Crypto

Generated by Doxygen 1.8.6

Fri Jun 28 2019 10:55:09

Contents

Chapter 1

Data encryption program Crypto.

Crypto - Advanced File Encryptor, based on simple XOR and reliable AES methods

Author

SergejBre sergej1@email.ua

Chapter 2

Crypto - Advanced File Encryptor, based on one simple XOR and on a reliable AES encryption method

Description of the Project

TODO

Project folders

TODO

Main features

TODO

Specification

TODO

System requirements

TODO

Compilation and installation

Here is a link to the installation program for Windows XP, Vista, 7, 8, 8.1 `crypto-setup.zip`

Further development

TODO

Chapter 3

Todo List

Member MainWindow::execute (void)

Password salt is taken from the release time of the program, taken in microseconds.

Member MainWindow::fileProcessing (const QString &file)

Make an extension for encrypted files! (".enc")

Member MainWindow::on_actionContents_triggered (void)

Add help for the program!

Member SettingsDialog::fillSettings (void)

The method of checking and reading from the configuration file.

Member SettingsDialog::fillSettingsUi (void)

Validation method for configuration data.

Member SettingsDialog::updateSettings (void)

Alternative method to set default data!

Chapter 4

Namespace Index

4.1 Namespace List

Here is a list of all namespaces with brief descriptions:

Chapter 5

Hierarchical Index

5.1 Class Hierarchy

This inheritance list is sorted roughly, but not completely, alphabetically:

Chapter 6

Class Index

6.1 Class List

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

Chapter 7

File Index

7.1 File List

Here is a list of all files with brief descriptions:

Chapter 8

Namespace Documentation

8.1 Ui Namespace Reference

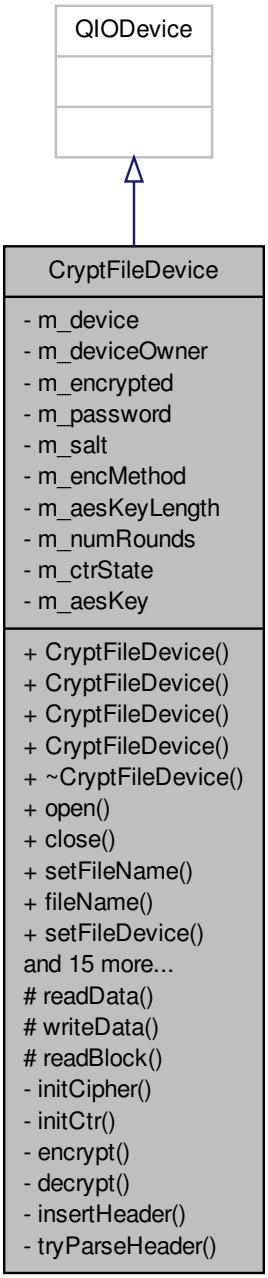
Chapter 9

Class Documentation

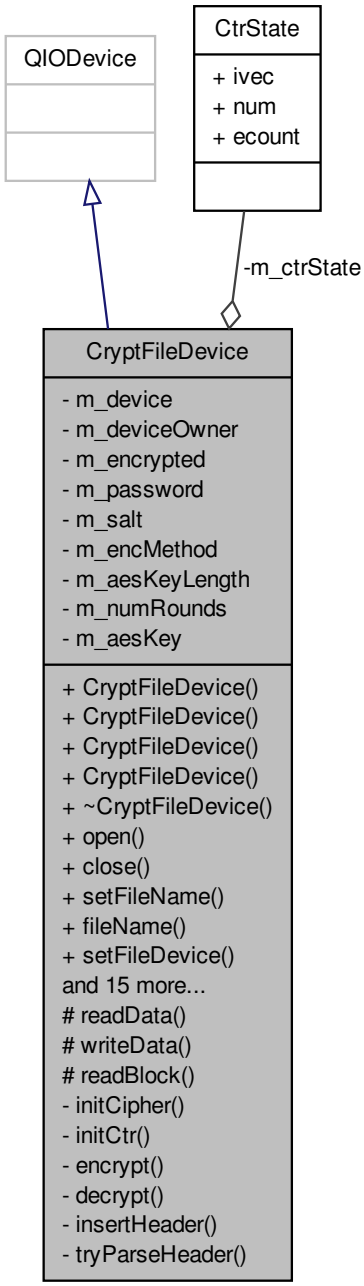
9.1 CryptFileDevice Class Reference

```
#include <cryptfiledevice.h>
```

Inheritance diagram for CryptFileDevice:



Collaboration diagram for CryptFileDevice:



Public Types

- enum AesKeyLength { kAesKeyLength128, kAesKeyLength192, kAesKeyLength256 }
- enum EncryptionMethod { XorCipher, AesCipher }

Signals

- void errorMessage (const QVariant &msg) const

Public Member Functions

- CryptFileDevice (QObject ∗parent=0)
- CryptFileDevice (QFileDevice ∗device, QObject ∗parent=0)
- CryptFileDevice (QFileDevice ∗device, const QByteArray &password, const QByteArray &salt, QObject ∗parent=0)
- CryptFileDevice (const QString &fileName, const QByteArray &password, const QByteArray &salt, QObject ∗parent=0)
- ∼CryptFileDevice ()
- bool open (OpenMode flags)

  CryptFileDevice::open.
- void close (void)

  CryptFileDevice::close.
- void setFileName (const QString &fileName)

  CryptFileDevice::setFileName.
- QString fileName (void) const

  CryptFileDevice::fileName.
- void setFileDevice (QFileDevice ∗device)

  CryptFileDevice::setFileDevice.
- void setPassword (const QByteArray &password)

  CryptFileDevice::setPassword.
- void setSalt (const QByteArray &salt)

  CryptFileDevice::setSalt.
- void setKeyLength (AesKeyLength keyLength)

  CryptFileDevice::setKeyLength.
- void setNumRounds (int numRounds)

  CryptFileDevice::setNumRounds.
- void setEncryptionMethod (EncryptionMethod enc)

  CryptFileDevice::setEncryptionMethod.
- bool isEncrypted (void) const

  CryptFileDevice::isEncrypted.
- qint64 size (void) const

  CryptFileDevice::size.
- bool atEnd (void) const

  CryptFileDevice::atEnd.
- qint64 bytesAvailable (void) const

  CryptFileDevice::bytesAvailable.
- qint64 pos (void) const

  CryptFileDevice::pos.
- bool seek (qint64 pos)

  CryptFileDevice::seek.
- bool flush (void)

  CryptFileDevice::flush.
- bool remove (void)

  CryptFileDevice::remove.
- bool exists (void) const

  CryptFileDevice::exists.
- bool rename (const QString &newName)

  CryptFileDevice::rename.

Protected Member Functions

- qint64 readData (char *data, qint64 length)
  CryptFileDevice::readData.
- qint64 writeData (const char *data, qint64 length)
  CryptFileDevice::writeData.
- qint64 readBlock (qint64 length, QByteArray &block)
  CryptFileDevice::readBlock.

Private Member Functions

- bool initCipher (void)
  CryptFileDevice::initCipher.
- void initCtr (CtrState *state, const unsigned char *iv)
  CryptFileDevice::initCtr.
- char * encrypt (const char *plainText, qint64 length)
  CryptFileDevice::encrypt.
- char * decrypt (const char *cipherText, qint64 length)
  CryptFileDevice::decrypt.
- void insertHeader (void)
  CryptFileDevice::insertHeader.
- bool tryParseHeader (void)
  CryptFileDevice::tryParseHeader.

Private Attributes

- QFileDevice * m_device = nullptr
- bool m_deviceOwner = false
- bool m_encrypted = false
- QByteArray m_password
- QByteArray m_salt
- EncryptionMethod m_encMethod
- AesKeyLength m_aesKeyLength = kAesKeyLength256
- int m_numRounds = 5
- CtrState m_ctrState
- AES_KEY m_aesKey

9.1.1 Member Enumeration Documentation

9.1.1.1 enum CryptFileDevice::AesKeyLength

Enumerator

kAesKeyLength128

kAesKeyLength192

kAesKeyLength256

9.1.1.2 enum CryptFileDevice::EncryptionMethod

Enumerator

XorCipher

AesCipher

9.1.2 Constructor & Destructor Documentation

9.1.2.1 CryptFileDevice::CryptFileDevice (QObject * *parent* = 0) [explicit]

9.1.2.2 CryptFileDevice::CryptFileDevice (QFileDevice * *device,* QObject * *parent* = 0) [explicit]

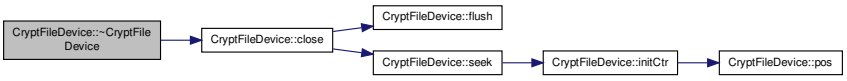
9.1.2.3 CryptFileDevice::CryptFileDevice (QFileDevice * *device,* const QByteArray & *password,* const QByteArray & *salt,* QObject * *parent* = 0) [explicit]

9.1.2.4 CryptFileDevice::CryptFileDevice (const QString & *fileName,* const QByteArray & *password,* const QByteArray & *salt,* QObject * *parent* = 0) [explicit]

9.1.2.5 CryptFileDevice::~CryptFileDevice ( )

References close(), m_device, and m_deviceOwner.

Here is the call graph for this function:



9.1.3 Member Function Documentation

9.1.3.1 bool CryptFileDevice::atEnd (void) const

CryptFileDevice::atEnd.

Returns

9.1.3.2 qint64 CryptFileDevice::bytesAvailable (void) const

CryptFileDevice::bytesAvailable.

Returns

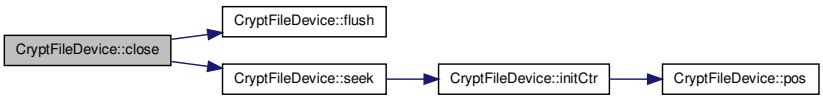
9.1.3.3 void CryptFileDevice::close (void)

CryptFileDevice::close.

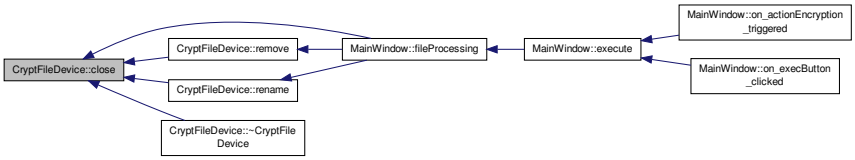
References flush(), m_device, m_encrypted, and seek().

Referenced by MainWindow::fileProcessing(), remove(), rename(), and ~CryptFileDevice().

Here is the call graph for this function:



Here is the caller graph for this function:



9.1.3.4 char * CryptFileDevice::decrypt (const char * *cipherText*, qint64 *len*) [private]

CryptFileDevice::decrypt.

Parameters

cipherText	
len	

Returns

References CtrState::ecount, CtrState::ivec, m_aesKey, m_ctrState, and CtrState::num.

Referenced by readBlock().

Here is the caller graph for this function:

9.1.3.5 char * CryptFileDevice::encrypt (const char * *plainText*, qint64 *length*) [private]

CryptFileDevice::encrypt.

Parameters

plainText	
length	

Returns

References AesCipher, CtrState::ecount, errorMessage(), CtrState::ivec, m_aesKey, m_ctrState, m_encMethod, m_password, CtrState::num, and XorCipher.

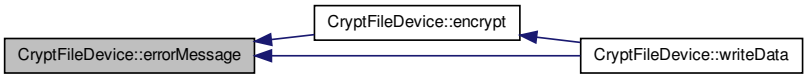
Referenced by writeData().

Here is the caller graph for this function:

9.1.3.6 void CryptFileDevice::errorMessage (const QVariant & *msg*) const [signal]

Referenced by encrypt(), and writeData().

Here is the caller graph for this function:



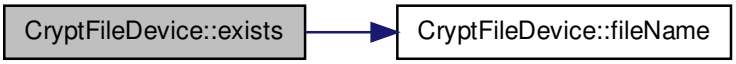
9.1.3.7 bool CryptFileDevice::exists (void) const

CryptFileDevice::exists.

Returns

References fileName(), and m_device.

Here is the call graph for this function:



9.1.3.8 QString CryptFileDevice::fileName (void) const

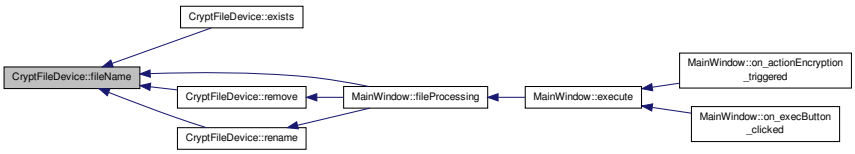
CryptFileDevice::fileName.

Returns

References m_device.

Referenced by exists(), MainWindow::fileProcessing(), remove(), and rename().

Here is the caller graph for this function:



9.1.3.9 bool CryptFileDevice::flush (void)

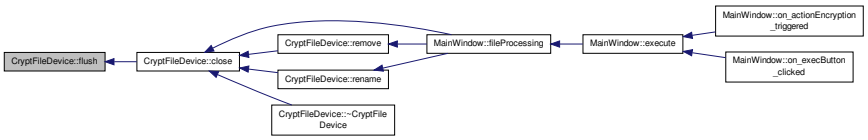
CryptFileDevice::flush.

Returns

References m_device.

Referenced by close().

Here is the caller graph for this function:



9.1.3.10 bool CryptFileDevice::initCipher (void) [private]

CryptFileDevice::initCipher.

Returns

References initCtr(), kAesKeyLength128, kAesKeyLength192, kAesKeyLength256, m_aesKey, m_aesKeyLength, m_ctrState, m_numRounds, m_password, and m_salt.

Referenced by open().

Here is the call graph for this function:

Here is the caller graph for this function:

9.1.3.11 void CryptFileDevice::initCtr (CtrState * *state*, const unsigned char * *iv*) [private]

CryptFileDevice::initCtr.

Parameters

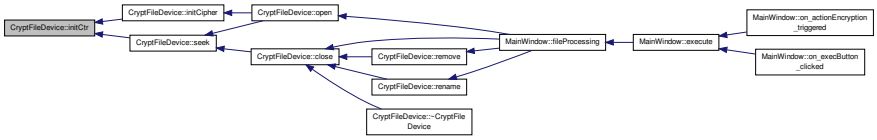
state	
iv	

References CtrState::ecount, CtrState::ivec, m_aesKey, CtrState::num, and pos().

Referenced by initCipher(), and seek().

Here is the call graph for this function:

Here is the caller graph for this function:



9.1.3.12 void CryptFileDevice::insertHeader (void) [private]

CryptFileDevice::insertHeader.

References kHeaderLength, m_aesKeyLength, m_device, m_numRounds, m_password, and m_salt.

9.1.3.13 bool CryptFileDevice::isEncrypted (void) const

CryptFileDevice::isEncrypted.

Returns

References m_encrypted.

9.1.3.14 bool CryptFileDevice::open (OpenMode *mode*)

CryptFileDevice::open.

Parameters

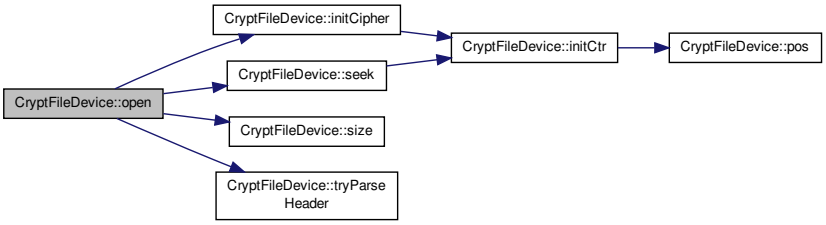
mode	

Returns

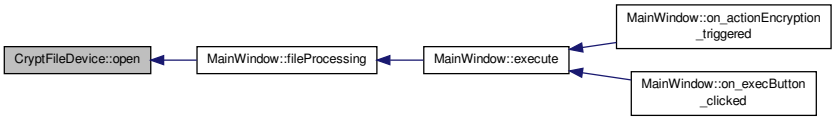
References AesCipher, initCipher(), kHeaderLength, m_device, m_encMethod, m_encrypted, m_password, seek(), size(), and tryParseHeader().

Referenced by MainWindow::fileProcessing().

Here is the call graph for this function:



Here is the caller graph for this function:



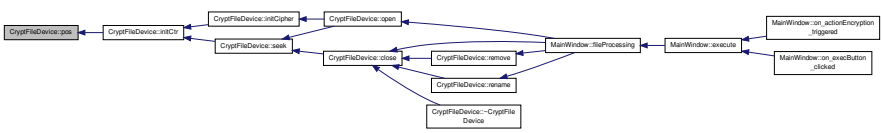
9.1.3.15 qint64 CryptFileDevice::pos (void) const

CryptFileDevice::pos.

Returns

Referenced by initCtr().

Here is the caller graph for this function:



9.1.3.16 qint64 CryptFileDevice::readBlock (qint64 *len,* QByteArray & *block*) [protected]

CryptFileDevice::readBlock.

Parameters

len	
block	

Returns

References decrypt(), and m_device.

Referenced by readData().

Here is the call graph for this function:

Here is the caller graph for this function:

9.1.3.17 qint64 CryptFileDevice::readData (char * *data*, qint64 *len*) [protected]

CryptFileDevice::readData.

Parameters

data	
len	

Returns

References m_device, m_encrypted, readBlock(), and size().

Here is the call graph for this function:

9.1.3.18 bool CryptFileDevice::remove (void)

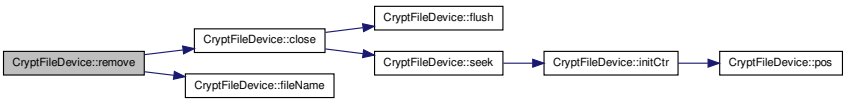
CryptFileDevice::remove.

Returns

References close(), fileName(), and m_device.

Referenced by MainWindow::fileProcessing().

Here is the call graph for this function:



Here is the caller graph for this function:

9.1.3.19 bool CryptFileDevice::rename (const QString & *newName*)

CryptFileDevice::rename.

Parameters

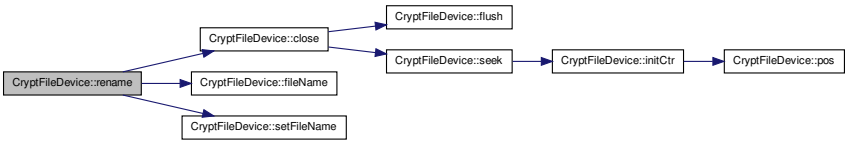
newName	

Returns

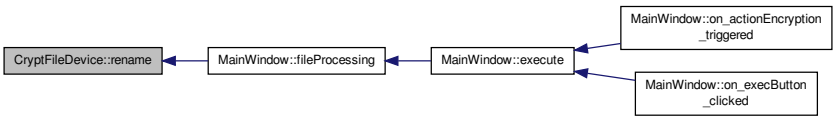
References close(), fileName(), m_device, and setFileName().

Referenced by MainWindow::fileProcessing().

Here is the call graph for this function:



Here is the caller graph for this function:



9.1.3.20 bool CryptFileDevice::seek (qint64 *pos*)

CryptFileDevice::seek.

Parameters

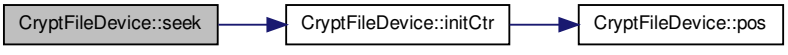
pos	

Returns

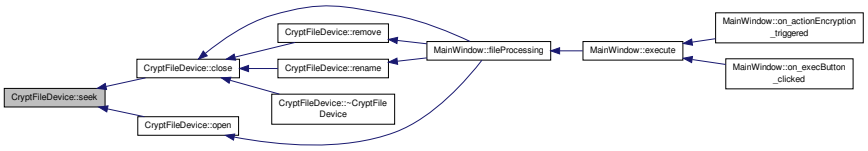
References initCtr(), CtrState::ivec, kHeaderLength, m_ctrState, m_device, and m_encrypted.

Referenced by close(), and open().

Here is the call graph for this function:



Here is the caller graph for this function:



9.1.3.21 void CryptFileDevice::setEncryptionMethod (CryptFileDevice::EncryptionMethod *enc*)

CryptFileDevice::setEncryptionMethod.

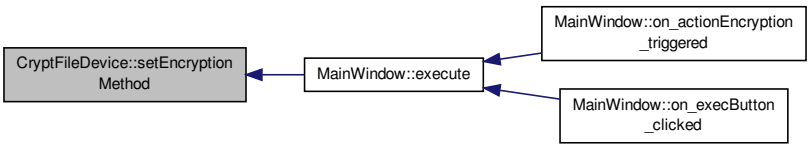
Parameters

enc	

References m_encMethod.

Referenced by MainWindow::execute().

Here is the caller graph for this function:



9.1.3.22 void CryptFileDevice::setFileDevice (QFileDevice * *device*)

CryptFileDevice::setFileDevice.

Parameters

device	

References m_device, and m_deviceOwner.

9.1.3.23 void CryptFileDevice::setFileName (const QString & *fileName*)

CryptFileDevice::setFileName.

Parameters

fileName	

References m_device, and m_deviceOwner.

Referenced by MainWindow::fileProcessing(), and rename().

Here is the caller graph for this function:

9.1.3.24 void CryptFileDevice::setKeyLength (AesKeyLength *keyLength*)

CryptFileDevice::setKeyLength.

Parameters

keyLength	

References m_aesKeyLength.

9.1.3.25 void CryptFileDevice::setNumRounds (int *numRounds*)

CryptFileDevice::setNumRounds.

Parameters

numRounds	

References m_numRounds.

9.1.3.26 void CryptFileDevice::setPassword (const QByteArray & *password*)

CryptFileDevice::setPassword.

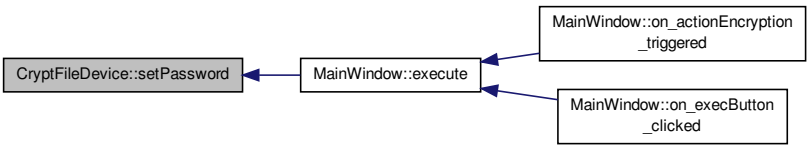
Parameters

password	

References m_password.

Referenced by MainWindow::execute().

Here is the caller graph for this function:



9.1.3.27 void CryptFileDevice::setSalt (const QByteArray & *salt*)

CryptFileDevice::setSalt.

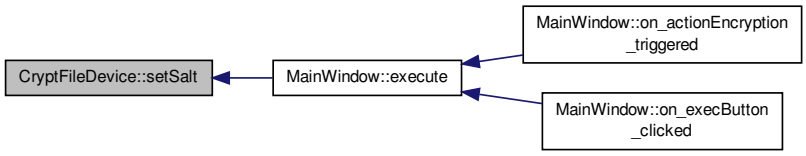
Parameters

salt	

References kSaltMaxLength, and m_salt.

Referenced by MainWindow::execute().

Here is the caller graph for this function:



9.1.3.28 qint64 CryptFileDevice::size (void) const

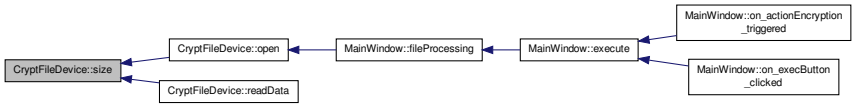
CryptFileDevice::size.

Returns

References kHeaderLength, m_device, and m_encrypted.

Referenced by open(), and readData().

Here is the caller graph for this function:



9.1.3.29 bool CryptFileDevice::tryParseHeader (void) [private]

CryptFileDevice::tryParseHeader.

Returns

References kHeaderLength, m_aesKeyLength, m_device, m_numRounds, m_password, and m_salt.

Referenced by open().

Here is the caller graph for this function:

9.1.3.30 qint64 CryptFileDevice::writeData (const char * *data*, qint64 *length*) [protected]

CryptFileDevice::writeData.

Parameters

data	
length	

Returns

References encrypt(), errorMessage(), m_device, and m_encrypted.

Here is the call graph for this function:

9.1.4 Member Data Documentation

9.1.4.1 AES_KEY CryptFileDevice::m_aesKey [private]

Referenced by decrypt(), encrypt(), initCipher(), and initCtr().

9.1.4.2 AesKeyLength CryptFileDevice::m_aesKeyLength = kAesKeyLength256 [private]

Referenced by initCipher(), insertHeader(), setKeyLength(), and tryParseHeader().

9.1.4.3 CtrState CryptFileDevice::m_ctrState [private]

Referenced by decrypt(), encrypt(), initCipher(), and seek().

9.1.4.4 QFileDevice* CryptFileDevice::m_device = nullptr [private]

Referenced by close(), exists(), fileName(), flush(), insertHeader(), open(), readBlock(), readData(), remove(), rename(), seek(), setFileDevice(), setFileName(), size(), tryParseHeader(), writeData(), and ~CryptFileDevice().

9.1.4.5 bool CryptFileDevice::m_deviceOwner = false [private]

Referenced by setFileDevice(), setFileName(), and ~CryptFileDevice().

9.1.4.6 EncryptionMethod CryptFileDevice::m_encMethod [private]

Referenced by encrypt(), open(), and setEncryptionMethod().

9.1.4.7 bool CryptFileDevice::m_encrypted = false [private]

Referenced by close(), isEncrypted(), open(), readData(), seek(), size(), and writeData().

9.1.4.8 int CryptFileDevice::m_numRounds = 5 [private]

Referenced by initCipher(), insertHeader(), setNumRounds(), and tryParseHeader().

9.1.4.9 QByteArray CryptFileDevice::m_password [private]

Referenced by encrypt(), initCipher(), insertHeader(), open(), setPassword(), and tryParseHeader().

9.1.4.10 QByteArray CryptFileDevice::m_salt [private]

Referenced by initCipher(), insertHeader(), setSalt(), and tryParseHeader().

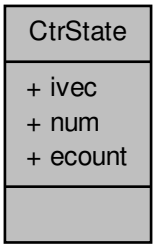
The documentation for this class was generated from the following files:

- cryptfiledevice.h
- cryptfiledevice.cpp

9.2 CtrState Struct Reference

`#include <cryptfiledevice.h>`

Collaboration diagram for CtrState:



Public Attributes

- unsigned char ivec [AES_BLOCK_SIZE]
- unsigned int num
- unsigned char ecount [AES_BLOCK_SIZE]

9.2.1 Member Data Documentation

9.2.1.1 unsigned char CtrState::ecount[AES_BLOCK_SIZE]

Referenced by CryptFileDevice::decrypt(), CryptFileDevice::encrypt(), and CryptFileDevice::initCtr().

9.2.1.2 unsigned char CtrState::ivec[AES_BLOCK_SIZE]

Referenced by CryptFileDevice::decrypt(), CryptFileDevice::encrypt(), CryptFileDevice::initCtr(), and CryptFileDevice::seek().

9.2.1.3 unsigned int CtrState::num

Referenced by CryptFileDevice::decrypt(), CryptFileDevice::encrypt(), and CryptFileDevice::initCtr().

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

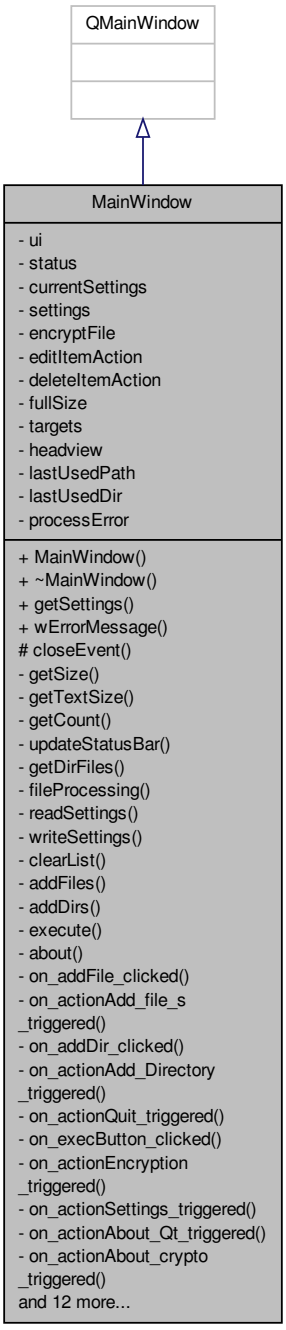
- cryptfiledevice.h

9.3 MainWindow Class Reference

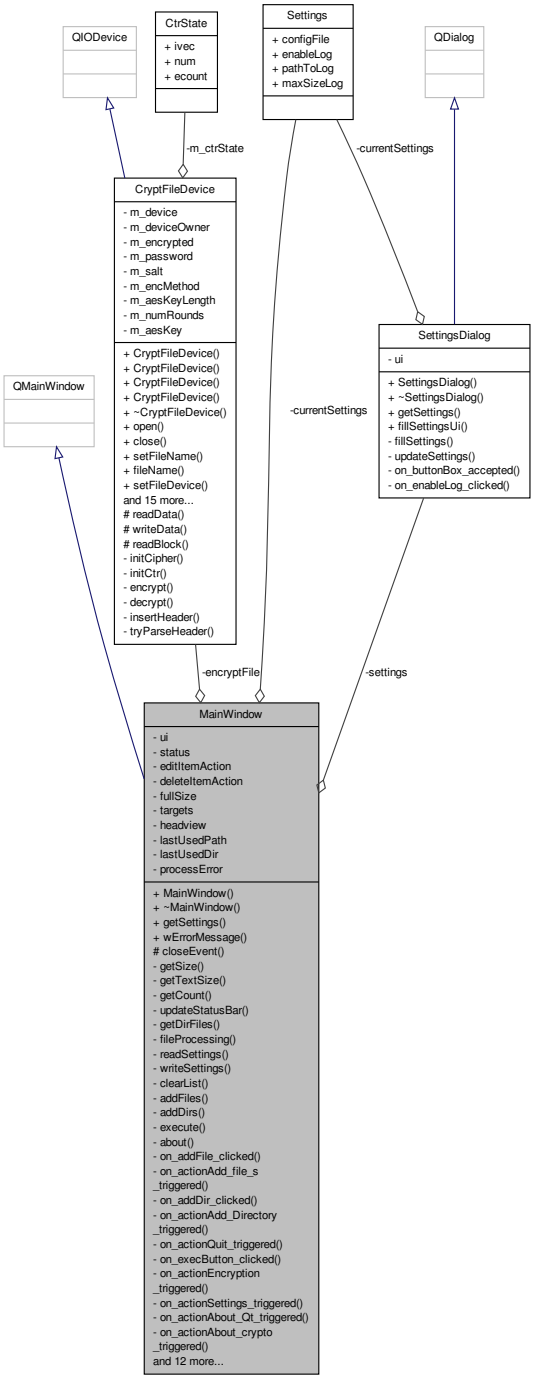
The MainWindow class is a back-end user interface.

`#include <mainwindow.h>`

Inheritance diagram for MainWindow:



Collaboration diagram for MainWindow:



Public Types

- enum ProcessStatus { PROCESS_STATUS_SUCCESS, PROCESS_STATUS_CONTINUE, PROCESS_STATUS_BREAK, PROCESS_STATUS_STATE_ERROR }

  The ProcessStatus type is used to handle errors that occur during encryption and data processing.

- enum DataType { File, Dir }

  The DataType type is used to distinguish data types in user selection dialogs.

Public Slots

- void wErrorMessage (const QVariant &message)

  Critical error message in a separate window.

Public Member Functions

- MainWindow (QWidget ∗parent=0)

  The constructor of the class MainWindow.
- ∼MainWindow ()

  The destructor of the class MainWindow.
- Settings ∗ getSettings (void) const

  get-function for the settings

Protected Member Functions

- void closeEvent (QCloseEvent ∗event) Q_DECL_OVERRIDE

  Handle program completion event.

Private Slots

- void on_addFile_clicked (void)

  Slot for adding a new file to the list.
- void on_actionAdd_file_s_triggered (void)

  Slot for adding the new files to the list.
- void on_addDir_clicked (void)

  Slot for adding a new directory to the list.
- void on_actionAdd_Directory_triggered (void)

  Slot for adding a new directory to the list.
- void on_actionQuit_triggered (void)

  Slot to exit the program.
- void on_execButton_clicked (void)

  Slot for data encryption procedure.
- void on_actionEncryption_triggered (void)

  Slot for data encryption procedure.
- void on_actionSettings_triggered (void)

  Slot for calling the user interface of the system settings.
- void on_actionAbout_Qt_triggered (void)

  A slot for issuing information about the Qt-Framework used.
- void on_actionAbout_crypto_triggered (void)

  Slot for displaying information about the program.
- void editItem (void)

  The function of editing an item from the list.
- void on_editEntry_clicked (void)

  Slot for editing an item from the list.
- void deleteItem (void)

  The function removes an item from the list.
- void on_deleteEntry_clicked (void)

  Slot to remove an item from the list.
- void on_actionContents_triggered (void)

Slot for calling assistance to the user of the program.

- void on_targetsList_currentCellChanged (int, int, int, int)

  Slot for changing the current cell from the list.

- void on_hidPassMode_clicked (bool checked)

  Slot for changing the password entry format.

- void on_passLine_textChanged (const QString &arg)

  Slot for compare between passwordLine and confirmLine.

- void on_passConfirmLine_textChanged (const QString &arg)

  Slot for compare between passwordLine and confirmLine.

- void on_clearList_clicked (void)

  Slot to clear the entire list in one click.

- void on_recurseDirs_clicked (void)

  Slot for calculate the size of the data in the directory.

- void on_actionFont_triggered (void)

  Slot for the font selection dialog.

Private Member Functions

- qint64 getSize (const QString &obj, DataType type) const

  The function solves the total size of the data selected for encryption.

- QString getTextSize (const qint64 size) const

  The function converts data size to text format (bytes/Kb/Mb/Gb)

- qint64 getCount (void) const

  The function return value is the number of data list items(files).

- void updateStatusBar (void) const

  The function displays on the status bar of the main window the number of list items and their total size.

- QStringList getDirFiles (const QString &dirPath) const

  The function reads the list of files of a given directory and all files of its subdirectories.

- ProcessStatus fileProcessing (const QString &file)

  The function encrypts / decrypts data.

- void readSettings (void)

  The function reads the parameters necessary for the user interface that were saved in the previous session.

- void writeSettings (void) const

  The function saves the user interface parameters that have been changed by the user in the current session.

- void clearList (void) const

  The function to clear the file list.

- void addFiles (void)

  The function adds the selected file to the list of items.

- void addDirs (void)

  The function adds the selected file directory to the list of items.

- void execute (void)

  The helper performs the data encryption / decryption.

- void about (void)

  Information about the program.

Private Attributes

- Ui::MainWindow * ui
- QLabel * status
- Settings * currentSettings
- SettingsDialog * settings
- CryptFileDevice * encryptFile
- QAction * editItemAction
- QAction * deleteItemAction
- qint64 fullSize
- QList< QPair< DataType, qint64 > > targets
- QHeaderView * headview
- QString lastUsedPath
- QString lastUsedDir
- bool processError

9.3.1 Detailed Description

The MainWindow class is a back-end user interface.

The MainWindow class provides the user with a number of Back-End functions that handle user events and reactions to these events.

9.3.2 Member Enumeration Documentation

9.3.2.1 enum MainWindow::DataType

The DataType type is used to distinguish data types in user selection dialogs.

Enumerator

File The data type is single file.

Dir The data type is a directory.

9.3.2.2 enum MainWindow::ProcessStatus

The ProcessStatus type is used to handle errors that occur during encryption and data processing.

Enumerator

PROCESS_STATUS_SUCCESS Execution has been successful.

PROCESS_STATUS_CONTINUE Provided file path(s) are invalid.

PROCESS_STATUS_BREAK I/O Error or not enough storage space.

PROCESS_STATUS_STATE_ERROR Bad allocation memory, etc.

9.3.3 Constructor & Destructor Documentation

9.3.3.1 MainWindow::MainWindow (QWidget * *parent* = 0) [explicit]

The constructor of the class MainWindow.

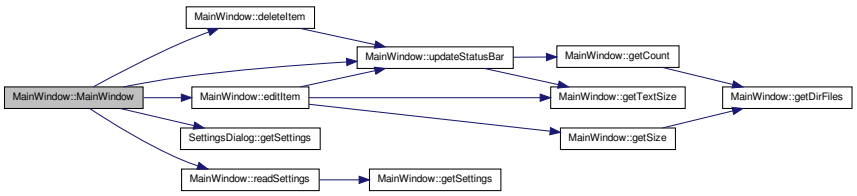
Sets default user interface parameters or uses saved values as parameters.

Parameters

parent	of the type QWidget*

References currentSettings, deleteItem(), deleteItemAction, editItem(), editItemAction, SettingsDialog::getSettings(), headview, readSettings(), settings, status, ui, and updateStatusBar().

Here is the call graph for this function:



9.3.3.2 MainWindow::~MainWindow ( )

The destructor of the class MainWindow.

References ui.

9.3.4 Member Function Documentation

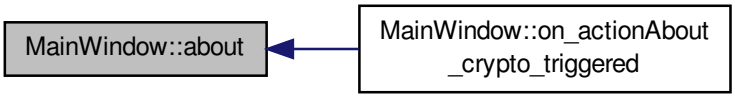
9.3.4.1 void MainWindow::about (void) [private]

Information about the program.

- Brief description of features.
- The date and release number of the program.
- Licensing restrictions and distribution of the program.
- Links to third-party libraries.

Referenced by on_actionAbout_crypto_triggered().

Here is the caller graph for this function:



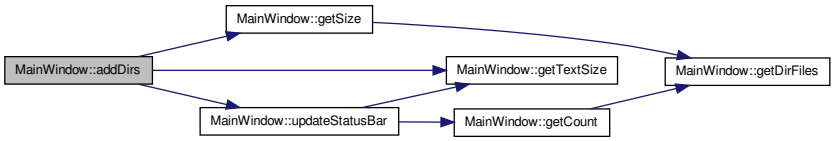
9.3.4.2 void MainWindow::addDirs (void) [private]

The function adds the selected file directory to the list of items.

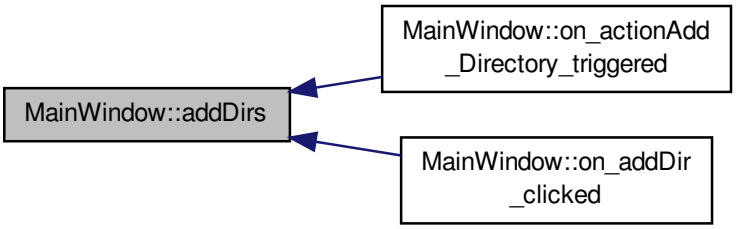
References Dir, fullSize, getSize(), getTextSize(), headview, lastUsedDir, targets, ui, and updateStatusBar().

Referenced by on_actionAdd_Directory_triggered(), and on_addDir_clicked().

Here is the call graph for this function:



Here is the caller graph for this function:



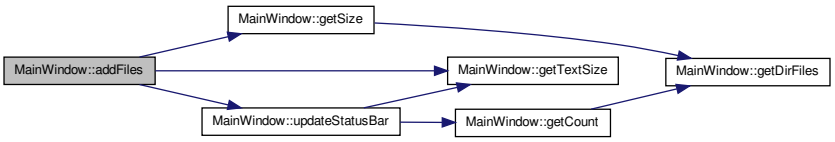
9.3.4.3 void MainWindow::addFiles (void) [private]

The function adds the selected file to the list of items.

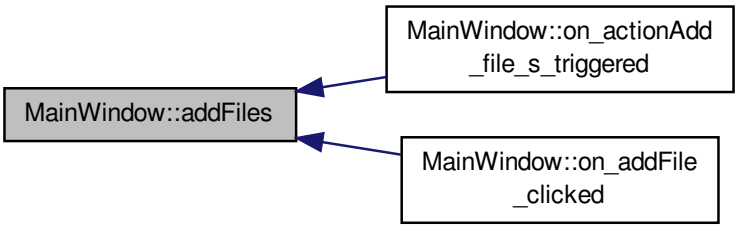
References File, fullSize, getSize(), getTextSize(), headview, lastUsedPath, targets, ui, and updateStatusBar().

Referenced by on_actionAdd_file_s_triggered(), and on_addFile_clicked().

Here is the call graph for this function:



Here is the caller graph for this function:



9.3.4.4 void MainWindow::clearList (void) const [private]

The function to clear the file list.

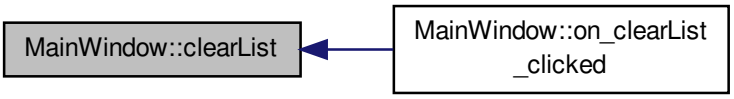
Warning

The function uses a recursive call!

References ui.

Referenced by on_clearList_clicked().

Here is the caller graph for this function:



9.3.4.5 void MainWindow::closeEvent (QCloseEvent * *event*) [protected]

Handle program completion event.

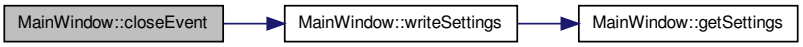
Before exiting the program, the user interface parameters are automatically saved.

Parameters

event	of type QCloseEvent*

References writeSettings().

Here is the call graph for this function:



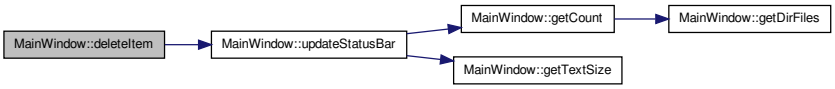
9.3.4.6 void MainWindow::deleteItem (void) [private], [slot]

The function removes an item from the list.

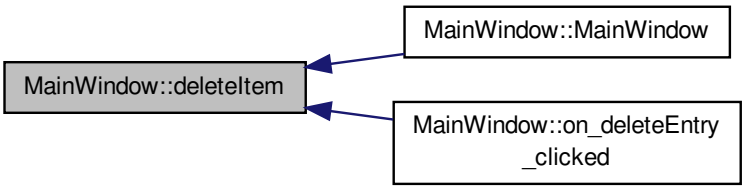
References fullSize, targets, ui, and updateStatusBar().

Referenced by MainWindow(), and on_deleteEntry_clicked().

Here is the call graph for this function:



Here is the caller graph for this function:



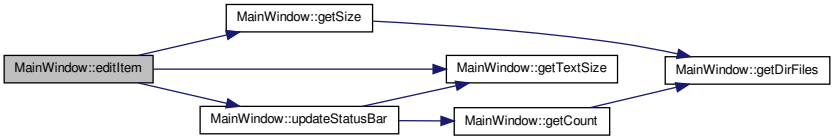
9.3.4.7 void MainWindow::editItem (void) [private], [slot]

The function of editing an item from the list.

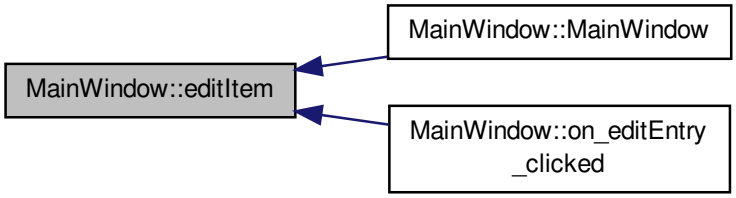
References Dir, File, fullSize, getSize(), getTextSize(), headview, lastUsedDir, lastUsedPath, targets, ui, and updateStatusBar().

Referenced by MainWindow(), and on_editEntry_clicked().

Here is the call graph for this function:



Here is the caller graph for this function:



9.3.4.8 void MainWindow::execute (void) [private]

The helper performs the data encryption / decryption.

Note

In the body of this function, a password is getting for encryption and an additional salt to the password is set.

Warning

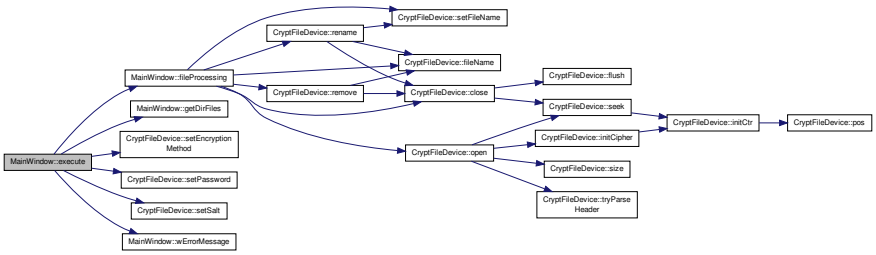
Password salt is taken from the release time of the program, taken in microseconds. Therefore, different editions of the program will not be fully compatible with each other! Encoded data from one release will not be decrypted by another release of the program, even if the password is known.

Todo Password salt is taken from the release time of the program, taken in microseconds.

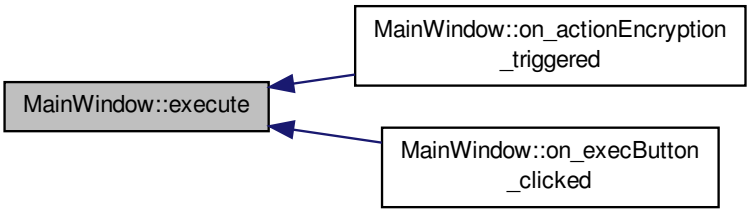
References CryptFileDevice::AesCipher, COEFF, Dir, encryptFile, File, fileProcessing(), fullSize, getDirFiles(), ONEKB, PROCESS_STATUS_BREAK, PROCESS_STATUS_CONTINUE, PROCESS_STATUS_STATE_ERROR, PROCESS_STATUS_SUCCESS, processError, CryptFileDevice::setEncryptionMethod(), CryptFileDevice::setPassword(), CryptFileDevice::setSalt(), targets, ui, wErrorMessage(), and CryptFileDevice::XorCipher.

Referenced by on_actionEncryption_triggered(), and on_execButton_clicked().

Here is the call graph for this function:



Here is the caller graph for this function:



9.3.4.9 MainWindow::ProcessStatus MainWindow::fileProcessing (const QString & *f*) [private]

The function encrypts / decrypts data.

Parameters

f	of the type QString&, path to the file.

Returns

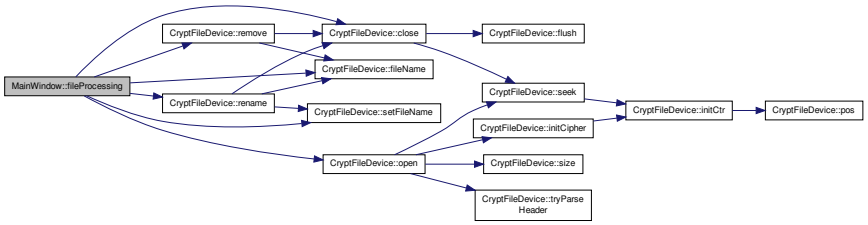
status of the coding/encoding process.

Todo Make an extension for encrypted files! (".enc")

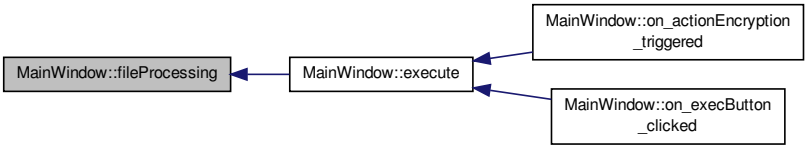
References CryptFileDevice::close(), COEFF, encryptFile, CryptFileDevice::fileName(), CryptFileDevice::open(), PROCESS_STATUS_BREAK, PROCESS_STATUS_CONTINUE, PROCESS_STATUS_SUCCESS, processError, CryptFileDevice::remove(), CryptFileDevice::rename(), CryptFileDevice::setFileName(), and ui.

Referenced by execute().

Here is the call graph for this function:



Here is the caller graph for this function:



9.3.4.10 qint64 MainWindow::getCount (void) const [private]

The function return value is the number of data list items(files).

Returns

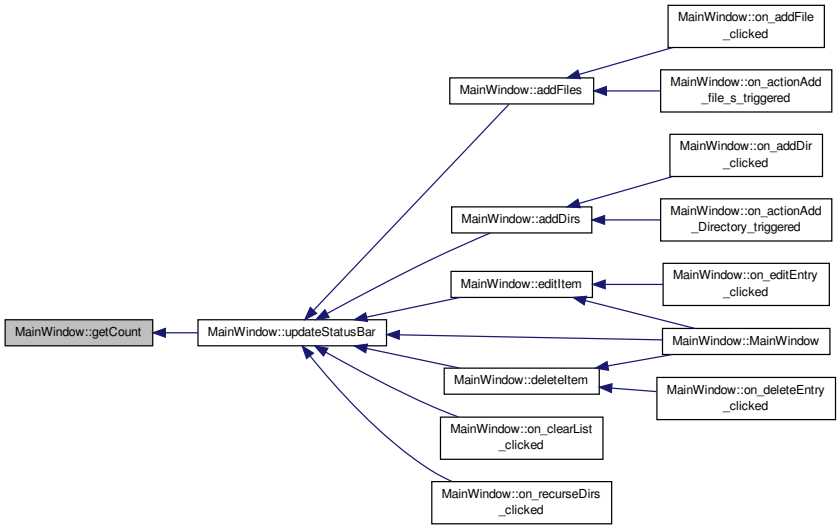
number of data list files

References Dir, File, getDirFiles(), targets, and ui.

Referenced by updateStatusBar().

Here is the call graph for this function:

Here is the caller graph for this function:



9.3.4.11 QStringList MainWindow::getDirFiles (const QString & *dirPath*) const [private]

The function reads the list of files of a given directory and all files of its subdirectories.

Parameters

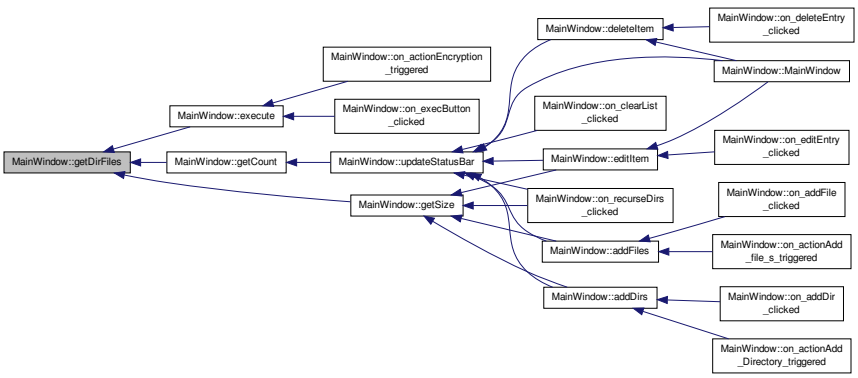
dirPath	of the type QString&

Returns

fileNames of the type QStringList a list of the files

Referenced by execute(), getCount(), and getSize().

Here is the caller graph for this function:



9.3.4.12 Settings * MainWindow::getSettings (void) const

get-function for the settings

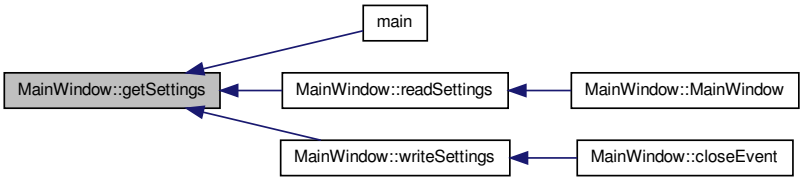
Returns

currentSettings of the type Settings*

References currentSettings.

Referenced by main(), readSettings(), and writeSettings().

Here is the caller graph for this function:



9.3.4.13 qint64 MainWindow::getSize (const QString & *obj*, DataType *type*) const [private]

The function solves the total size of the data selected for encryption.

Parameters

obj	of the type QString&, path to the data
type	of the type enum DataType {File, Dir}

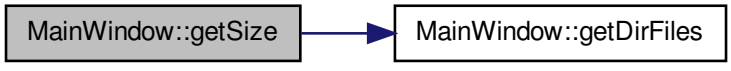
Returns

size of the type qint64, the size of the data.

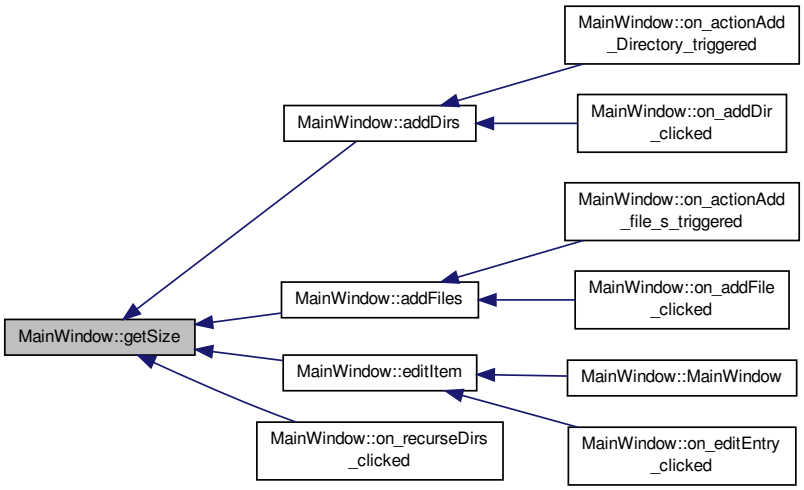
References Dir, File, getDirFiles(), and ui.

Referenced by addDirs(), addFiles(), editItem(), and on_recurseDirs_clicked().

Here is the call graph for this function:



Here is the caller graph for this function:



9.3.4.14 QString MainWindow::getTextSize (const qint64 *size*) const [private]

The function converts data size to text format (bytes/Kb/Mb/Gb)

Parameters

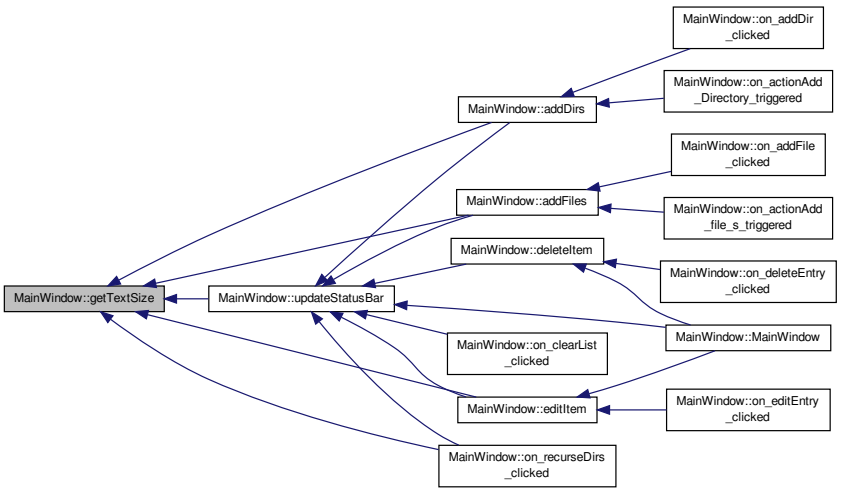
size	of the type qint64, the size of the data

Returns

string, as the text.

Referenced by addDirs(), addFiles(), editItem(), on_recurseDirs_clicked(), and updateStatusBar().

Here is the caller graph for this function:

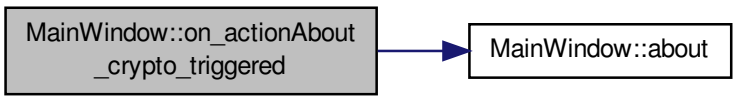


9.3.4.15 void MainWindow::on_actionAbout_crypto_triggered (void) [private], [slot]

Slot for displaying information about the program.

References about().

Here is the call graph for this function:



9.3.4.16 void MainWindow::on_actionAbout_Qt_triggered (void) [private], [slot]

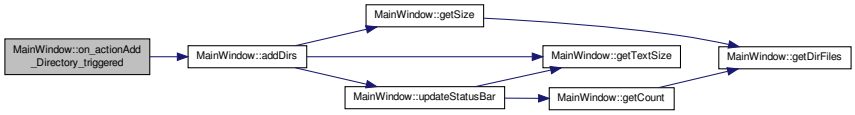
A slot for issuing information about the Qt-Framework used.

9.3.4.17 void MainWindow::on_actionAdd_Directory_triggered (void) [private], [slot]

Slot for adding a new directory to the list.

References addDirs().

Here is the call graph for this function:

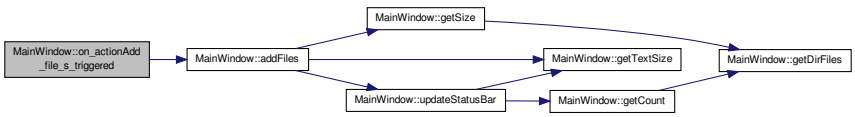


9.3.4.18 void MainWindow::on_actionAdd_file_s_triggered (void) [private], [slot]

Slot for adding the new files to the list.

References addFiles().

Here is the call graph for this function:



9.3.4.19 void MainWindow::on_actionContents_triggered (void) [private], [slot]

Slot for calling assistance to the user of the program.

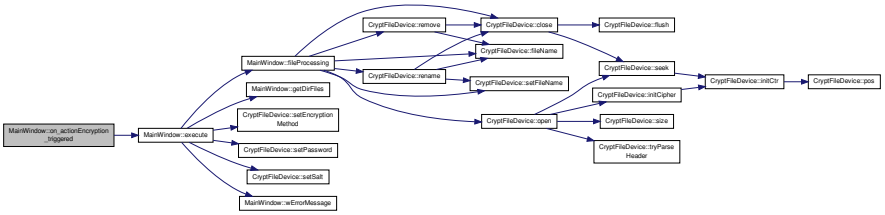
Todo Add help for the program!

9.3.4.20 void MainWindow::on_actionEncryption_triggered (void) [private], [slot]

Slot for data encryption procedure.

References execute().

Here is the call graph for this function:



9.3.4.21 void MainWindow::on_actionFont_triggered (void) [private], [slot]

Slot for the font selection dialog.

Selection of the screen font.

Warning

The font is installed immediately for the entire program, i.e. for all graphic forms!

9.3.4.22 void MainWindow::on_actionQuit_triggered (void) [private], [slot]

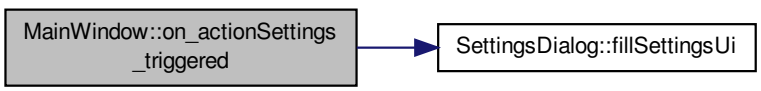
Slot to exit the program.

9.3.4.23 void MainWindow::on_actionSettings_triggered (void) [private], [slot]

Slot for calling the user interface of the system settings.

References SettingsDialog::fillSettingsUi(), and settings.

Here is the call graph for this function:

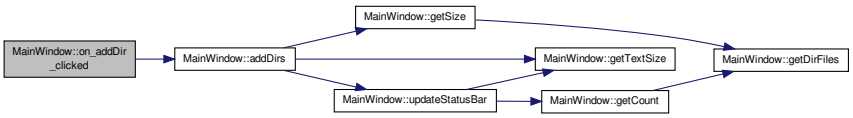


9.3.4.24 void MainWindow::on_addDir_clicked (void) [private],[slot]

Slot for adding a new directory to the list.

References addDirs().

Here is the call graph for this function:

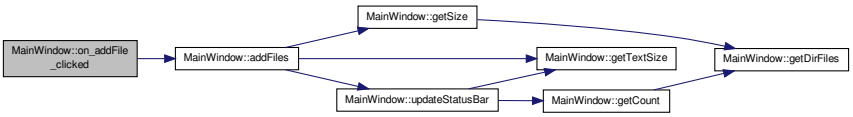


9.3.4.25 void MainWindow::on_addFile_clicked (void) [private],[slot]

Slot for adding a new file to the list.

References addFiles().

Here is the call graph for this function:

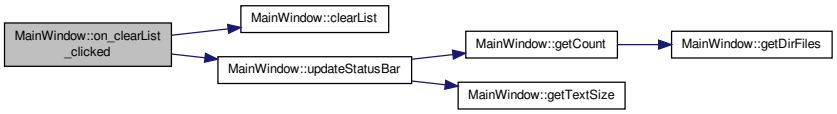


9.3.4.26 void MainWindow::on_clearList_clicked (void) [private],[slot]

Slot to clear the entire list in one click.

References clearList(), fullSize, targets, ui, and updateStatusBar().

Here is the call graph for this function:



9.3.4.27 void MainWindow::on_deleteEntry_clicked (void) [private],[slot]

Slot to remove an item from the list.

References deleteItem().

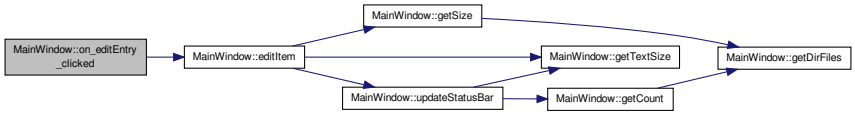
Here is the call graph for this function:

9.3.4.28 void MainWindow::on_editEntry_clicked (void) [private], [slot]

Slot for editing an item from the list.

References editItem().

Here is the call graph for this function:

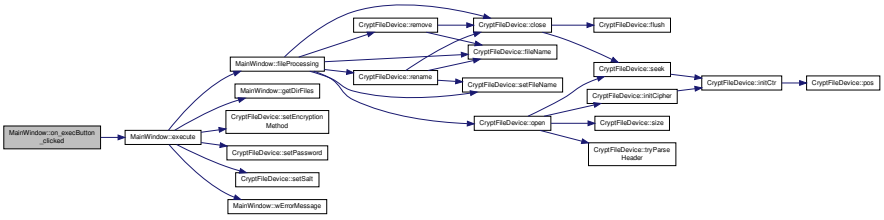


9.3.4.29 void MainWindow::on_execButton_clicked (void) [private], [slot]

Slot for data encryption procedure.

References execute().

Here is the call graph for this function:



9.3.4.30 void MainWindow::on_hidPassMode_clicked (bool *checked*) [private], [slot]

Slot for changing the password entry format.

Slot switch to hidden mode for the Password.

Parameters

checked	of type bool

References ui.

9.3.4.31 void MainWindow::on_passConfirmLine_textChanged (const QString & *arg*) [private], [slot]

Slot for compare between passwordLine and confirmLine.

Parameters

arg	of type QString&

References ui.

9.3.4.32 void MainWindow::on_passLine_textChanged (const QString & *arg*) [private],[slot]

Slot for compare between passwordLine and confirmLine.

Parameters

arg	of type QString&

References ui.

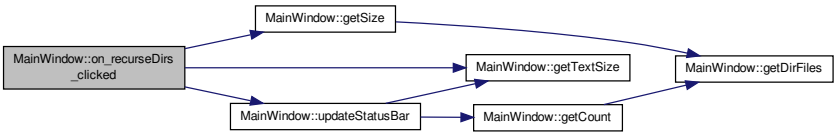
9.3.4.33 void MainWindow::on_recurseDirs_clicked (void) [private],[slot]

Slot for calculate the size of the data in the directory.

Process all subdirectories recursively.

References Dir, fullSize, getSize(), getTextSize(), headview, targets, ui, and updateStatusBar().

Here is the call graph for this function:



9.3.4.34 void MainWindow::on_targetsList_currentCellChanged (int , int , int , int) [private],[slot]

Slot for changing the current cell from the list.

References deleteItemAction, editItemAction, and ui.

9.3.4.35 void MainWindow::readSettings (void) [private]

The function reads the parameters necessary for the user interface that were saved in the previous session.

Such important parameters will be read as

- the position of the window on the screen and window size,
- interface font and its size,
- the user interface settings (overwrite of the data, recurse of dir's, etc.)
- error and event logging options.

References Settings::enableLog, getSettings(), lastUsedDir, lastUsedPath, Settings::maxSizeLog, Settings::pathToLog, settings, and ui.

Referenced by MainWindow().

Here is the call graph for this function:

Here is the caller graph for this function:

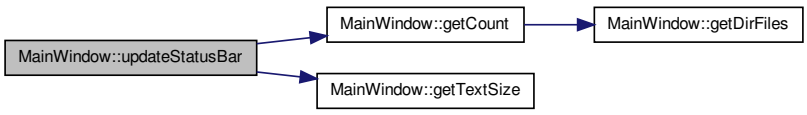
9.3.4.36 void MainWindow::updateStatusBar (void) const [private]

The function displays on the status bar of the main window the number of list items and their total size.

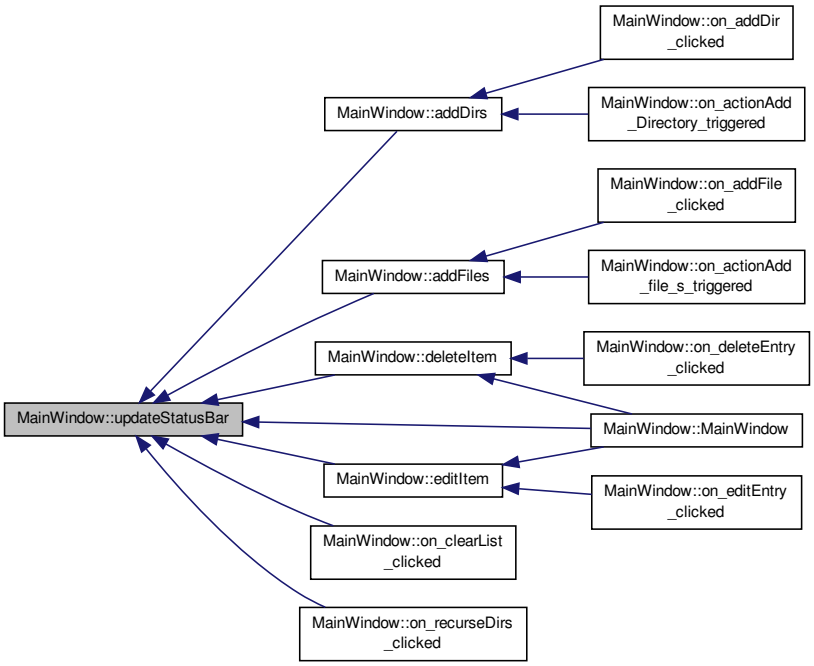
References fullSize, getCount(), getTextSize(), and status.

Referenced by addDirs(), addFiles(), deleteItem(), editItem(), MainWindow(), on_clearList_clicked(), and on_recurseDirs_clicked().

Here is the call graph for this function:



Here is the caller graph for this function:



9.3.4.37 void MainWindow::wErrorMessage (const QVariant & *message*) [slot]

Critical error message in a separate window.

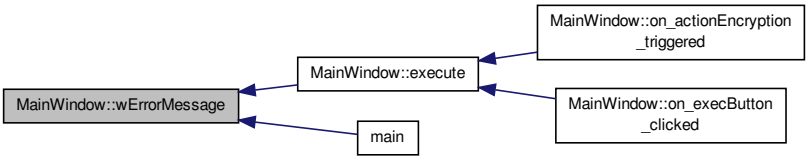
Parameters

message	of the type QString, error message.

References processError.

Referenced by execute(), and main().

Here is the caller graph for this function:



9.3.4.38 void MainWindow::writeSettings (void) const [private]

The function saves the user interface parameters that have been changed by the user in the current session.

Such parameters will be updated as

- the position of the window on the screen and window size,
- interface font and its size,
- the user interface settings (overwrite of the data, recurse of dir's, etc.)
- error and event logging options.

References getSettings(), lastUsedDir, lastUsedPath, settings, and ui.

Referenced by closeEvent().

Here is the call graph for this function:

Here is the caller graph for this function:

9.3.5 Member Data Documentation

9.3.5.1 Settings* MainWindow::currentSettings [private]

Referenced by getSettings(), and MainWindow().

9.3.5.2 QAction* MainWindow::deleteItemAction [private]

Referenced by MainWindow(), and on_targetsList_currentCellChanged().

9.3.5.3 QAction* MainWindow::editItemAction [private]

Referenced by MainWindow(), and on_targetsList_currentCellChanged().

9.3.5.4 CryptFileDevice* MainWindow::encryptFile [private]

Referenced by execute(), and fileProcessing().

9.3.5.5 qint64 MainWindow::fullSize [private]

Referenced by addDirs(), addFiles(), deleteItem(), editItem(), execute(), on_clearList_clicked(), on_recurseDirs_clicked(), and updateStatusBar().

9.3.5.6 QHeaderView* MainWindow::headview [private]

Referenced by addDirs(), addFiles(), editItem(), MainWindow(), and on_recurseDirs_clicked().

9.3.5.7 QString MainWindow::lastUsedDir [private]

Referenced by addDirs(), editItem(), readSettings(), and writeSettings().

9.3.5.8 QString MainWindow::lastUsedPath [private]

Referenced by addFiles(), editItem(), readSettings(), and writeSettings().

9.3.5.9 bool MainWindow::processError [private]

Referenced by execute(), fileProcessing(), and wErrorMessage().

9.3.5.10 SettingsDialog* MainWindow::settings [private]

Referenced by MainWindow(), on_actionSettings_triggered(), readSettings(), and writeSettings().

9.3.5.11 QLabel* MainWindow::status [private]

Referenced by MainWindow(), and updateStatusBar().

9.3.5.12 QList<QPair<DataType, qint64> > MainWindow::targets [private]

Referenced by addDirs(), addFiles(), deleteItem(), editItem(), execute(), getCount(), on_clearList_clicked(), and on_recurseDirs_clicked().

9.3.5.13 Ui::MainWindow* MainWindow::ui [private]

Referenced by addDirs(), addFiles(), clearList(), deleteItem(), editItem(), execute(), fileProcessing(), getCount(), getSize(), MainWindow(), on_clearList_clicked(), on_hidPassMode_clicked(), on_passConfirmLine_textChanged(), on_passLine_textChanged(), on_recurseDirs_clicked(), on_targetsList_currentCellChanged(), readSettings(), writeSettings(), and ~MainWindow().

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

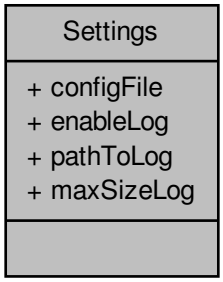
- mainwindow.h
- mainwindow.cpp

9.4 Settings Struct Reference

The Settings structure.

`#include <settings.h>`

Collaboration diagram for Settings:



Public Attributes

- QString configFile

  Path to the configuration file.
- bool enableLog

  Enables / disables the logging procedure.
- QString pathToLog

  Path to the log/debug file.
- quint32 maxSizeLog

  This field determines the maximum size of the log file (in Kb)

9.4.1 Detailed Description

The Settings structure.

The structure contains specific fields for storing logging/debugging parameters. Motivation. This structure is necessary for storing user settings of the debugging system. It differs from the main system of user settings in that it allows you to save settings in a separate file. This is required for compatibility with other platforms.

9.4.2 Member Data Documentation

9.4.2.1 QString Settings::configFile

Path to the configuration file.

9.4.2.2 bool Settings::enableLog

Enables / disables the logging procedure.

Referenced by SettingsDialog::fillSettingsUi(), main(), SettingsDialog::on_buttonBox_accepted(), MainWindow::readSettings(), and SettingsDialog::updateSettings().

9.4.2.3 quint32 Settings::maxSizeLog

This field determines the maximum size of the log file (in Kb)

Referenced by SettingsDialog::fillSettingsUi(), main(), MainWindow::readSettings(), and SettingsDialog::updateSettings().

9.4.2.4 QString Settings::pathToLog

Path to the log/debug file.

Referenced by SettingsDialog::fillSettingsUi(), main(), MainWindow::readSettings(), and SettingsDialog::updateSettings().

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

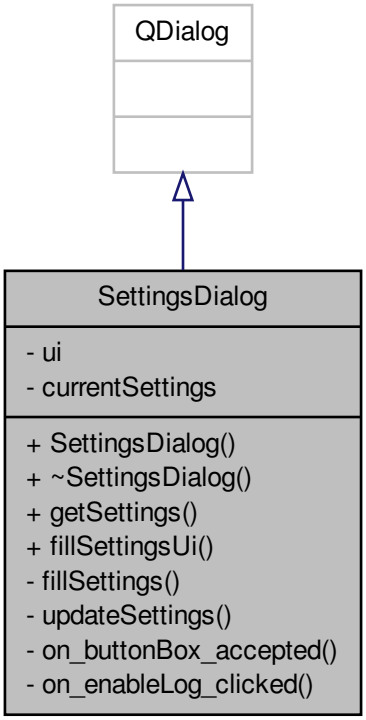
- settings.h

9.5 SettingsDialog Class Reference

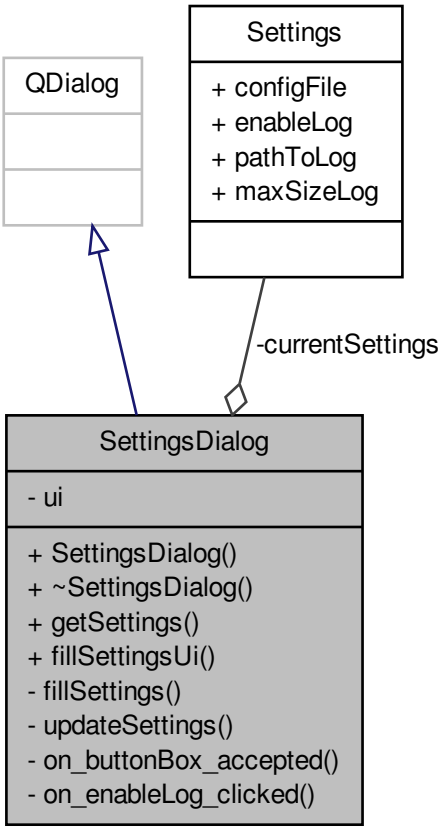
The SettingsDialog class.

`#include <settingsdialog.h>`

Inheritance diagram for SettingsDialog:



Collaboration diagram for SettingsDialog:



Public Member Functions

- SettingsDialog (QWidget *parent=0)

  The constructor of the class SettingsDialog.
- ~SettingsDialog ()

  The destructor of the class SettingsDialog.
- Settings * getSettings (void)

  get-function for the settings
- void fillSettingsUi (void)

  The fillSettingsUi function.

Private Slots

- void on_buttonBox_accepted (void)

  Slot on_buttonBox_accepted for confirmation and acceptance of changes in settings.
- void on_enableLog_clicked (bool checked)

  The on_enableLog_clicked slot for controlling user interface parameters.

Private Member Functions

- void fillSettings (void)

  The fillSettings function.
- void updateSettings (void)

  The updateSettings function.

Private Attributes

- Ui::SettingsDialog * ui
- Settings currentSettings

9.5.1 Detailed Description

The SettingsDialog class.

The SettingsDialog class provides the user with a number of Back-End functions that handle user events and reactions to these events.

9.5.2 Constructor & Destructor Documentation

9.5.2.1 SettingsDialog::SettingsDialog (QWidget * *parent* = 0) [explicit]

The constructor of the class SettingsDialog.

Sets default user interface parameters or uses saved values as parameters.

Parameters

parent	of the type QWidget*

9.5.2.2 SettingsDialog::~SettingsDialog ( )

The destructor of the class SettingsDialog.

References ui.

9.5.3 Member Function Documentation

9.5.3.1 void SettingsDialog::fillSettings (void) [private]

The fillSettings function.

Note

Filling data from the configuration file. Alternative approach. Now it is disabled.

Todo The method of checking and reading from the configuration file.

9.5.3.2 void SettingsDialog::fillSettingsUi (void)

The fillSettingsUi function.

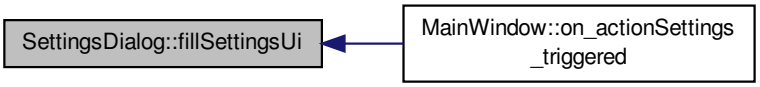
The function fills the Ui parameters that it takes from the Settings.

Todo Validation method for configuration data.

References currentSettings, Settings::enableLog, Settings::maxSizeLog, Settings::pathToLog, and ui.

Referenced by MainWindow::on_actionSettings_triggered().

Here is the caller graph for this function:



9.5.3.3 Settings * SettingsDialog::getSettings (void)

get-function for the settings

Returns

currentSettings of the type Settings

References currentSettings.

Referenced by MainWindow::MainWindow().

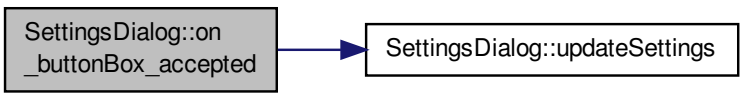
Here is the caller graph for this function:

9.5.3.4 void SettingsDialog::on_buttonBox_accepted (void) [private], [slot]

Slot on_buttonBox_accepted for confirmation and acceptance of changes in settings.

References currentSettings, Settings::enableLog, ui, and updateSettings().

Here is the call graph for this function:



9.5.3.5 void SettingsDialog::on_enableLog_clicked (bool *checked*) [private], [slot]

The on_enableLog_clicked slot for controlling user interface parameters.

The other setting parameters are then hidden.

Parameters

checked	of type bool. Checks whether logging is enabled.

References ui.

9.5.3.6 void SettingsDialog::updateSettings (void) [private]

The updateSettings function.

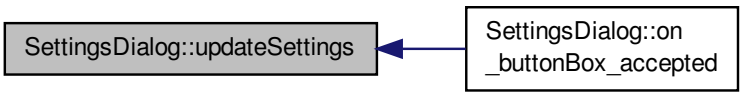
This function updates the settings from the user interface in the Settings.

Todo Alternative method to set default data!

References currentSettings, Settings::enableLog, Settings::maxSizeLog, Settings::pathToLog, and ui.

Referenced by on_buttonBox_accepted().

Here is the caller graph for this function:



9.5.4 Member Data Documentation

9.5.4.1 Settings SettingsDialog::currentSettings [private]

Referenced by fillSettingsUi(), getSettings(), on_buttonBox_accepted(), and updateSettings().

9.5.4.2 Ui::SettingsDialog* SettingsDialog::ui [private]

Referenced by fillSettingsUi(), on_buttonBox_accepted(), on_enableLog_clicked(), updateSettings(), and ~SettingsDialog().

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

- settingsdialog.h
- settingsdialog.cpp

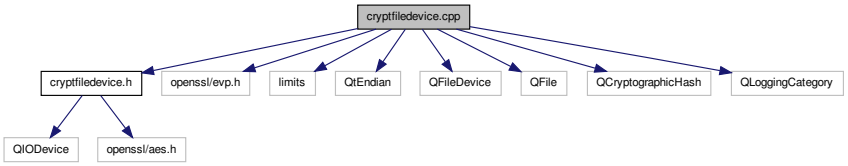
Chapter 10

File Documentation

10.1 cryptfiledevice.cpp File Reference

```
#include "cryptfiledevice.h"
#include <openssl/evp.h>
#include <limits>
#include <QtEndian>
#include <QFileDevice>
#include <QFile>
#include <QCryptographicHash>
#include <QLoggingCategory>
```

Include dependency graph for cryptfiledevice.cpp:



Variables

- static int const kHeaderLength = 128
- static int const kSaltMaxLength = 8

10.1.1 Variable Documentation

10.1.1.1 int const kHeaderLength = 128 [static]

Referenced by CryptFileDevice::insertHeader(), CryptFileDevice::open(), CryptFileDevice::seek(), CryptFileDevice::size(), and CryptFileDevice::tryParseHeader().

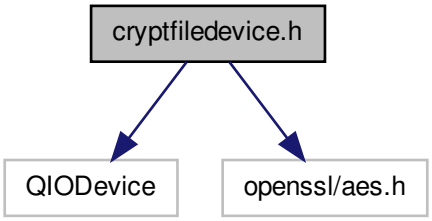
10.1.1.2 int const kSaltMaxLength = 8 [static]

Referenced by CryptFileDevice::setSalt().

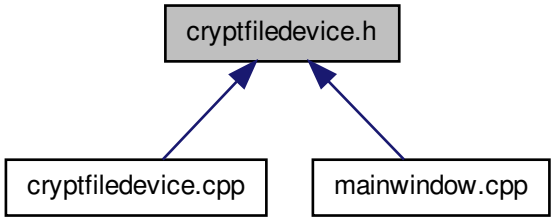
10.2 cryptfiledevice.h File Reference

```
#include <QIODevice>
#include <openssl/aes.h>
```

Include dependency graph for cryptfiledevice.h:



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



Classes

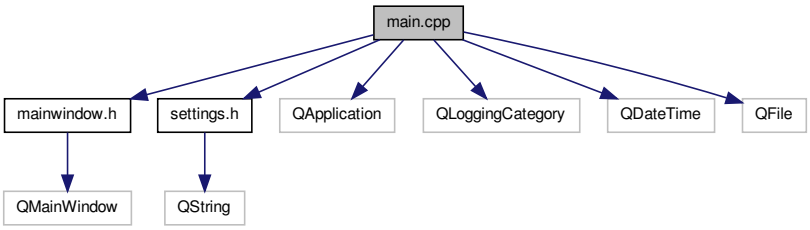
- struct CtrState
- class CryptFileDevice

10.3 main.cpp File Reference

The file contains two important functions, main() and logMessageOutput()

```
#include "mainwindow.h"
#include "settings.h"
#include <QApplication>
#include <QLoggingCategory>
#include <QDateTime>
#include <QFile>
```

Include dependency graph for main.cpp:



Macros

- #define ONEKB 1024

Functions

- void logMessageOutput (const QtMsgType type, const QMessageLogContext &context, const QString &msg)
  The function logMessageOutput is a message handler.
- int main (int argc, char ∗argv[])
  main function

Variables

- static QScopedPointer< QFile > m_logFile

10.3.1 Detailed Description

The file contains two important functions, main() and logMessageOutput() The main function executes an instance of a GUI Qt application, sets it up with the specified special parameters, and installs a Qt message handler defined in the logMessageOutput function. In addition, a log journal of the application messages is set up.

10.3.2 Macro Definition Documentation

10.3.2.1 #define ONEKB 1024

Referenced by main().

10.3.3 Function Documentation

10.3.3.1 void logMessageOutput (const QtMsgType *type*, const QMessageLogContext & *context*, const QString & *msg*)

The function logMessageOutput is a message handler.

This function redirects the messages by their category (QtDebugMsg, QtInfoMsg, QtWarningMsg, QtCriticalMsg, QtFatalMsg) to the log file (m_logFile). The message handler is a function that prints out debug messages, warnings, critical and fatal error messages. The Qt library (debug mode) contains hundreds of warning messages that are printed when internal errors (usually invalid function arguments) occur. Qt built in release mode also contains such warnings unless QT_NO_WARNING_OUTPUT and/ or QT_NO_DEBUG_OUTPUT have been set during compilation. If you implement your own message handler, you get total control of these messages.

Parameters

in	*type*	of the type QtMsgType
in	*context*	of type QMessageLogContext
in	*msg*	of the type QString

Note

- The output of messages is also output to the terminal console. This is for debugging purposes.
- Additional information, such as a line of code, the name of the source file, function names cannot be displayed for the release of the program.

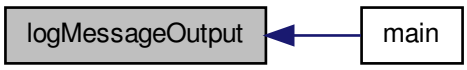
Warning

- The default message handler prints the message to the standard output under X11 or to the debugger under Windows. If it is a fatal message, the application aborts immediately.
- Only one message handler can be defined, since this is usually done on an application-wide basis to control debug output.

References m_logFile.

Referenced by main().

Here is the caller graph for this function:



10.3.3.2 int main (int *argc*, char * *argv[]*)

main function

In this function, an instance of a GUI Qt application app is executed and set up with the parameters entered.

Parameters

argc	this parameter is ignored because it is a GUI application.
argv	this parameter is ignored because it is a GUI application.

Returns

value of the function QApplication::exec() Enters the main event loop and waits until exit() is called. Returns the value that was set to exit() (which is 0 if exit() is called via quit()).

Note

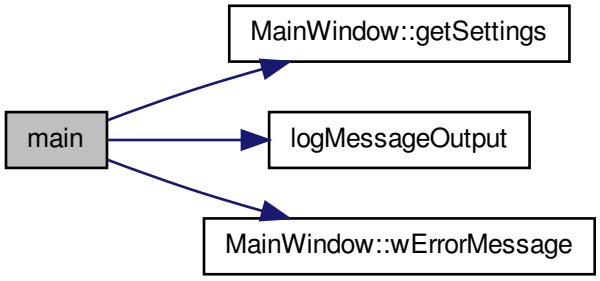
The program parameters (argc, argv) are ignored.

Warning

none

References Settings::enableLog, MainWindow::getSettings(), logMessageOutput(), Settings::maxSizeLog, ONEKB, Settings::pathToLog, and MainWindow::wErrorMessage().

Here is the call graph for this function:



10.3.4 Variable Documentation

10.3.4.1 QScopedPointer<QFile> m_logFile [static]

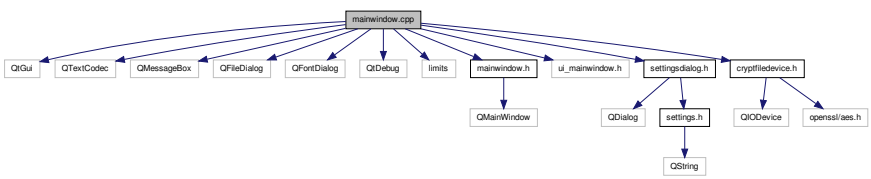
Referenced by logMessageOutput().

10.4 mainwindow.cpp File Reference

This file contains the definition of methods and interfaces of the MainWindow class.

```
#include <QtGui>
#include <QTextCodec>
#include <QMessageBox>
#include <QFileDialog>
#include <QFontDialog>
#include <QtDebug>
#include <limits>
#include "mainwindow.h"
#include "ui_mainwindow.h"
#include "settingsdialog.h"
#include "cryptfiledevice.h"
```

Include dependency graph for mainwindow.cpp:



Macros

- #define COEFF 1048576
- #define ONEKB 1024

10.4.1 Detailed Description

This file contains the definition of methods and interfaces of the MainWindow class.

10.4.2 Macro Definition Documentation

10.4.2.1 #define COEFF 1048576

Referenced by MainWindow::execute(), and MainWindow::fileProcessing().

10.4.2.2 #define ONEKB 1024

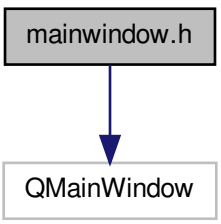
Referenced by MainWindow::execute().

10.5 mainwindow.h File Reference

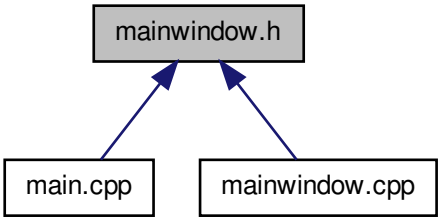
This file contains the declaration of the class MainWindow.

```
#include <QMainWindow>
```

Include dependency graph for mainwindow.h:



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



Classes

- class MainWindow

  The MainWindow class is a back-end user interface.

Namespaces

- Ui

10.5.1 Detailed Description

This file contains the declaration of the class MainWindow.

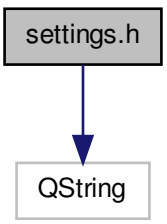
10.6 README.md File Reference

10.7 settings.h File Reference

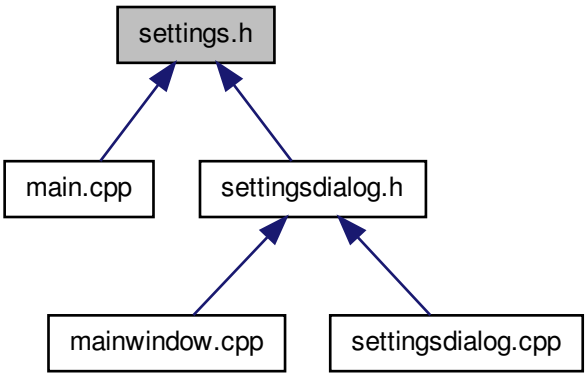
This file contains the declaration of the structure Settings.

```
#include <QString>
```

Include dependency graph for settings.h:



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



Classes

- struct Settings

  The Settings structure.

10.7.1 Detailed Description

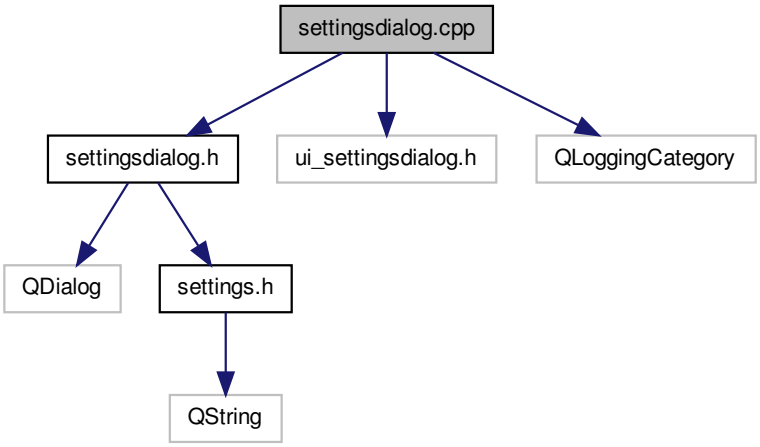
This file contains the declaration of the structure Settings.

10.8 settingsdialog.cpp File Reference

This file contains the definition of methods and interfaces of the SettingsDialog class.

```
#include "settingsdialog.h"
#include "ui_settingsdialog.h"
#include <QLoggingCategory>
```

Include dependency graph for settingsdialog.cpp:



10.8.1 Detailed Description

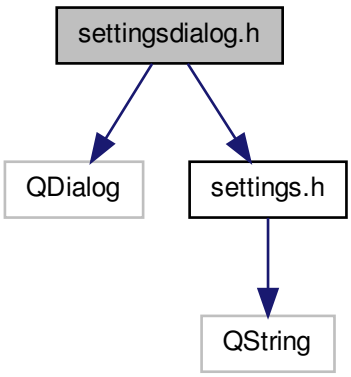
This file contains the definition of methods and interfaces of the SettingsDialog class.

10.9 settingsdialog.h File Reference

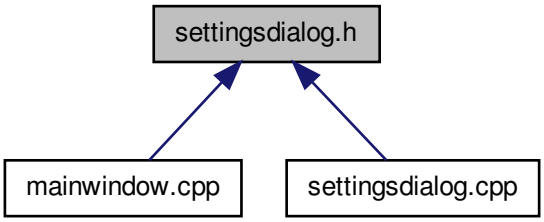
This file contains the declaration of the class SettingsDialog.

```
#include <QDialog>
#include "settings.h"
```

Include dependency graph for settingsdialog.h:



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



Classes

- class SettingsDialog

  The SettingsDialog class.

Namespaces

- Ui

10.9.1 Detailed Description

This file contains the declaration of the class SettingsDialog.

Index