Zymkey App Utils: C++

Generated by Doxygen 1.8.13

Contents

Chapter 1

Intro

The Zymkey App Utils library provides an API which allows user space applications to incorporate Zymkey's cryptographic features, including:

- Generation of random numbers
- Locking and unlocking of data objects
- ECDSA signature generation and verification

In addition, the Zymkey App Utils library provides interfaces for administrative functions, such as:

- Control of the LED
- Setting the i2c address (i2c units only)
- Setting the tap detection sensitivity

A Note About Files

Some of the interfaces can take a filename as an argument. The following rules must be observed when using these interfaces:

- Absolute path names must be provided.
- For destination filenames, the permissions of the path (or existing file) must be set:
  - Write permissions for all.
  - Write permissions for common group: in this case, user `zymbit` must be added to the group that has permissions for the destination directory path and/or existing file.
  - Destination path must be fully owned by user and/or group `zymbit`.
- Similar rules exist for source filenames:
  - Read permissions for all.
  - Read permissions for common group: in this case, user `zymbit` must be added to the group that has permissions for the source directory path and/or existing file.
  - Source path must be fully owned by user and/or group `zymbit`.

Crypto Features

Random Number Generation

This feature is useful when the default host random number generator is suspected of having cryptographic weakness. It can also be used to supplement existing random number generation sources. Zymkey bases its random number generation on an internal TRNG (True Random Number Generator) and performs well under Fourmilab's ent.

Data Locker

Zymkey includes a feature, called Data Locking. This feature is essentially an AES encryption of the data block followed by an ECDSA signature trailer.

Data Locker Keys

In addition to a unique ECDSA private/public key pair, each Zymkey has two unique AES keys that are programmed at the factory. These keys are referred to as "one-way" and "shared":

- "one-way": the one-way key is completely self contained on the Zymkey and is never exported or changeable. Consequently, data that is locked using a Zymkey cannot be unlocked on another system (host/SD card/↩ Zymkey: See Binding).
- "shared": the shared key is used whenever the data is intended to be published to the Zymbit cloud. Using the shared key allows the Zymbit cloud to unlock the data.

ECDSA Operations

Each Zymkey comes out of the factory with a unique ECDSA private/public key pair. The private key is randomly programmed within hardware at the time of manufactor and never exported. In fact, Zymbit doesn't even know what the value of the private key is.

There are three ECDSA operations available:

- Generate signature: the Zymkey is cabable of generating an ECDSA signature.
- Verification signature: the Zymkey is capable of verifying an ECDSA signature.
- Export the ECDSA public key and saving it to a file in PEM format. This operation is useful for generating a Certificate Signing Request (CSR).

Other Features

LED

The Zymkey has an LED which can be turned on, off or flashed at an interval.

i2c Address

For Zymkeys with an i2c interface, the base address can be changed to work around addressing conflicts. The default address is 0x30, but can be changed in the ranges 0x30 - 0x37 and 0x60 - 0x67.

Tap Sensitivity

The Zymkey has an accelerometer which can perform tap detection. The sensitivity of the tap detection is configurable.

Currently tap can only be detected via the Zymbit cloud.

Programming Language Support

Currently, C, C++ and Python are supported.

Binding

Before a Zymkey can be effectively used on a host computer, it must be "bound" to it. Binding is a process where a "fingerprint" is made which is composed of the host computer and its SD card serial numbers as well as the Zymkey serial number. If the host computer or SD card is changed from the time of binding, the Zymkey will refuse to accept commands.

To learn more about binding your zymkey, go to the Zymbit Community "Getting Started"page for your Zymkey model (e.g. `Getting Started with ZYMKEY`)

Chapter 2

Hierarchical Index

2.1 Class Hierarchy

This inheritance list is sorted roughly, but not completely, alphabetically:

Chapter 3

Class Index

3.1 Class List

Here are the classes, structs, unions and interfaces with brief descriptions:

Chapter 4

File Index

4.1 File List

Here is a list of all documented files with brief descriptions:

Chapter 5

Class Documentation

5.1 zkAppUtils::accelData Struct Reference

Structure typedef used for retrieval of accelerometer data.

`#include <zkAppUtilsClass.h>`

Public Attributes

- double **x**
- double **y**
- double **z**
- int **tapDirX**
- int **tapDirY**
- int **tapDirZ**

5.1.1 Detailed Description

Structure typedef used for retrieval of accelerometer data.

The documentation for this struct was generated from the following file:

- zkAppUtilsClass.h

5.2 zkAppUtils::commonException Class Reference

Exception class derived from std::exception.

`#include <zkAppUtilsClass.h>`

Inheritance diagram for zkAppUtils::commonException:

Collaboration diagram for zkAppUtils::commonException:

Public Member Functions

- commonException (std::string status)
  Constructor.
- virtual ~commonException () throw ()
  Destructor.
- const char * what () const throw ()
  Override of std::exception what() method.

5.2.1 Detailed Description

Exception class derived from std::exception.

5.2.2 Constructor & Destructor Documentation

5.2.2.1 commonException()

```
zkAppUtils::commonException::commonException (
            std::string status )
```

Constructor.

Parameters

status	A string which described the verbose exception

The documentation for this class was generated from the following file:

- zkAppUtilsClass.h

5.3 zkAppUtils::timeoutException Class Reference

Exception class derived from std::exception.

`#include <zkAppUtilsClass.h>`

Inheritance diagram for zkAppUtils::timeoutException:

Collaboration diagram for zkAppUtils::timeoutException:

Public Member Functions

- timeoutException (std::string status)
  Constructor.
- virtual ~timeoutException () throw ()
  Destructor.
- const char * what () const throw ()
  Override of std::exception what() method.

5.3.1 Detailed Description

Exception class derived from std::exception.

5.3.2 Constructor & Destructor Documentation

5.3.2.1 timeoutException()

```
zkAppUtils::timeoutException::timeoutException (
            std::string status )
```

Constructor.

Parameters

status	A string which described the verbose exception

The documentation for this class was generated from the following file:

- zkAppUtilsClass.h

5.4 zkAppUtils::zkClass Class Reference

The main class.

```
#include <zkAppUtilsClass.h>
```

Public Member Functions

- zkClass ()
  Constructor: a Zymkey context is opened.
- virtual ~zkClass ()
  Destructor: the Zymkey context is closed.
- void createRandDataFile (std::string &dst_filename, int rdata_sz)
  Write random data to a file.
- byteArray * getRandBytes (int rdata_sz)
  Generate a block of random data.
- void lockData (const std::string &src_pt_filename, const std::string &dst_ct_filename, bool use_shared_key=false)
  Locks data from a plaintext source file and stores locked data object to a destination file.
- void lockData (const byteArray &src_pt_data, const std::string &dst_ct_filename, bool use_shared_key=false)
  Locks data from a plaintext source byte container and stores locked data object to a destination file.
- byteArray * lockData (const std::string &src_pt_filename, bool use_shared_key=false)
  Locks data from a plaintext source file and stores locked data object in a container of unsigned bytes.

- byteArray * lockData (const byteArray &src_pt_data, bool use_shared_key=false)

  Locks data from a plaintext source byte container and stores locked data object in a container of unsigned bytes.
- void unlockData (const std::string &src_ct_filename, const std::string &dst_pt_filename, bool use_shared_↩ key=false)

  Unocks a locked data object from source file and stores unlocked data object to a destination file.
- void unlockData (const byteArray &src_ct_data, const std::string &dst_pt_filename, bool use_shared_↩ key=false)

  Unlocks a locked data object contained in an unsigned byte container and stores plaintext data to a destination file.
- byteArray * unlockData (const std::string &src_ct_filename, bool use_shared_key=false)

  Unlocks a locked data object from a plaintext source file and stores plaintext data in a container of unsigned bytes.
- byteArray * unlockData (const byteArray &src_ct_bytes, bool use_shared_key=false)

  Unlocks a locked data object contained in an unsigned byte container and stores plaintext data in a container of unsigned bytes.
- byteArray * genECDSASigFromDigest (byteArray &digest, int slot=0)

  Generate a signature from a data digest using the Zymkey's private key.
- bool verifyECDSASigFromDigest (byteArray &digest, byteArray &sig, int slot=0)

  Verify a signature from a data digest using the Zymkey's public key. The public key is not given as an input. Rather, the Zymkey uses its own copy of the private key. This insures that the public key that matches the private key is used.
- bool verifyECDSASigFromDigest (byteArray &digest, byteArray &sig, byteArray &foreign_pubkey, bool sig_↩ is_der=false, ZK_FOREIGN_PUBKEY_TYPE foreign_pubkey_type=ZK_FOREIGN_PUBKEY_NISTP256)

  Verify a signature from a data digest using the Zymkey's public key. The public key is not given as an input. Rather, the Zymkey uses its own copy of the private key. This insures that the public key that matches the private key is used.
- void saveECDSAPubKey2File (std::string dst_filename, int slot=0)

  Save the public key that matches the Zymkey's private key into a PEM formatted file. Mainly used for Certificate Signing Request (CSR) generation.
- byteArray * getECDSAPubKey (int slot=0)

  Get a container of bytes which contains the ECDSA public key.
- void ledOff ()

  Turn LED off.
- void ledOn ()

  Turn LED on.
- void ledFlash (uint32_t on_ms, uint32_t off_ms=0, uint32_t num_flashes=0)

  Flash LED a certain number of times.
- void setI2CAddr (int addr)

  Sets the i2c address (i2c versions only). Used in case of i2c bus device address conflict.
- uint32_t getTime (bool precise_time=false)

  Get current GMT time.
- void setTapSensitivity (float pct, ZK_ACCEL_AXIS_TYPE axis=ZK_ACCEL_AXIS_ALL)

  Sets the sensitivity of the tap detection as a percentage for an individual axis or all axes.
- void waitForTap (uint32_t timeout_ms)

  Wait for a tap event to be detected.
- void getAccelerometerData (accelData &accel_data)

  Get current accelerometer data and tap info.
- void waitForPerimeterEvent (uint32_t timeout_ms)

  Wait for a perimeter breach event to be detected.
- void getPerimeterDetectInfo (uint32_t **timestamp_sec, int &num_timestamps)

  Get current perimeter detect info.
- void clearPerimeterDetectEvents ()

  Clear perimeter detect info.
- void setPerimeterEventAction (int channel, uint32_t action_flags)

  Set perimeter breach action.

5.4.1 Detailed Description

The main class.

5.4.2 Constructor & Destructor Documentation

5.4.2.1 zkClass()

```
zkAppUtils::zkClass::zkClass ( )
```

Constructor: a Zymkey context is opened.

Exceptions

zkAppUtilsException	upon error

5.4.3 Member Function Documentation

5.4.3.1 clearPerimeterDetectEvents()

```
void zkAppUtils::zkClass::clearPerimeterDetectEvents ( )
```

Clear perimeter detect info.

This function clears all perimeter detect info and rearms all perimeter detect channels

Exceptions

zkAppUtilsException	upon error

5.4.3.2 createRandDataFile()

```
void zkAppUtils::zkClass::createRandDataFile (
            std::string & dst_filename,
            int rdata_sz )
```

Write random data to a file.

Parameters

dst_filename	A string containing the absolute path to the file where random data is written
rdata_sz	The number of random bytes to generate

Exceptions

zkAppUtilsException	upon error

5.4.3.3 genECDSASigFromDigest()

```
byteArray* zkAppUtils::zkClass::genECDSASigFromDigest (
            byteArray & digest,
            int slot = 0 )
```

Generate a signature from a data digest using the Zymkey's private key.

Parameters

digest	Currently, this should be a SHA256 digest
slot	The key slot to use for verification.

Returns

Byte container with binary signature

Exceptions

zkAppUtilsException	upon error

5.4.3.4 getAccelerometerData()

```
void zkAppUtils::zkClass::getAccelerometerData (
            accelData & accel_data )
```

Get current accelerometer data and tap info.

This function gets the most recent accelerometer data in units of g forces plus the tap direction per axis. Array index 0 = x 1 = y 2 = z

Parameters

accel_data	(output) The current accelerometer data and tap information

Exceptions

zkAppUtilsException	upon error

5.4.3.5 getECDSAPubKey()

```
byteArray* zkAppUtils::zkClass::getECDSAPubKey (
            int slot = 0 )
```

Get a container of bytes which contains the ECDSA public key.

Parameters

slot	Reserved for future use.

Returns

Byte container with binary public key

Exceptions

zkAppUtilsException	upon error

5.4.3.6 getPerimeterDetectInfo()

```
void zkAppUtils::zkClass::getPerimeterDetectInfo (
            uint32_t ** timestamp_sec,
            int & num_timestamps )
```

Get current perimeter detect info.

This function gets the timestamp of the first perimeter detect event for the given channel

Parameters

timestamps_sec	(output) The timestamps for when any events occurred. 0 if no events have occurred on a given channel. This array is allocated by this routine and so it must be freed by the caller. num_timestamps (output) The number of timestamps in the returned array

Exceptions

zkAppUtilsException	upon error

5.4.3.7 getRandBytes()

```
byteArray* zkAppUtils::zkClass::getRandBytes (
            int rdata_sz )
```

Generate a block of random data.

Parameters

rdata_sz	The number of random bytes to generate

Returns

A pointer to container with the random bytes

Exceptions

zkAppUtilsException	upon error

5.4.3.8 getTime()

```
uint32_t zkAppUtils::zkClass::getTime (
            bool precise_time = false )
```

Get current GMT time.

This method is called to get the time directly from a Zymkey's Real Time Clock (RTC)

Parameters

precise_time	(input) If true, this API returns the time after the next second falls. This means that the caller could be blocked up to one second. If false, the API returns immediately with the current time reading.

Returns

The time in seconds from the epoch (Jan. 1, 1970).

Exceptions

zkAppUtilsException	upon error

5.4.3.9 ledFlash()

```
void zkAppUtils::zkClass::ledFlash (
            uint32_t on_ms,
            uint32_t off_ms = 0,
            uint32_t num_flashes = 0 )
```

Flash LED a certain number of times.

Parameters

on_ms	Number of milliseconds that the LED will be on during a flash cycle
off_ms	Number of milliseconds that the LED will be off during a flash cycle
num_flashes	Number of flash cycles to execute. 0 = infinite.

Exceptions

zkAppUtilsException	upon error

5.4.3.10 ledOff()

```
void zkAppUtils::zkClass::ledOff ( )
```

Turn LED off.

Exceptions

zkAppUtilsException	upon error

5.4.3.11 ledOn()

```
void zkAppUtils::zkClass::ledOn ( )
```

Turn LED on.

Exceptions

zkAppUtilsException	upon error

5.4.3.12 lockData() [1/4]

```
void zkAppUtils::zkClass::lockData (
```

```
            const std::string & src_pt_filename,
            const std::string & dst_ct_filename,
            bool use_shared_key = false )
```

Locks data from a plaintext source file and stores locked data object to a destination file.

Parameters

src_pt_filename	Absolute path to source plaintext file
dst_ct_filename	Absolute path to destination file which will contain locked data object
use_shared_key	Determines whether one-way or shared key is used for locking. Default = false.

Exceptions

zkAppUtilsException	upon error

5.4.3.13 lockData() [2/4]

```
void zkAppUtils::zkClass::lockData (
            const byteArray & src_pt_data,
            const std::string & dst_ct_filename,
            bool use_shared_key = false )
```

Locks data from a plaintext source byte container and stores locked data object to a destination file.

Parameters

src_pt_data	Unsigned byte container which holds source plaintext data
dst_ct_filename	Absolute path to destination file which will contain locked data object
use_shared_key	Determines whether one-way or shared key is used for locking. Default = false.

Exceptions

zkAppUtilsException	upon error

5.4.3.14 lockData() [3/4]

```
byteArray* zkAppUtils::zkClass::lockData (
            const std::string & src_pt_filename,
            bool use_shared_key = false )
```

Locks data from a plaintext source file and stores locked data object in a container of unsigned bytes.

Parameters

src_pt_filename	Absolute path to source plaintext file
use_shared_key	Determines whether one-way or shared key is used for locking. Default = false.

Returns

Byte container with locked data

Exceptions

zkAppUtilsException	upon error

5.4.3.15 lockData() [4/4]

```
byteArray* zkAppUtils::zkClass::lockData (
            const byteArray & src_pt_data,
            bool use_shared_key = false )
```

Locks data from a plaintext source byte container and stores locked data object in a container of unsigned bytes.

Parameters

src_pt_data	Unsigned byte container which holds source plaintext data
use_shared_key	Determines whether one-way or shared key is used for locking. Default = false.

Returns

Byte container with locked data

Exceptions

zkAppUtilsException	upon error

5.4.3.16 saveECDSAPubKey2File()

```
void zkAppUtils::zkClass::saveECDSAPubKey2File (
            std::string dst_filename,
            int slot = 0 )
```

Save the public key that matches the Zymkey's private key into a PEM formatted file. Mainly used for Certificate Signing Request (CSR) generation.

Parameters

dst_filename	Absolute location where the PEM formatted file is to be stored
slot	Reserved for future use

Exceptions

zkAppUtilsException	upon error

5.4.3.17 setI2CAddr()

```
void zkAppUtils::zkClass::setI2CAddr (
            int addr )
```

Sets the i2c address (i2c versions only). Used in case of i2c bus device address conflict.

Parameters

addr	The i2c address to set. Upon successful change, the Zymkey will reset itself. However, if the address is the same as the current setting, the Zymkey will not reset. Valid address ranges are 0x30 - 0x37 and 0x60 - 0x67.

Exceptions

zkAppUtilsException	upon error

5.4.3.18 setPerimeterEventAction()

```
void zkAppUtils::zkClass::setPerimeterEventAction (
            int channel,
            uint32_t action_flags )
```

Set perimeter breach action.

This function specifies the action to take when a perimeter breach event occurs. The possible actions are any combination of:

1. Notify host
2. Zymkey self-destruct

Parameters

channel	(input) The channel that the action flags will be applied to action_flags (input) The actions to apply to the perimeter event channel: (a) Notify (ZK_PERIMETER_EVENT_ACTION_NOTIFY) (b) Self-destruct (ZK_PERIMETER_EVENT_ACTION_SELF_DESTRUCT)

Exceptions

zkAppUtilsException	upon error

5.4.3.19 setTapSensitivity()

```
void zkAppUtils::zkClass::setTapSensitivity (
            float pct,
            ZK_ACCEL_AXIS_TYPE axis = ZK_ACCEL_AXIS_ALL )
```

Sets the sensitivity of the tap detection as a percentage for an individual axis or all axes.

Parameters

pct	Sensitivity expressed in percentage. 0% = off, 100% = maximum.
axis	X, Y, Z or all (default).

Exceptions

zkAppUtilsException	upon error

5.4.3.20 unlockData() [1/4]

```
void zkAppUtils::zkClass::unlockData (
            const std::string & src_ct_filename,
            const std::string & dst_pt_filename,
            bool use_shared_key = false )
```

Unocks a locked data object from source file and stores unlocked data object to a destination file.

Parameters

src_ct_filename	Absolute path to source file which contains locked data object
dst_pt_filename	Absolute path to destination file which will contain unlocked plaintext data
use_shared_key	Determines whether one-way or shared key is used for locking. Default = false.

Exceptions

zkAppUtilsException	upon error

5.4.3.21 unlockData() [2/4]

```
void zkAppUtils::zkClass::unlockData (
            const byteArray & src_ct_data,
            const std::string & dst_pt_filename,
            bool use_shared_key = false )
```

Unlocks a locked data object contained in an unsigned byte container and stores plaintext data to a destination file.

Parameters

src_ct_data	Unsigned byte container which holds locked data object
dst_pt_filename	Absolute path to destination file which will contain unlocked plaintext data
use_shared_key	Determines whether one-way or shared key is used for locking. Default = false.

Exceptions

zkAppUtilsException	upon error

5.4.3.22 unlockData() [3/4]

```
byteArray* zkAppUtils::zkClass::unlockData (
            const std::string & src_ct_filename,
            bool use_shared_key = false )
```

Unlocks a locked data object from a plaintext source file and stores plaintext data in a container of unsigned bytes.

Parameters

src_ct_filename	Absolute path to source file which contains locked data object
use_shared_key	Determines whether one-way or shared key is used for locking. Default = false.

Returns

Byte container with plaintext data

Exceptions

zkAppUtilsException	upon error

5.4.3.23 unlockData() [4/4]

```
byteArray* zkAppUtils::zkClass::unlockData (
            const byteArray & src_ct_bytes,
            bool use_shared_key = false )
```

Unlocks a locked data object contained in an unsigned byte container and stores plaintext data in a container of unsigned bytes.

Parameters

src_ct_data	Unsigned byte container which holds locked data object
use_shared_key	Determines whether one-way or shared key is used for locking. Default = false.

Returns

Byte container with plaintext data

Exceptions

zkAppUtilsException	upon error

5.4.3.24 verifyECDSASigFromDigest() [1/2]

```
bool zkAppUtils::zkClass::verifyECDSASigFromDigest (
            byteArray & digest,
            byteArray & sig,
            int slot = 0 )
```

Verify a signature from a data digest using the Zymkey's public key. The public key is not given as an input. Rather, the Zymkey uses its own copy of the private key. This insures that the public key that matches the private key is used.

Parameters

digest	Currently, this should be a SHA256 digest
sig	The signature to verify
slot	The key slot to use for verification.

Returns

true = signature verification passed, false = signature verification failed

5.4.3.25 verifyECDSASigFromDigest() [2/2]

```
bool zkAppUtils::zkClass::verifyECDSASigFromDigest (
            byteArray & digest,
            byteArray & sig,
            byteArray & foreign_pubkey,
            bool sig_is_der = false,
            ZK_FOREIGN_PUBKEY_TYPE foreign_pubkey_type = ZK_FOREIGN_PUBKEY_NISTP256 )
```

Verify a signature from a data digest using the Zymkey's public key. The public key is not given as an input. Rather, the Zymkey uses its own copy of the private key. This insures that the public key that matches the private key is used.

Parameters

digest	Currently, this should be a SHA256 digest
sig	The signature to verify
foreign_pubkey	A foreign public key to perform the verifications against.

Returns

true = signature verification passed, false = signature verification failed

5.4.3.26 waitForPerimeterEvent()

```
void zkAppUtils::zkClass::waitForPerimeterEvent (
            uint32_t timeout_ms )
```

Wait for a perimeter breach event to be detected.

This function is called in order to wait for a perimeter breach event to occur. This function blocks the calling thread unless called with a timeout of zero. Note that, in order to receive perimeter events, the zymkey must have been configured to notify the host on either or both of the perimeter detect channels via a call to "zkSetPerimeterEvent↩ Action".

Parameters

timeout_ms	(input) The maximum amount of time in milliseconds to wait for a perimeter event to arrive.

Exceptions

zkAppUtilsException	upon error

5.4.3.27 waitForTap()

```
void zkAppUtils::zkClass::waitForTap (
            uint32_t timeout_ms )
```

Wait for a tap event to be detected.

This function is called in order to wait for a tap event to occur. This function blocks the calling thread unless called with a timeout of zero.

Parameters

timeout_ms	(input) The maximum amount of time in milliseconds to wait for a tap event to arrive.

Exceptions

zkAppUtilsTimeoutException	upon timeout or zkAppUtilsException upon other errors

The documentation for this class was generated from the following file:

- zkAppUtilsClass.h

Chapter 6

File Documentation

6.1 zkAppUtilsClass.h File Reference

C++ interface to Zymkey Application Utilities Library.

```
#include <zk_app_utils.h>
#include <exception>
#include <string>
#include <vector>
```

Include dependency graph for zkAppUtilsClass.h:

Index