExt (char *)

  Set new file extension for backup file.

3.1.1 Detailed Description

Options class.

This class is responsible for the CVSNT LCP internal options data. This includes reading and writing the options as well as providing the data in a useful form for the GUI

3.1.2 Constructor & Destructor Documentation

3.1.2.1 Options::Options ()

Constructor.

The constructor creates an instance of the parser and reads the option file. If it does not exist, the internal values will be pre-set with the defaults value and the option file will be created

3.1.2.2 Options::~Options ()

Destructor.

The destructor cleans up the used memory

3.1.3 Member Function Documentation

3.1.3.1 int Options::readOptions (void)

Read the options from file.

This method reads the option file and parses the values found into the internal data

Return values:

NO_ERROR Option file successfully read

COULD_NOT_OPEN_FILE The option file could not be found

NO_VALID_ENTRIES The option file contained not one valid entry

3.1.3.2 int Options::writeOptions (void)

Write the current options to file.

This methods writes the current internal option data to the option file

Return values:

NO_ERROR Options successfully written

3.1.3.3 void Options::setInetdName (char *)

Set new name of inetd.

Parameters:

← ***newInetd*** New name for inetd

3.1.3.4 void Options::setCfgFilename (char *)

Set name for used CVSNT configuration file.

Parameters:

← ***newCfgFile*** New name for used CVSNT configuration file

3.1.3.5 bool Options::getBackupFlag (void)

Return backup flag.

Return flag whether or not the configuration file should be backed up or not

3.1.3.6 void Options::setBackupFlag (bool)

Set backup flag.

Set flag whether or not the original configuration file should be backed up before writing

Parameters:

← ***backupFlag*** New value for backup flag

3.1.3.7 void Options::setBackupExt (char *)

Set new file extension for backup file.

Parameters:

← ***newExt*** New file extension

The documentation for this class was generated from the following files:

- Programming/Projects/GladeProjects/cvsntlcp/src/Options.hpp
- Programming/Projects/GladeProjects/cvsntlcp/src/Options.cpp

3.2 Parser Class Reference

Parser class.

`#include <Parser.hpp>`

Public Member Functions

- Parser (char *)

  Constructor.

- Parser ()
- ~Parser ()

  Destructor.

- int setConfigFile (char *)

  Set (new) configuration file.

- char * getConfigFile (void)

  Returns the currently used filename.

- char * findValue (char *)

  Find the value of a given property.

3.2.1 Detailed Description

Parser class.

This class is responsible to load and parse a configuration file. It stores all properties and values internally and returns a correspondig value of a given property

3.2.2 Constructor & Destructor Documentation

3.2.2.1 Parser::Parser (char *)

Constructor.

Sets up the class

Parameters:

← ***configFile*** Configuration file to parse

3.2.2.2 Parser::Parser ()

This is an overloaded member function, provided for convenience. It differs from the above function only in what argument(s) it accepts.

3.2.2.3 Parser::~Parser ()

Destructor.

The destructor disposes the memory used by the class

3.2.3 Member Function Documentation

3.2.3.1 int Parser::setConfigFile (char *)

Set (new) configuration file.

This method accepts a filename for a (new) configuration file to parse. It discards all old data and opens and parses the new file

Parameters:

← ***newFile*** Filename of (new) configuration file

Return values:

NO_ERROR New configuration file successfully parsed

COULD_NOT_OPEN_FILE The given file could not be opened

NO_VALID_ENTRIES No valid entries were found in the given file

3.2.3.2 char * Parser::getConfigFile (void)

Returns the currently used filename.

Returns:

Filename of configuration file

3.2.3.3 char * Parser::findValue (char *)

Find the value of a given property.

This method returns the value of the given property or NULL if the given property could not be found in the list of all properties

Parameters:

← ***prop*** Property

Return values:

<***value***> Value of given property

NULL If given property was not found in the list

The documentation for this class was generated from the following files:

- Programming/Projects/GladeProjects/cvsntlcp/src/Parser.hpp
- Programming/Projects/GladeProjects/cvsntlcp/src/Parser.cpp

Chapter 4

cvsntLCP File Documentation

4.1 Programming/Projects/GladeProjects/cvsntlcp/src/Errors.hpp File Reference

Defines

- #define NO_ERROR 0

  Success code.

- #define COULD_NOT_OPEN_FILE -1001

  The requested file could not be opened.

- #define NO_VALID_ENTRIES -1002

  The given file contains no entries.

Functions

- bool no_error (int code)

  Checks if the given code means NO_ERROR.

- bool success (int code)

  Checks if the given code returned a success code.

- bool failed (int code)

  Checks if the code is a warning or error code.

- bool is_warning (int code)

  Checks if the given code is a warning code.

- bool is_error (int code)

  Checks if the given code means error.

4.1.1 Detailed Description

This file contains a number of status, warning and error codes as well as some inline functions to handle them easily

Author:

Andreas Tscharner

Date:

2006-10-01

4.1.2 Function Documentation

4.1.2.1 bool failed (int *code*) [inline]

Checks if the code is a warning or error code.

Parameters:

← ***code*** Status, warning or error code

Return values:

true Code means error or warning

false Code does not mean error or warning

4.1.2.2 bool is_error (int *code*) [inline]

Checks if the given code means error.

Parameters:

← ***code*** Status, warning or error code

Return values:

true Code is error code

false Code means not an error

4.1.2.3 bool is_warning (int *code*) [inline]

Checks if the given code is a warning code.

Parameters:

← ***code*** Status, warning or error code

Return values:

true Code is a warning code

false Code is not warning

4.1.2.4 bool no_error (int *code*) [inline]

Checks if the given code means NO_ERROR.

Parameters:

← ***code*** Status, warning or error code

Return values:

true Code means NO_ERROR

false Code means not NO_ERROR

4.1.2.5 bool success (int *code*) [inline]

Checks if the given code returned a success code.

Parameters:

← ***code*** Status, warning or error code

Return values:

true Code is success code

false Code is not success code

4.2 Programming/Projects/GladeProjects/cvsntlcp/src/Macros.hpp File Reference

```
#include <stdio.h>
```

Defines

- #define DEBUG_PRINT(x)

  Macro to print information.

- #define START_C_FUNC extern "C" {

  Macro to start a C function within a C++ context.

- #define END_C_FUNC }

  Macro to end a C function.

4.2.1 Detailed Description

This file contains the some useful macros

Author:

Andreas Tscharner

Date:

2006-12-30

4.2.2 Define Documentation

4.2.2.1 #define DEBUG_PRINT(x)

Macro to print information.

This macro prints out information when in debug mode (provide –enable-debug to ./configure). It expands to nothing if it's a release version.

Parameters:

← ***x*** Text (including the parentheses) to be passed to printf

4.2.2.2 #define START_C_FUNC extern "C" {

Macro to start a C function within a C++ context.

This macro starts a C function in a C++ file

4.3 Programming/Projects/GladeProjects/cvsntlcp/src/Options.hpp File Reference

```
#include "Parser.hpp"
```

Classes

- class Options

  Options class.

Defines

- #define OPTION_FILE ".cvsntlpcrc"

  Name of option file.

- #define INETD_NAME "inetd"

  Default inetd server name.

- #define CVSNT_CFG_FILE "/etc/cvsnt/PServer"

  Default CVSNT configuration file.

- #define BACKUP_DEFAULT true

  Default setting for backup of config file.

- #define BACK_EXT ".bak"

  Default backup extension for config file.

- #define INETD_CFG_ENTRY "INetServer"

  Name for inetd server in options file.

- #define CVSNTCFG_CFG_ENTRY "ConfigFile"

  Name for CVSNT config file in options file.

- #define BACKUP_CFG_ENTRY "BackupFile"

  Name for backup flag in options file.

- #define BACKEXT_CFG_ENTRY "BackupExt"

  Name for backup extension in options file.

4.3.1 Detailed Description

This file contains the definition of the options class

Author:

Andreas Tscharner

Date:

2006-11-12

4.4 Programming/Projects/GladeProjects/cvsntlcp/src/Options-Dlg.cc File Reference

```
#include "Options.hpp"
#include "Macros.hpp"
#include <gtk/gtk.h>
#include <glade/glade.h>
```

4.4.1 Detailed Description

This file contains the dialog handling of the options dialog

Author:

Andreas Tscharner

Date:

2006-12-30

4.5 Programming/Projects/GladeProjects/cvsntlcp/src/Parser.cpp File Reference

```
#include "Parser.hpp"
#include "Errors.hpp"
#include <stdio.h>
#include <string.h>
```

4.5.1 Detailed Description

This file contains the implementation of the parser class

Author:

Andreas Tscharner

Date:

2006-09-25

4.6 Programming/Projects/GladeProjects/cvsntlcp/src/Parser.hpp File Reference

Classes

- class Parser

  Parser class.

4.6.1 Detailed Description

This file contains the definition of the parser class

Author:

Andreas Tscharner

Date:

2006-09-02

Index