

Yubico Universal 2nd Factor C Library

COLLABORATORS

	<i>TITLE :</i> Yubico Universal 2nd Factor C Library		
<i>ACTION</i>	<i>NAME</i>	<i>DATE</i>	<i>SIGNATURE</i>
WRITTEN BY		January 26, 2015	

REVISION HISTORY

NUMBER	DATE	DESCRIPTION	NAME

Contents

1	Yubico Universal 2nd Factor C Library	1
1.1	u2f-server	1
1.2	u2f-server-version	10
2	Index	13

Chapter 1

Yubico Universal 2nd Factor C Library

This is a C library that implements the server-side of the U2F protocol. More precisely, it provides an API for generating the JSON blobs required by U2F devices to perform the U2F Registration and U2F Authentication operations, and functionality for verifying the cryptographic operations.

1.1 u2f-server

u2f-server —

Functions

u2fs_rc	u2fs_global_init ()
void	u2fs_global_done ()
const char *	u2fs_strerror ()
const char *	u2fs_strerror_name ()
u2fs_rc	u2fs_init ()
void	u2fs_done ()
u2fs_rc	u2fs_set_origin ()
u2fs_rc	u2fs_set_appid ()
u2fs_rc	u2fs_set_challenge ()
u2fs_rc	u2fs_set_keyHandle ()
u2fs_rc	u2fs_set_publicKey ()
u2fs_rc	u2fs_registration_challenge ()
u2fs_rc	u2fs_registration_verify ()
const char *	u2fs_get_registration_keyHandle ()
const char *	u2fs_get_registration_publicKey ()
void	u2fs_free_reg_res ()
u2fs_rc	u2fs_authentication_challenge ()
u2fs_rc	u2fs_authentication_verify ()
u2fs_rc	u2fs_get_authentication_result ()
void	u2fs_free_auth_res ()

Types and Values

#define	U2FS_CHALLENGE_RAW_LEN
#define	U2FS_CHALLENGE_B64U_LEN
#define	U2FS_PUBLIC_KEY_LEN

#define	U2FS_COUNTER_LEN
enum	u2fs_rc
enum	u2fs_initflags
typedef	u2fs_ctx_t
typedef	u2fs_reg_res_t
typedef	u2fs_auth_res_t

Description

Functions

u2fs_global_init ()

```
u2fs_rc
u2fs_global_init (u2fs_initflags flags);
```

Initialize the library. This function is not guaranteed to be thread safe and must be invoked on application startup.

Parameters

flags	initialization flags, ORed u2fs_initflags.	
-------	---	--

Returns

On success **U2FS_OK** (integer 0) is returned, and on errors an **u2fs_rc** error code.

u2fs_global_done ()

```
void
u2fs_global_done (void);
```

Release all resources from the library. Call this function when no further use of the library is needed.

u2fs_strerror ()

```
const char~*
u2fs_strerror (int err);
```

Convert return code to human readable string explanation of the reason for the particular error code.

This string can be used to output a diagnostic message to the user.

This function is one of few in the library that can be used without a successful call to **u2fs_global_init()**.

Parameters

err	error code	
-----	------------	--

Returns

Returns a pointer to a statically allocated string containing an explanation of the error code *err*.

u2fs_strerror_name ()

```
const char~*
u2fs_strerror_name (int err);
```

Convert return code to human readable string representing the error code symbol itself. For example, `u2fs_strerror_name(U2FS_OK)` returns the string "U2FS_OK".

This string can be used to output a diagnostic message to the user.

This function is one of few in the library that can be used without a successful call to `u2fs_global_init()`.

Parameters

err	error code	
-----	------------	--

Returns

Returns a pointer to a statically allocated string containing a string version of the error code `err`, or NULL if the error code is not known.

u2fs_init ()

```
u2fs_rc
u2fs_init (u2fs_ctx_t **ctx);
```

Initialize the U2F server context handle.

Parameters

ctx	pointer to output variable holding a context handle.	
-----	---	--

Returns

On success `U2FS_OK` (integer 0) is returned, and on errors an `u2fs_rc` error code.

u2fs_done ()

```
void
u2fs_done (u2fs_ctx_t *ctx);
```

Deallocate resources associated with context `ctx`.

Parameters

ctx	a context handle, from <code>u2fs_init()</code>	
-----	--	--

u2fs_set_origin ()

```
u2fs_rc
```

```
u2fs_set_origin (u2fs_ctx_t *ctx,  
                const char *origin);
```

Stores *origin* within *ctx*. If a value is already present, it is cleared and the memory is released.

Parameters

ctx	a context handle, from <code>u2fs_init()</code>	
origin	the origin of a registration request	

Returns

On success `U2FS_OK` (integer 0) is returned, and on errors a `u2fs_rc` error code.

u2fs_set_appid ()

```
u2fs_rc  
u2fs_set_appid (u2fs_ctx_t *ctx,  
               const char *appid);
```

Stores *appid* within *ctx*. If a value is already present, it is cleared and the memory is released.

Parameters

ctx	a context handle, from <code>u2fs_init()</code>	
appid	the appid of a registration request	

Returns

On success `U2FS_OK` (integer 0) is returned, and on errors a `u2fs_rc` error code.

u2fs_set_challenge ()

```
u2fs_rc  
u2fs_set_challenge (u2fs_ctx_t *ctx,  
                   const char *challenge);
```

Stores a given *challenge* within *ctx*. If a value is already present, it is cleared and the memory is released.

Parameters

ctx	a context handle, from <code>u2fs_init()</code>	
challenge	a 43-byte long, websafe Base64 encoded challenge (viz RFC4648 Section 5)	

Returns

On success **U2FS_OK** (integer 0) is returned, and on errors an **u2fs_rc** error code.

u2fs_set_keyHandle ()

```
u2fs_rc
u2fs_set_keyHandle (u2fs_ctx_t *ctx,
                   const char *keyHandle);
```

Stores a given *keyHandle* within *ctx* . If a value is already present, it is cleared and the memory is released.

Parameters

ctx	a context handle, from u2fs_init()	
keyHandle	a registered key-handle in websafe Base64 form, to use for signing, as returned by the U2F registration.	

Returns

On success **U2FS_OK** (integer 0) is returned, and on errors an **u2fs_rc** error code.

u2fs_set_publicKey ()

```
u2fs_rc
u2fs_set_publicKey (u2fs_ctx_t *ctx,
                   const unsigned char *publicKey);
```

Decode *publicKey* and store within *ctx* . If a value is already present, it is cleared and the memory is released.

Parameters

ctx	a context handle, from u2fs_init()	
publicKey	a 65-byte raw EC public key as returned from registration.	

Returns

On success **U2FS_OK** (integer 0) is returned, and on errors a **u2fs_rc** error code.

u2fs_registration_challenge ()

```
u2fs_rc
u2fs_registration_challenge (u2fs_ctx_t *ctx,
                           char **output);
```

Get a U2F RegistrationData JSON structure, used as the challenge in a U2F device registration.

Parameters

ctx	a context handle, from <code>u2fs_init()</code>	
output	pointer to output string with JSON data of RegistrationData.	

Returns

On success `U2FS_OK` (integer 0) is returned, and on errors an `u2fs_rc` error code.

u2fs_registration_verify ()

```
u2fs_rc
u2fs_registration_verify (u2fs_ctx_t *ctx,
                          const char *response,
                          u2fs_reg_res_t **output);
```

Get a U2F registration response and check its validity.

Parameters

ctx	a context handle, from <code>u2fs_init()</code> .	
response	a U2F registration response message Base64 encoded.	
output	pointer to output structure containing the relevant data for a well formed request. Memory should be free'd.	

Returns

On success `U2FS_OK` (integer 0) is returned and `output` is filled up with the user public key, the key handle and the attestation certificate. On errors a `u2fs_rc` error code.

u2fs_get_registration_keyHandle ()

```
const char~*
u2fs_get_registration_keyHandle (u2fs_reg_res_t *result);
```

Get the Base64 keyHandle obtained during the U2F registration operation. The memory is allocate by the library, and must not be deallocated by the caller.

Parameters

result	a registration result obtained from <code>u2fs_registration_verify()</code>	
--------	---	--

Returns

On success the pointer to the buffer containing the keyHandle is returned, and on errors NULL.

u2fs_get_registration_publicKey ()

```
const char~*
u2fs_get_registration_publicKey (u2fs_reg_res_t *result);
```

Extract the raw user public key obtained during the U2F registration operation. The memory is allocated by the library, and must not be deallocated by the caller. The returned buffer pointer holds **U2FS_PUBLIC_KEY_LEN** bytes.

Parameters

result	a registration result obtained from u2fs_registration_verify()
--------	---

Returns

On success the pointer to the buffer containing the user public key is returned, and on errors NULL.

u2fs_free_reg_res ()

```
void
u2fs_free_reg_res (u2fs_reg_res_t *result);
```

Deallocate resources associated with *result*.

Parameters

result	a registration result as generated by u2fs_registration_verify()
--------	---

u2fs_authentication_challenge ()

```
u2fs_rc
u2fs_authentication_challenge (u2fs_ctx_t *ctx,
                               char **output);
```

Get a U2F AuthenticationData JSON structure, used as the challenge in a U2F authentication procedure.

Parameters

ctx	a context handle, from u2fs_init()
output	pointer to output string with JSON data of AuthenticationData.

Returns

On success **U2FS_OK** (integer 0) is returned, and on errors a **u2fs_rc** error code.

u2fs_authentication_verify ()

```
u2fs_rc
u2fs_authentication_verify (u2fs_ctx_t *ctx,
                           const char *response,
                           u2fs_auth_res_t **output);
```

Get a U2F authentication response and check its validity.

Parameters

ctx	a context handle, from u2fs_init()	
response	pointer to output string with JSON data.	
output	pointer to output structure containing the relevant data for a well formed request. Memory should be free'd.	

Returns

On a successful verification **U2FS_OK** (integer 0) is returned and *output* is filled with the authentication result (same as the returned value), the counter received from the token and the user presence information. On errors a **u2fs_rc** error code is returned.

u2fs_get_authentication_result ()

```
u2fs_rc
u2fs_get_authentication_result (u2fs_auth_res_t *result,
                               u2fs_rc *verified,
                               uint32_t *counter,
                               uint8_t *user_presence);
```

Unpack the authentication result obtained from a U2F authentication procedure into its components. If any of the output parameters is set to NULL, that parameter will be ignored.

Parameters

result	an authentication result obtained from u2fs_authentication_verify()	
verified	output parameter for the authentication result	
counter	output parameter for the counter value	
user_presence	output parameter for the user presence byte	

Returns

On success **U2FS_OK** is returned, and on errors a **u2fs_rc** error code. The value *verified* is set to **U2FS_OK** on a successful authentication, and to 0 otherwise *counter* is filled with the value of the counter provided by the token. A *user_presence* value of 1 will determine the actual presence of the user (yubikey touched) during the authentication.

u2fs_free_auth_res ()

```
void
u2fs_free_auth_res (u2fs_auth_res_t *result);
```

Deallocate resources associated with *result* .

Parameters

result	an authentication result as generated by u2fs_authentication_verify()
--------	--

Types and Values

U2FS_CHALLENGE_RAW_LEN

```
#define U2FS_CHALLENGE_RAW_LEN 32
```

U2FS_CHALLENGE_B64U_LEN

```
#define U2FS_CHALLENGE_B64U_LEN 43
```

U2FS_PUBLIC_KEY_LEN

```
#define U2FS_PUBLIC_KEY_LEN 65
```

U2FS_COUNTER_LEN

```
#define U2FS_COUNTER_LEN 4
```

enum u2fs_rc

Error codes.

Members

U2FS_OK	Success.
U2FS_MEMORY_ERROR	Memory error.

U2FS_JSON_ERROR	Json er- ror.
U2FS_BASE64_ERROR	Base64 er- ror.
U2FS_CRYPTO_ERROR	Cryptographic er- ror.
U2FS_ORIGIN_ERROR	Origin mis- match.
U2FS_CHALLENGE_ERROR	Challenge er- ror.
U2FS_SIGNATURE_ERROR	Signature mis- match.
U2FS_FORMAT_ERROR	Message for- mat er- ror.

enum u2fs_initflags

Flags passed to `u2fs_global_init()`.

Members

U2FS_DEBUG	Print de- bug mes- sages.
------------	---------------------------------------

u2fs_ctx_t

```
typedef struct u2fs_ctx u2fs_ctx_t;
```

u2fs_reg_res_t

```
typedef struct u2fs_reg_res u2fs_reg_res_t;
```

u2fs_auth_res_t

```
typedef struct u2fs_auth_res u2fs_auth_res_t;
```

1.2 u2f-server-version

u2f-server-version —

Functions

`const char *` | `u2fs_check_version ()`

Types and Values

<code>#define</code>	<code>U2FS_VERSION_STRING</code>
<code>#define</code>	<code>U2FS_VERSION_NUMBER</code>
<code>#define</code>	<code>U2FS_VERSION_MAJOR</code>
<code>#define</code>	<code>U2FS_VERSION_MINOR</code>
<code>#define</code>	<code>U2FS_VERSION_PATCH</code>

Description

Functions

`u2fs_check_version ()`

```
const char~*
u2fs_check_version (const char *req_version);
```

Check that the version of the library is at minimum the requested one and return the version string; return NULL if the condition is not satisfied. If a NULL is passed to this function, no check is done, but the version string is simply returned.

See `U2FS_VERSION_STRING` for a suitable *req_version* string.

Parameters

<code>req_version</code>	Required version number, or NULL.
--------------------------	--------------------------------------

Returns

Version string of run-time library, or NULL if the run-time library does not meet the required version number.

Types and Values

`U2FS_VERSION_STRING`

```
#define U2FS_VERSION_STRING "0.0.0"
```

Pre-processor symbol with a string that describe the header file version number. Used together with `u2fs_check_version()` to verify header file and run-time library consistency.

`U2FS_VERSION_NUMBER`

```
#define U2FS_VERSION_NUMBER 0x000000
```

Pre-processor symbol with a hexadecimal value describing the header file version number. For example, when the header version is 1.2.3 this symbol will have the value 0x01020300. The last two digits are only used between public releases, and will otherwise be 00.

U2FS_VERSION_MAJOR

```
#define U2FS_VERSION_MAJOR 0
```

Pre-processor symbol with a decimal value that describe the major level of the header file version number. For example, when the header version is 1.2.3 this symbol will be 1.

U2FS_VERSION_MINOR

```
#define U2FS_VERSION_MINOR 0
```

Pre-processor symbol with a decimal value that describe the minor level of the header file version number. For example, when the header version is 1.2.3 this symbol will be 2.

U2FS_VERSION_PATCH

```
#define U2FS_VERSION_PATCH 0
```

Pre-processor symbol with a decimal value that describe the patch level of the header file version number. For example, when the header version is 1.2.3 this symbol will be 3.

Chapter 2

Index

U

- u2fs_auth_res_t, 10
- u2fs_authentication_challenge, 7
- u2fs_authentication_verify, 8
- U2FS_CHALLENGE_B64U_LEN, 9
- U2FS_CHALLENGE_RAW_LEN, 9
- u2fs_check_version, 11
- U2FS_COUNTER_LEN, 9
- u2fs_ctx_t, 10
- u2fs_done, 3
- u2fs_free_auth_res, 9
- u2fs_free_reg_res, 7
- u2fs_get_authentication_result, 8
- u2fs_get_registration_keyHandle, 6
- u2fs_get_registration_publicKey, 7
- u2fs_global_done, 2
- u2fs_global_init, 2
- u2fs_init, 3
- u2fs_initflags, 10
- U2FS_PUBLIC_KEY_LEN, 9
- u2fs_rc, 9
- u2fs_reg_res_t, 10
- u2fs_registration_challenge, 5
- u2fs_registration_verify, 6
- u2fs_set_appid, 4
- u2fs_set_challenge, 4
- u2fs_set_keyHandle, 5
- u2fs_set_origin, 3
- u2fs_set_publicKey, 5
- u2fs_strerror, 2
- u2fs_strerror_name, 3
- U2FS_VERSION_MAJOR, 12
- U2FS_VERSION_MINOR, 12
- U2FS_VERSION_NUMBER, 11
- U2FS_VERSION_PATCH, 12
- U2FS_VERSION_STRING, 11
