

helix_c99_demo
0.2

Generated by Doxygen 1.8.5

Fri Oct 2 2020 15:33:42

Contents

1	Command-Line Encrypt/Decrypt utility using Helix Library	1
2	File Index	3
2.1	File List	3
3	File Documentation	5
3.1	demo.c File Reference	5
3.1.1	Macro Definition Documentation	7
3.1.1.1	ERROR_ARGPARSE_INVALID	7
3.1.1.2	ERROR_HELIX_ACCOUNT	7
3.1.1.3	ERROR_HELIX_ACCOUNT_CREATE	7
3.1.1.4	ERROR_HELIX_ACCOUNT_LOGIN	7
3.1.1.5	ERROR_HELIX_DECRYPT_EMPTY	7
3.1.1.6	ERROR_HELIX_DECRYPT_SIZE	7
3.1.1.7	ERROR_HELIX_DECRYPT_STATUS	7
3.1.1.8	ERROR_HELIX_ENCRYPT_EMPTY	7
3.1.1.9	ERROR_HELIX_ENCRYPT_RECIPIENT	7
3.1.1.10	ERROR_HELIX_MODULE	7
3.1.1.11	ERROR_HELIX_SERVER	7
3.1.1.12	ERROR_INPUT_MALLOC	7
3.1.1.13	ERROR_INPUT_NAME	7
3.1.1.14	ERROR_INPUT_READ	7
3.1.1.15	ERROR_INPUT_READSIZE	7
3.1.1.16	ERROR_NONE	7
3.1.1.17	ERROR_OUTPUT_NAME	7
3.1.1.18	ERROR_OUTPUT_WRITE	7
3.1.1.19	ERROR_SYNTAX	7
3.1.1.20	MAX_FILEPATH_LENGTH	7
3.1.2	Function Documentation	7
3.1.2.1	accountCreate	7
3.1.2.2	accountDelete	8
3.1.2.3	accountLogin	8

3.1.2.4	authenticateWithHelixNetwork	9
3.1.2.5	connectToHelixKeyServer	9
3.1.2.6	decryptFromBytes	10
3.1.2.7	disconnectFromHelixKeyServer	10
3.1.2.8	encryptFromBytes	11
3.1.2.9	loadHelixModule	11
3.1.2.10	main	12
3.1.2.11	readBytesFromFile	12
3.1.2.12	unloadHelixModule	13
3.1.2.13	writeBytesToFile	13
3.1.3	Variable Documentation	13
3.1.3.1	dec	13
3.1.3.2	DEFAULT_KEY_SERVER	13
3.1.3.3	DEFAULT_KEY_SERVER_PORT	13
3.1.3.4	enc	13
3.1.3.5	end	13
3.1.3.6	help	13
3.1.3.7	in	13
3.1.3.8	key_server	13
3.1.3.9	key_server_port	14
3.1.3.10	out	14
3.1.3.11	pass	14
3.1.3.12	simulated_id	14
3.1.3.13	user	14
3.2	mainpage.md File Reference	14

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-decryptd.

Chapter 2

File Index

2.1 File List

Here is a list of all files with brief descriptions:

demo.c	5
----------------------------------	---

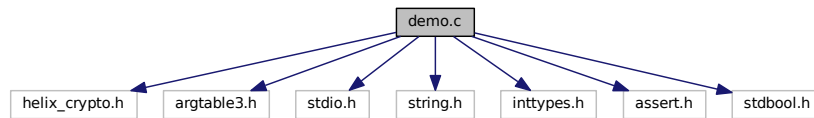
Chapter 3

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 activities 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.

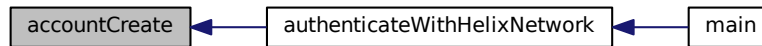
Parameters

<code>in</code>	<code><i>account</i></code>	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.

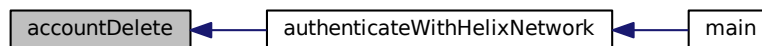
Parameters

in	<i>account</i>	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.

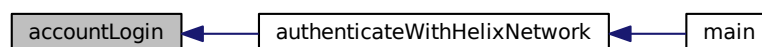
Parameters

in	<i>account</i>	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.

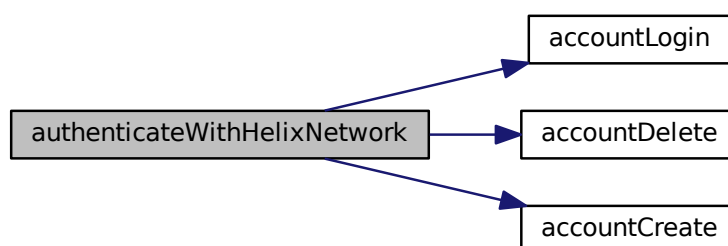
Parameters

in	<i>account</i>	the account name to identify as
----	----------------	---------------------------------

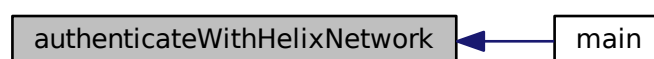
Returns

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.

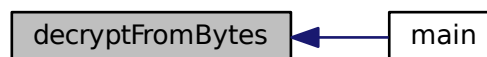
Parameters

in	<i>blob</i>	the encrypted content to decrypt
in	<i>len</i>	the size of the content to decrypt
in	<i>password</i>	the password to use when decrypting the content
out	<i>outBytes</i>	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	<i>recipientAccount</i>	the name of the target to encrypt this message for
in	<i>content</i>	the content to encrypt
in	<i>len</i>	the size of the content to encrypt
in	<i>password</i>	the password to encrypt the content with
out	<i>outBytes</i>	the number of bytes of the encryption result

Returns

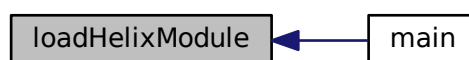
the encrypted bytes result

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)`

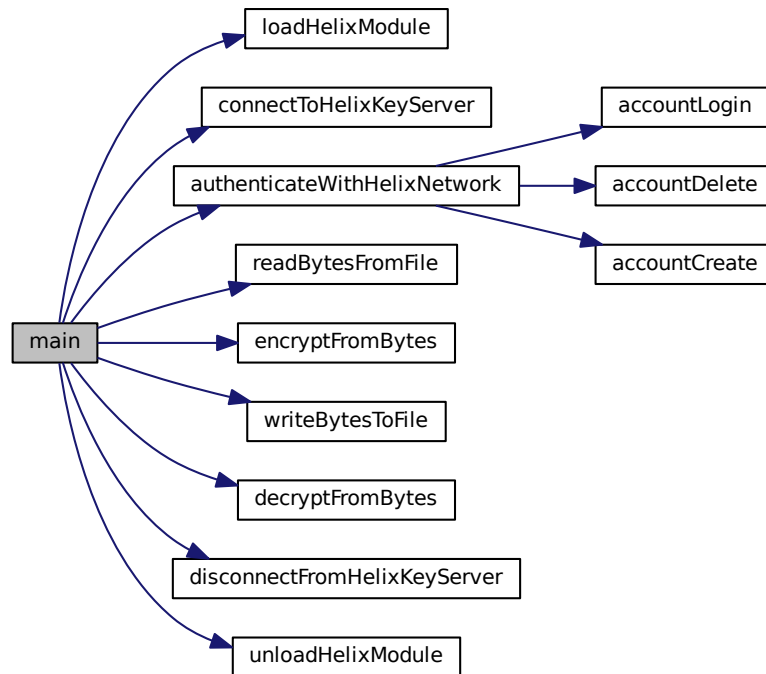
Here is the caller graph for this function:



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	<i>path</i>	the path of the file to read
out	<i>bytesRead</i>	the number of bytes read

Returns

the bytes read

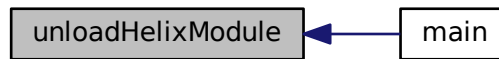
Here is the caller graph for this function:



3.1.2.12 void unloadHelixModule (void)

Unload Helix Module. This call disables all Helix module activities 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	<i>path</i>	the path of the file to write to
in	<i>content</i>	the bytes to write
in	<i>count</i>	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

accountCreate
demo.c, 7

accountDelete
demo.c, 8

accountLogin
demo.c, 8

authenticateWithHelixNetwork
demo.c, 8

connectToHelixKeyServer
demo.c, 9

DEFAULT_KEY_SERVER
demo.c, 13

dec
demo.c, 13

decryptFromBytes
demo.c, 10

demo.c, 5

- accountCreate, 7
- accountDelete, 8
- accountLogin, 8
- authenticateWithHelixNetwork, 8
- connectToHelixKeyServer, 9
- DEFAULT_KEY_SERVER, 13
- dec, 13
- decryptFromBytes, 10
- disconnectFromHelixKeyServer, 10
- ERROR_HELIX_ACCOUNT, 7
- ERROR_HELIX_MODULE, 7
- ERROR_HELIX_SERVER, 7
- ERROR_INPUT_MALLOC, 7
- ERROR_INPUT_NAME, 7
- ERROR_INPUT_READ, 7
- ERROR_NONE, 7
- ERROR_OUTPUT_NAME, 7
- ERROR_OUTPUT_WRITE, 7
- ERROR_SYNTAX, 7
- enc, 13
- encryptFromBytes, 11
- end, 13
- help, 13
- in, 13
- key_server, 13
- key_server_port, 13
- loadHelixModule, 11
- MAX_FILEPATH_LENGTH, 7
- main, 11
- out, 14
- pass, 14
- readBytesFromFile, 12
- simulated_id, 14
- unloadHelixModule, 13
- user, 14
- writeBytesToFile, 13

disconnectFromHelixKeyServer
demo.c, 10

ERROR_HELIX_ACCOUNT
demo.c, 7

ERROR_HELIX_MODULE
demo.c, 7

ERROR_HELIX_SERVER
demo.c, 7

ERROR_INPUT_MALLOC
demo.c, 7

ERROR_INPUT_NAME
demo.c, 7

ERROR_INPUT_READ
demo.c, 7

ERROR_INPUT_READSIZE
demo.c, 7

ERROR_NONE
demo.c, 7

ERROR_OUTPUT_NAME
demo.c, 7

ERROR_OUTPUT_WRITE
demo.c, 7

ERROR_SYNTAX
demo.c, 7

enc
demo.c, 13

encryptFromBytes
demo.c, 11

end
demo.c, 13

help
demo.c, 13

in
demo.c, 13

key_server
demo.c, 13

key_server_port
demo.c, 13

loadHelixModule
demo.c, 11

MAX_FILEPATH_LENGTH

demo.c, [7](#)

main

demo.c, [11](#)

mainpage.md, [14](#)

out

demo.c, [14](#)

pass

demo.c, [14](#)

readBytesFromFile

demo.c, [12](#)

simulated_id

demo.c, [14](#)

unloadHelixModule

demo.c, [13](#)

user

demo.c, [14](#)

writeBytesToFile

demo.c, [13](#)