helix_c99_demo

0.2

Generated by Doxygen 1.8.5

Fri Oct 2 2020 15:33:42

Contents

Chapter 1

Command-Line Encrypt/Decrypt utility using Helix Library

Synopsys

Allows to encrypt/decrypt a given file via Helix. Need to supply path to input, and desired output path (if none, cwd).

```
./helix-util [-e/-d] <input_file> <output_file> <password>
```

where

- -e(ncrypt), -d(ecrypt), optional (do none by default)
- input_file is the path to the input file, required
- output_file is the path to the output file, optional (do cwd with same name as default)

in any case, if applicable, encrypted and decrypted files will have the appropriate type at the end of the file name. i.e `my_text.txt` becomes `my_text.txt-encrypted` and `my_text.txt-decrypted`.

Chapter 2

File Index

2.1 File List

Here is a list of all files with brief descriptions:

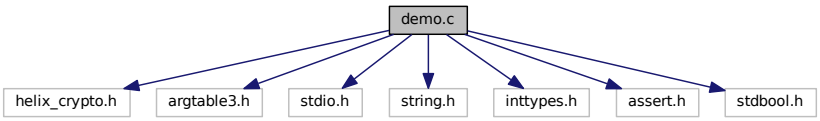
Chapter 3

File Documentation

3.1 demo.c File Reference

```
#include "helix_crypto.h"
#include "argtable3.h"
#include <stdio.h>
#include <string.h>
#include <inttypes.h>
#include <assert.h>
#include <stdbool.h>
```

Include dependency graph for demo.c:



Macros

- #define ERROR_ARGPARSE_INVALID 18
- #define ERROR_HELIX_ACCOUNT 17
- #define ERROR_HELIX_ACCOUNT_CREATE 10
- #define ERROR_HELIX_ACCOUNT_LOGIN 11
- #define ERROR_HELIX_DECRYPT_EMPTY 15
- #define ERROR_HELIX_DECRYPT_SIZE 16
- #define ERROR_HELIX_DECRYPT_STATUS 14
- #define ERROR_HELIX_ENCRYPT_EMPTY 13
- #define ERROR_HELIX_ENCRYPT_RECIPIENT 12
- #define ERROR_HELIX_MODULE 8
- #define ERROR_HELIX_SERVER 9
- #define ERROR_INPUT_MALLOC 5
- #define ERROR_INPUT_NAME 2
- #define ERROR_INPUT_READ 3
- #define ERROR_INPUT_READSIZE 4
- #define ERROR_NONE 0
- #define ERROR_OUTPUT_NAME 6

- #define ERROR_OUTPUT_WRITE 7
- #define ERROR_SYNTAX 1
- #define MAX_FILEPATH_LENGTH 2048

Functions

- bool accountCreate (const char *account)

  Internal helper method to handle account creation.
- bool accountDelete (const char *account)

  Internal helper method to handle account deletion.
- bool accountLogin (const char *account)

  Internal helper method to handle account login.
- int authenticateWithHelixNetwork (const char *account)

  Perform authentication (of existing) or creation (of new) account in Helix Network.
- invokeStatus_t connectToHelixKeyServer (void)

  Connect to Helix key-server (that was specified at Helix initialization time).
- uint8_t * decryptFromBytes (uint8_t *blob, size_t len, const char *password, size_t *outBytes)

  Given some encrypted content, decrypt it.
- void disconnectFromHelixKeyServer (void)

  Disconnect from Helix key-server. This is a blocking call - its return signals orderly discontinuity of all network activities.
- uint8_t * encryptFromBytes (const char *recipientAccount, uint8_t *content, size_t len, const char *password, size_t *outBytes)

  Given some plain content, encrypt it for a given target user.
- void loadHelixModule (const char *, uint16_t, const char *, const char *)
- int main (int argc, char **argv)

  The main function of the demo.
- uint8_t * readBytesFromFile (const char *path, size_t *bytesRead)

  Reads bytes from a given file.
- void unloadHelixModule (void)

  Unload Helix Module. This call disables all Helix module actitivies and delete its runtime state from memory.
- void writeBytesToFile (const char *path, const uint8_t *content, size_t count)

  Writes bytes to a file.

Variables

- struct arg_lit * dec = NULL
- const char DEFAULT_KEY_SERVER [128] = "service.blakfx.us"
- const uint16_t DEFAULT_KEY_SERVER_PORT = 5567
- struct arg_lit * enc = NULL
- struct arg_end * end = NULL
- struct arg_lit * help = NULL
- struct arg_str * in = NULL
- struct arg_str * key_server = NULL
- struct arg_int * key_server_port = NULL
- struct arg_str * out = NULL
- struct arg_str * pass = NULL
- struct arg_str * simulated_id = NULL
- struct arg_str * user = NULL

3.1.1 Macro Definition Documentation

3.1.1.1 #define ERROR_ARGPARSE_INVALID 18

3.1.1.2 #define ERROR_HELIX_ACCOUNT 17

3.1.1.3 #define ERROR_HELIX_ACCOUNT_CREATE 10

3.1.1.4 #define ERROR_HELIX_ACCOUNT_LOGIN 11

3.1.1.5 #define ERROR_HELIX_DECRYPT_EMPTY 15

3.1.1.6 #define ERROR_HELIX_DECRYPT_SIZE 16

3.1.1.7 #define ERROR_HELIX_DECRYPT_STATUS 14

3.1.1.8 #define ERROR_HELIX_ENCRYPT_EMPTY 13

3.1.1.9 #define ERROR_HELIX_ENCRYPT_RECIPIENT 12

3.1.1.10 #define ERROR_HELIX_MODULE 8

3.1.1.11 #define ERROR_HELIX_SERVER 9

3.1.1.12 #define ERROR_INPUT_MALLOC 5

3.1.1.13 #define ERROR_INPUT_NAME 2

3.1.1.14 #define ERROR_INPUT_READ 3

3.1.1.15 #define ERROR_INPUT_READSIZE 4

3.1.1.16 #define ERROR_NONE 0

3.1.1.17 #define ERROR_OUTPUT_NAME 6

3.1.1.18 #define ERROR_OUTPUT_WRITE 7

3.1.1.19 #define ERROR_SYNTAX 1

3.1.1.20 #define MAX_FILEPATH_LENGTH 2048

3.1.2 Function Documentation

3.1.2.1 bool accountCreate (const char * *account*)

Internal helper method to handle account creation.

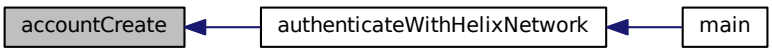
Parameters

in	*account*	name of the account to create

Returns

whether creation succeeded or not

Here is the caller graph for this function:



3.1.2.2 bool accountDelete (const char * *account*)

Internal helper method to handle account deletion.

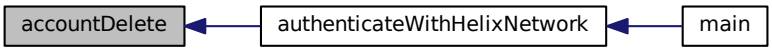
Parameters

in	*account*	name of the account to delete

Returns

whether account deletion succeeded or not

Here is the caller graph for this function:



3.1.2.3 bool accountLogin (const char * *account*)

Internal helper method to handle account login.

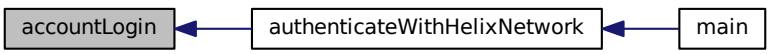
Parameters

in	*account*	name of the account to log into

Returns

whether account login succeeded or not

Here is the caller graph for this function:



3.1.2.4 int authenticateWithHelixNetwork (const char * *account*)

Perform authentication (of existing) or creation (of new) account in Helix Network.

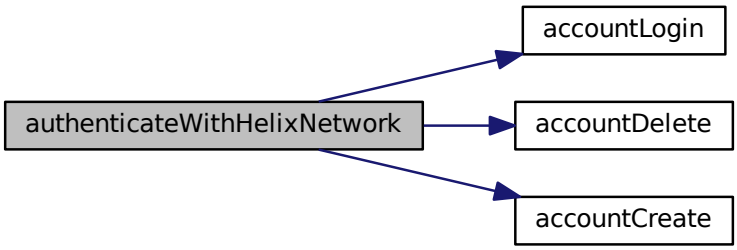
Parameters

in	*account*	the account name to identify as

Returns

result of user authentication attempt

Here is the call graph for this function:



Here is the caller graph for this function:

3.1.2.5 invokeStatus_t connectToHelixKeyServer (void)

Connect to Helix key-server (that was specified at Helix initialization time).

Returns

result status of the connection attempt

Here is the caller graph for this function:

3.1.2.6 uint8_t * decryptFromBytes (uint8_t * *blob*, size_t *len*, const char * *password*, size_t * *outBytes*)

Given some encrypted content, decrypt it.

Parameters

in	*blob*	the encrypted content to decrypt
in	*len*	the size of the content to decrypt
in	*password*	the password to use when decrypting the content
out	*outBytes*	the number of bytes of the decryption result

Returns

the decrypted bytes result

Here is the caller graph for this function:

3.1.2.7 void disconnectFromHelixKeyServer (void)

Disconnect from Helix key-server. This is a blocking call - its return signals orderly discontinuity of all network activities.

Here is the caller graph for this function:

3.1.2.8 uint8_t * encryptFromBytes (const char * *recipientAccount*, uint8_t * *content*, size_t *len*, const char * *password*, size_t * *outBytes*)

Given some plain content, encrypt it for a given target user.

Parameters

in	*recipientAccount*	the name of the target to encrypt this message for
in	*content*	the content to encrypt
in	*len*	the size of the content to encrypt
in	*password*	the password to encrypt the content with
out	*outBytes*	the number of bytes of the encryption result

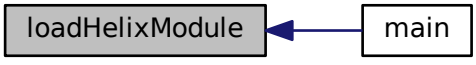
Returns

the encrypted bytes result

Here is the caller graph for this function:

3.1.2.9 void loadHelixModule (const char * *server_ip*, uint16_t *server_port*, const char * *account*, const char * *device*)

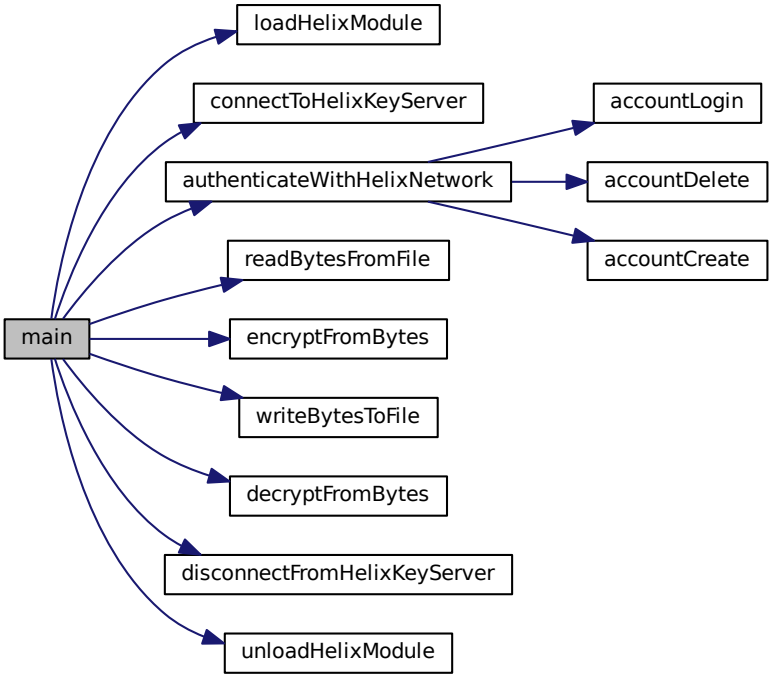
Here is the caller graph for this function:



3.1.2.10 int main (int *argc*, char ** *argv*)

The main function of the demo.

Here is the call graph for this function:



3.1.2.11 uint8_t * readBytesFromFile (const char * *path*, size_t * *bytesRead*)

Reads bytes from a given file.

Parameters

in	*path*	the path of the file to read
out	*bytesRead*	the number of bytes read

Returns

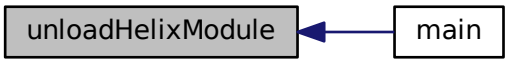
the bytes read

Here is the caller graph for this function:

3.1.2.12 void unloadHelixModule (void)

Unload Helix Module. This call disables all Helix module actitivies and delete its runtime state from memory.

Here is the caller graph for this function:



3.1.2.13 void writeBytesToFile (const char * *path*, const uint8_t * *content*, size_t *count*)

Writes bytes to a file.

Parameters

in	*path*	the path of the file to write to
in	*content*	the bytes to write
in	*count*	the number of bytes to write

Here is the caller graph for this function:

3.1.3 Variable Documentation

3.1.3.1 struct arg_lit * dec = NULL

3.1.3.2 const char DEFAULT_KEY_SERVER[128] = "service.blakfx.us"

3.1.3.3 const uint16_t DEFAULT_KEY_SERVER_PORT = 5567

3.1.3.4 struct arg_lit * enc = NULL

3.1.3.5 struct arg_end* end = NULL

3.1.3.6 struct arg_lit* help = NULL

3.1.3.7 struct arg_str* in = NULL

3.1.3.8 struct arg_str * key_server = NULL

3.1.3.9 struct arg_int* key_server_port = NULL

3.1.3.10 struct arg_str * out = NULL

3.1.3.11 struct arg_str * pass = NULL

3.1.3.12 struct arg_str * simulated_id = NULL

3.1.3.13 struct arg_str * user = NULL

3.2 mainpage.md File Reference

Index