Yubico Universal 2nd Factor C Library

COLLABORATORS			
	TITLE : Yubico Universal 2nd Factor C Library		
ACTION	*NAME*	*DATE*	*SIGNATURE*
WRITTEN BY		September 16, 2014	

REVISION HISTORY			
NUMBER	DATE	DESCRIPTION	NAME

Contents

Chapter 1

Yubico Universal 2nd Factor C Library

This is a C library that implements the host-side of the U2F protocol. More precisely, it provides an API for applications that wishes to talk to a U2F device and perform the U2F Register and U2F Authenticate operations.

1.1 u2f-host

u2f-host —

Functions

u2fh_rc	u2fh_global_init ()
void	u2fh_global_done ()
const char *	u2fh_strerror ()
const char *	u2fh_strerror_name ()
u2fh_rc	u2fh_devs_init ()
u2fh_rc	u2fh_devs_discover ()
void	u2fh_devs_done ()
u2fh_rc	u2fh_register ()
u2fh_rc	u2fh_authenticate ()
u2fh_rc	u2fh_sendrecv ()
u2fh_rc	u2fh_get_device_description ()
int	u2fh_is_alive ()

Description

Functions

u2fh_global_init ()

```
u2fh_rc
u2fh_global_init (u2fh_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 u2fh_initflags.	

Returns

On success U2FH_OK (integer 0) is returned, and on errors an u2fh_rc error code.

u2fh_global_done ()

```
void
u2fh_global_done (void);
```

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

u2fh_strerror ()

```
const char~*
u2fh_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 u2fh_global_init().

Parameters

err	error code	

Returns

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

u2fh_strerror_name ()

```
const char~*
u2fh_strerror_name (int err);
```

Convert return code to human readable string representing the error code symbol itself. For example, u2fh_strerror_name(U2FH_OK) returns the string "U2FH_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 u2fh_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.

u2fh_devs_init ()

```
u2fh_rc
u2fh_devs_init (u2fh_devs **devs);
```

Initialize device handle.

Parameters

devs	pointer to u2fh_devs type to initialize.	

Returns

On success U2FH_OK (integer 0) is returned, on memory allocation errors U2FH_MEMORY_ERROR is returned, or another u2fh_rc error code is returned.

u2fh_devs_discover ()

```
u2fh_rc
u2fh_devs_discover (u2fh_devs *devs,
                    unsigned *max_index);
```

Discover and open new devices. This function can safely be called several times and will free resources associated with unplugged devices and open new.

Parameters

devs	device handle, from u2fh_devs_init().	
max_index	will on return be set to the maximum index, may be NULL; if there is 1 device this will be 0, if there are 2 devices this will be 1, and so on.	

Returns

On success, U2FH_OK (integer 0) is returned, when no U2F device could be found U2FH_NO_U2F_DEVICE is returned, or another u2fh_rc error code.

u2fh_devs_done ()

```
void
u2fh_devs_done (u2fh_devs *devs);
```

Release all resources associated with *devs* . This function must be called when you are finished with a device handle.

Parameters

devs	device handle, from u2fh_devs_init().	

u2fh_register ()

```
u2fh_rc
u2fh_register (u2fh_devs *devs,
               const char *challenge,
               const char *origin,
               char **response,
               u2fh_cmdflags flags);
```

Perform the U2F Register operation.

Parameters

devs	a device set handle, from u2fh_devs_init() and u2fh_devs_discover().	
challenge	string with JSON data containing the challenge.	
origin	U2F origin URL.	
response	pointer to output string with JSON data.	
flags	set of ORed u2fh_cmdflags values.	

Returns

On success U2FH_OK (integer 0) is returned, and on errors an u2fh_rc error code.

u2fh_authenticate ()

```
u2fh_rc
u2fh_authenticate (u2fh_devs *devs,
                   const char *challenge,
                   const char *origin,
                   char **response,
                   u2fh_cmdflags flags);
```

Perform the U2F Authenticate operation.

Parameters

devs	a device handle, from u2fh_devs_init() and u2fh_devs_discover().	
challenge	string with JSON data containing the challenge.	
origin	U2F origin URL.	
response	pointer to output string with JSON data.	
flags	set of ORed u2fh_cmdflags values.	

Returns

On success U2FH_OK (integer 0) is returned, and on errors an u2fh_rc error code.

u2fh_sendrecv ()

```
u2fh_rc
u2fh_sendrecv (u2fh_devs *devs,
               unsigned  index,
               uint8_t cmd,
               const unsigned char *send,
               uint16_t sendlen,
               unsigned char *recv,
               size_t *recvlen);
```

Send a command with data to the device at *index* .

Parameters

devs	device handle, from u2fh_devs_init().	
index	index of device	
cmd	command to run	
send	buffer of data to send	
sendlen	length of data to send	
recv	buffer of data to receive	
recvlen	length of data to receive	

Returns

U2FH_OK on success, another u2fh_rc error code otherwise.

u2fh_get_device_description ()

```
u2fh_rc
u2fh_get_device_description (u2fh_devs *devs,
                             unsigned  index,
                             char *out,
                             size_t *len);
```

Get the device description of the device at *index* . Stores the string in *out* .

Parameters

devs	device_handle, from u2fh_devs_init().	
index	index of device	
out	buffer for storing device description	
len	maximum amount of data to store in *out* . Will be updated.	

Returns

U2FH_OK on success.

u2fh_is_alive ()

```
int
u2fh_is_alive (u2fh_devs *devs,
               unsigned  index);
```

Get the liveliness of the device *index* .

Parameters

devs	device_handle, from u2fh_devs_init().	
index	index of device	

Returns

1 if the device is considered alive, 0 otherwise.

Types and Values

1.2 u2f-host-types

u2f-host-types —

Types and Values

enum	u2fh_rc
enum	u2fh_initflags
enum	u2fh_cmdflags
typedef	u2fh_devs

Description

Functions

Types and Values

enum u2fh_rc

Error codes.

Members

U2FH_OK	Success.

U2FH_MEMORY_ERROR	Memory error.
U2FH_TRANSPORT_ERROR	Transport (e.g., USB) error.
U2FH_JSON_ERROR	Json error.
U2FH_BASE64_ERROR	Base64 error.
U2FH_NO_U2F_DEVICE	Missing U2F device.

enum u2fh_initflags

Flags passed to u2fh_global_init().

Members

U2FH_DEBUG	Print debug messages.

enum u2fh_cmdflags

Flags passed to u2fh_register() and u2fh_authenticate().

Members

U2FH_REQUEST_USER_PRESENCE	Request user presence.

u2fh_devs

```
typedef struct u2fh_devs u2fh_devs;
```

1.3 u2f-host-version

u2f-host-version —

Functions

const char *	u2fh_check_version ()

Types and Values

#define	U2FH_VERSION_STRING
#define	U2FH_VERSION_NUMBER
#define	U2FH_VERSION_MAJOR
#define	U2FH_VERSION_MINOR
#define	U2FH_VERSION_PATCH

Description

Functions

u2fh_check_version ()

```
const char~*
u2fh_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 U2FH_VERSION_STRING for a suitable *req_version* string.

Parameters

req_version	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

U2FH_VERSION_STRING

```
#define U2FH_VERSION_STRING "0.0"
```

Pre-processor symbol with a string that describe the header file version number. Used together with u2fh_check_version() to verify header file and run-time library consistency.

U2FH_VERSION_NUMBER

```
#define U2FH_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.

U2FH_VERSION_MAJOR

```
#define U2FH_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.

U2FH_VERSION_MINOR

```
#define U2FH_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.

U2FH_VERSION_PATCH

```
#define U2FH_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