Yubico YubiKey NEO Manager C Library

COLLABORATORS			
	TITLE : Yubico YubiKey NEO Manager C Library		
ACTION	*NAME*	*DATE*	*SIGNATURE*
WRITTEN BY		March 25, 2014	

REVISION HISTORY			
NUMBER	DATE	DESCRIPTION	NAME

Contents

Chapter 1

Yubico YubiKey NEO Manager C Library

This is a C library to interact with the YubiKey NEO. There is a command line tool "ykneomgr" for interactive use. It supports querying the YubiKey NEO for firmware version, operation mode (OTP/CCID) and serial number. You may also mode switch the device and manage applets (list, delete and install).

For more information about Yubico and the YubiKey, see: https://www.yubico.com/

1.1 ykneomgr

ykneomgr —

Synopsis

```
ykneomgr_rc         ykneomgr_global_init              (ykneomgr_initflags flags);
void                ykneomgr_global_done              (void);
const char *        ykneomgr_strerror                 (int err);
const char *        ykneomgr_strerror_name            (int err);
ykneomgr_rc         ykneomgr_init                     (ykneomgr_dev **dev);
void                ykneomgr_done                     (ykneomgr_dev *dev);
ykneomgr_rc         ykneomgr_list_devices             (ykneomgr_dev *dev,
                                                       char *devicestr,
                                                       size_t *len);
ykneomgr_rc         ykneomgr_connect                  (ykneomgr_dev *dev,
                                                       const char *name);
ykneomgr_rc         ykneomgr_discover                 (ykneomgr_dev *dev);
uint8_t             ykneomgr_get_version_major        (ykneomgr_dev *dev);
uint8_t             ykneomgr_get_version_minor        (ykneomgr_dev *dev);
uint8_t             ykneomgr_get_version_build        (ykneomgr_dev *dev);
uint8_t             ykneomgr_get_mode                 (ykneomgr_dev *dev);
uint32_t            ykneomgr_get_serialno             (ykneomgr_dev *dev);
ykneomgr_rc         ykneomgr_modeswitch               (ykneomgr_dev *dev,
                                                       uint8_t mode);
ykneomgr_rc         ykneomgr_authenticate             (ykneomgr_dev *dev,
                                                       const uint8_t *key);
ykneomgr_rc         ykneomgr_applet_list              (ykneomgr_dev *dev,
                                                       char *appletstr,
                                                       size_t *len);
ykneomgr_rc         ykneomgr_applet_delete            (ykneomgr_dev *dev,
                                                       const uint8_t *aid,
```

```
                                                    size_t aidlen);
ykneomgr_rc          ykneomgr_applet_install       (ykneomgr_dev *dev,
                                                    const char *capfile);
```

Description

Details

ykneomgr_global_init ()

```
ykneomgr_rc          ykneomgr_global_init          (ykneomgr_initflags flags);
```

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

flags : initialization flags, ORed ykneomgr_initflags.

Returns : On success YKNEOMGR_OK (integer 0) is returned, and on errors an ykneomgr_rc error code.

ykneomgr_global_done ()

```
void                 ykneomgr_global_done          (void);
```

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

ykneomgr_strerror ()

```
const char *         ykneomgr_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 ykneomgr_global_init().

err : error code

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

ykneomgr_strerror_name ()

```
const char *         ykneomgr_strerror_name        (int err);
```

Convert return code to human readable string representing the error code symbol itself. For example, ykneomgr_strerror_name(YKNEOM returns the string "YKNEOMGR_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 ykneomgr_global_init().

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.

ykneomgr_init ()

```
ykneomgr_rc         ykneomgr_init                   (ykneomgr_dev **dev);
```

Create a YubiKey NEO device handle. The handle must be deallocated using ykneomgr_done() when you no longer need it.

dev **:** pointer to newly allocated device handle.

Returns **:** On success, YKNEOMGR_OK (integer 0) is returned, or another ykneomgr_rc error code.

ykneomgr_done ()

```
void                ykneomgr_done                   (ykneomgr_dev *dev);
```

Release all resources allocated to a YubiKey NEO device handle.

dev **:** device handle to deallocate, created by ykneomgr_init().

ykneomgr_list_devices ()

```
ykneomgr_rc         ykneomgr_list_devices           (ykneomgr_dev *dev,
                                                     char *devicestr,
                                                     size_t *len);
```

List devices.

dev **:** a ykneomgr_dev device handle.

devicestr **:** output buffer to hold string, or NULL.

len **:** on input length of *devicestr* buffer, on output holds output length

Returns **:** On success, YKNEOMGR_OK (integer 0) is returned, or another ykneomgr_rc error code.

ykneomgr_connect ()

```
ykneomgr_rc         ykneomgr_connect                (ykneomgr_dev *dev,
                                                     const char *name);
```

Establish connection to a named PCSC device and verify that it has the YubiKey NEO applet. The *name* string should be a PCSC device name; you can use the command line tool "pcsc_scan" to list connected devices.

dev **:** a ykneomgr_dev device handle.

name **:** input string with device name to connect to.

Returns **:** On success, YKNEOMGR_OK (integer 0) is returned, when no device could be found YKNEOMGR_NO_DEVICE is returned, or another ykneomgr_rc error code.

ykneomgr_discover ()

```
ykneomgr_rc         ykneomgr_discover               (ykneomgr_dev *dev);
```

Discover and establish connection to a YubiKey NEO. The function will return an error if more than one device is present, or if no device is present.

dev **:** a ykneomgr_dev device handle.

Returns **:** On success, YKNEOMGR_OK (integer 0) is returned, when no device could be found YKNEOMGR_NO_DEVICE is returned, when too many devices are present YKNEOMGR_TOO_MANY_DEVICES is returned, or another ykneomgr_rc error code.

ykneomgr_get_version_major ()

```
uint8_t             ykneomgr_get_version_major          (ykneomgr_dev *dev);
```

Get major version of a YubiKey NEO. Versions are in the form of MAJOR.MINOR.BUILD, for example 3.0.4, in which case this function would return 3.

dev : a ykneomgr_dev device handle.

Returns : the YubiKey NEO major version number.

ykneomgr_get_version_minor ()

```
uint8_t             ykneomgr_get_version_minor          (ykneomgr_dev *dev);
```

Get minor version of a YubiKey NEO. Versions are in the form of MINOR.MINOR.BUILD, for example 3.0.4, in which case this function would return 0.

dev : a ykneomgr_dev device handle.

Returns : the YubiKey NEO minor version number.

ykneomgr_get_version_build ()

```
uint8_t             ykneomgr_get_version_build          (ykneomgr_dev *dev);
```

Get build version of a YubiKey NEO. Versions are in the form of BUILD.MINOR.BUILD, for example 3.0.4, in which case this function would return 4.

dev : a ykneomgr_dev device handle.

Returns : the YubiKey NEO build version number.

ykneomgr_get_mode ()

```
uint8_t             ykneomgr_get_mode                   (ykneomgr_dev *dev);
```

Get mode of a YubiKey NEO.

dev : a ykneomgr_dev device handle.

Returns : the YubiKey NEO device mode.

ykneomgr_get_serialno ()

```
uint32_t            ykneomgr_get_serialno               (ykneomgr_dev *dev);
```

Get serial number of a YubiKey NEO, if visible.

dev : a ykneomgr_dev device handle.

Returns : the YubiKey NEO device mode, or 0 if not visible.

ykneomgr_modeswitch ()

```
ykneomgr_rc         ykneomgr_modeswitch                 (ykneomgr_dev *dev,
                                                         uint8_t mode);
```

Mode switch a YubiKey NEO.

dev **:** a ykneomgr_dev device handle.

mode **:** new mode to switch the device into

Returns **:** On success, YKNEOMGR_OK (integer 0) is returned, or another ykneomgr_rc error code.

ykneomgr_authenticate ()

```
ykneomgr_rc         ykneomgr_authenticate               (ykneomgr_dev *dev,
                                                         const uint8_t *key);
```

Authenticate to the device, to prepare for privileged function access.

dev **:** a ykneomgr_dev device handle.

key **:** Double-DES key in binary, 16 bytes

Returns **:** On success, YKNEOMGR_OK (integer 0) is returned, or another ykneomgr_rc error code.

ykneomgr_applet_list ()

```
ykneomgr_rc         ykneomgr_applet_list                (ykneomgr_dev *dev,
                                                         char *appletstr,
                                                         size_t *len);
```

List installed applets.

dev **:** a ykneomgr_dev device handle.

appletstr **:** output buffer to hold string, or NULL.

len **:** on input length of *appletstr* buffer, on output holds output length

Returns **:** On success, YKNEOMGR_OK (integer 0) is returned, or another ykneomgr_rc error code.

ykneomgr_applet_delete ()

```
ykneomgr_rc         ykneomgr_applet_delete              (ykneomgr_dev *dev,
                                                         const uint8_t *aid,
                                                         size_t aidlen);
```

Delete specified applet.

dev **:** a ykneomgr_dev device handle.

aid **:** aid to delete.

aidlen **:** length of *aid* buffer.

Returns **:** On success, YKNEOMGR_OK (integer 0) is returned, or another ykneomgr_rc error code.

ykneomgr_applet_install ()

```
ykneomgr_rc         ykneomgr_applet_install             (ykneomgr_dev *dev,
                                                         const char *capfile);
```

Install specified applet.

dev **:** a ykneomgr_dev device handle.

capfile **:** string with path filename to CAP file

Returns **:** On success, YKNEOMGR_OK (integer 0) is returned, or another ykneomgr_rc error code.

1.2 ykneomgr-types

ykneomgr-types —

Synopsis

```
enum                ykneomgr_rc;
enum                ykneomgr_initflags;
typedef             ykneomgr_dev;
```

Description

Details

enum ykneomgr_rc

```
typedef enum {
  YKNEOMGR_OK = 0,
  YKNEOMGR_MEMORY_ERROR = -1,
  YKNEOMGR_NO_DEVICE = -2,
  YKNEOMGR_TOO_MANY_DEVICES = -3,
  YKNEOMGR_BACKEND_ERROR = -4,
} ykneomgr_rc;
```

Error codes.

YKNEOMGR_OK Success.

YKNEOMGR_MEMORY_ERROR Memory error.

YKNEOMGR_NO_DEVICE No device found.

YKNEOMGR_TOO_MANY_DEVICES Too many devices found.

YKNEOMGR_BACKEND_ERROR Input/Output error.

enum ykneomgr_initflags

```
typedef enum {
  YKNEOMGR_DEBUG = 1
} ykneomgr_initflags;
```

Flags passed to ykneomgr_global_init().

YKNEOMGR_DEBUG Print debug messages.

ykneomgr_dev

```
typedef struct ykneomgr_dev ykneomgr_dev;
```

1.3 ykneomgr-version

ykneomgr-version —

Synopsis

```
#define             YKNEOMGR_VERSION_STRING
#define             YKNEOMGR_VERSION_NUMBER
#define             YKNEOMGR_VERSION_MAJOR
#define             YKNEOMGR_VERSION_MINOR
#define             YKNEOMGR_VERSION_PATCH
const char *        ykneomgr_check_version              (const char *req_version);
```

Description

Details

YKNEOMGR_VERSION_STRING

```
#define YKNEOMGR_VERSION_STRING "0.1.0"
```

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

YKNEOMGR_VERSION_NUMBER

```
#define YKNEOMGR_VERSION_NUMBER 0x000100
```

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.

YKNEOMGR_VERSION_MAJOR

```
#define YKNEOMGR_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.

YKNEOMGR_VERSION_MINOR

```
#define YKNEOMGR_VERSION_MINOR 1
```

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.

YKNEOMGR_VERSION_PATCH

```
#define YKNEOMGR_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.

ykneomgr_check_version ()

```
const char *        ykneomgr_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 YKNEOMGR_VERSION_STRING for a suitable *req_version* string.

***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.

Chapter 2

Index