Zymkey App Utils

Generated by Doxygen 1.8.13

Contents

1 Intro **1**

2 Data Structure Index **5**

2.1 Data Structures 5

3 File Index **7**

3.1 File List 7

4 Data Structure Documentation **9**

4.1 zkAccelAxisDataType Struct Reference 9

4.1.1 Detailed Description 9

4.1.2 Field Documentation 9

4.1.2.1 g 9

4.1.2.2 tapDirection 9

5 File Documentation **11**

5.1 zk_app_utils.h File Reference 11

5.1.1 Detailed Description 13

5.1.2 Function Documentation 14

5.1.2.1 zkClearPerimeterDetectEvents() 14

5.1.2.2 zkClose() 14

5.1.2.3 zkCreateRandDataFile() 15

5.1.2.4 zkGenECDSASigFromDigest() 15

5.1.2.5 zkGetAccelerometerData() 16

5.1.2.6 zkGetECDSAPubKey() 16

5.1.2.7 zkGetPerimeterDetectInfo() 17
5.1.2.8 zkGetRandBytes() 17
5.1.2.9 zkGetTime() 17
5.1.2.10 zkLEDFlash() 18
5.1.2.11 zkLEDOff() 18
5.1.2.12 zkLEDOn() 19
5.1.2.13 zkLockDataB2B() 19
5.1.2.14 zkLockDataB2F() 20
5.1.2.15 zkLockDataF2B() 20
5.1.2.16 zkLockDataF2F() 21
5.1.2.17 zkOpen() 22
5.1.2.18 zkSaveECDSAPubKey2File() 22
5.1.2.19 zkSetI2CAddr() 22
5.1.2.20 zkSetPerimeterEventAction() 23
5.1.2.21 zkSetTapSensitivity() 23
5.1.2.22 zkUnlockDataB2B() 24
5.1.2.23 zkUnlockDataB2F() 25
5.1.2.24 zkUnlockDataF2B() 25
5.1.2.25 zkUnlockDataF2F() 26
5.1.2.26 zkVerifyECDSASigFromDigest() 27
5.1.2.27 zkVerifyECDSASigFromDigestWithForeignKey() 27
5.1.2.28 zkWaitForPerimeterEvent() 28
5.1.2.29 zkWaitForTap() 28

Index **29**

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

Data Structure Index

2.1 Data Structures

Here are the data structures with brief descriptions:

zkAccelAxisDataType
ZkGetAccelerometer data output 9

Chapter 3

File Index

3.1 File List

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

zk_app_utils.h
C interface to Zymkey Application Utilities Library 11

Chapter 4

Data Structure Documentation

4.1 zkAccelAxisDataType Struct Reference

zkGetAccelerometer data output.

```
#include <zk_app_utils.h>
```

Data Fields

- double g
- int tapDirection

4.1.1 Detailed Description

zkGetAccelerometer data output.

4.1.2 Field Documentation

4.1.2.1 g

```
double zkAccelAxisDataType::g
```

the axis reading in units of g-force

4.1.2.2 tapDirection

```
int zkAccelAxisDataType::tapDirection
```

the direction of the force along the axis which caused a tap event: -1 = negative +1 = positive 0 = did not cause a tap event

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

- zk_app_utils.h

Chapter 5

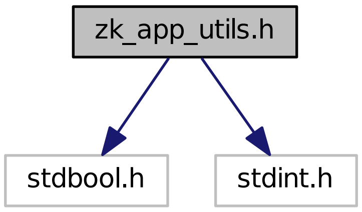
File Documentation

5.1 zk_app_utils.h File Reference

C interface to Zymkey Application Utilities Library.

```
#include <stdbool.h>
#include <stdint.h>
```

Include dependency graph for zk_app_utils.h:



Data Structures

- struct zkAccelAxisDataType

  zkGetAccelerometer data output.

Macros

- #define ZK_PERIMETER_EVENT_ACTION_NOTIFY (1 << 0)

  Perimeter breach action flag definitions.
- #define **ZK_PERIMETER_EVENT_ACTION_SELF_DESTRUCT** (1 << 1)

Typedefs

- typedef void ∗ **zkCTX**
- typedef enum ZK_FOREIGN_PUBKEY_TYPE ZK_FOREIGN_PUBKEY_TYPE

  Supported key types for signature validation against foreign public keys.
- typedef enum ZK_ACCEL_AXIS_TYPE ZK_ACCEL_AXIS_TYPE

  Accelerometer axis enum, used to set tap sensitivity.
- typedef struct zkAccelAxisDataType zkAccelAxisDataType

  zkGetAccelerometer data output.

Enumerations

- enum ZK_FOREIGN_PUBKEY_TYPE { **ZK_FOREIGN_PUBKEY_NISTP256**, **ZK_FOREIGN_PUBKEY_↩ SECP256K1** }

  Supported key types for signature validation against foreign public keys.
- enum ZK_ACCEL_AXIS_TYPE { **ZK_ACCEL_AXIS_X**, **ZK_ACCEL_AXIS_Y**, **ZK_ACCEL_AXIS_Z**, **ZK_↩ ACCEL_AXIS_ALL** }

  Accelerometer axis enum, used to set tap sensitivity.

Functions

- int zkOpen (zkCTX ∗ctx)

  Open a Zymkey context.
- int zkClose (zkCTX ctx)

  Close a Zymkey context.
- int zkCreateRandDataFile (zkCTX ctx, const char ∗dst_filename, int rdata_sz)

  Fill a file with random numbers.
- int zkGetRandBytes (zkCTX ctx, uint8_t ∗∗rdata, int rdata_sz)

  Get an array of random bytes.
- int zkLockDataF2F (zkCTX ctx, const char ∗src_pt_filename, const char ∗dst_ct_filename, bool use_shared↩ _key)

  Lock up source (plaintext) data from a file and store the results (ciphertext) in a destination file.
- int zkLockDataB2F (zkCTX ctx, const uint8_t ∗src_pt, int src_pt_sz, const char ∗dst_ct_filename, bool use↩ _shared_key)

  Lock up source (plaintext) data from a byte array and store the results (ciphertext) in a destination file.
- int zkLockDataF2B (zkCTX ctx, const char ∗src_pt_filename, uint8_t ∗∗dst_ct, int ∗dst_ct_sz, bool use_↩ shared_key)

  Lock up source (plaintext) data from a file and store the results (ciphertext) in a destination byte array.
- int zkLockDataB2B (zkCTX ctx, const uint8_t ∗src_pt, int src_pt_sz, uint8_t ∗∗dst_ct, int ∗dst_ct_sz, bool use_shared_key)

  Lock up source (plaintext) data from a byte array and store the results (ciphertext) in a destination byte array.
- int zkUnlockDataF2F (zkCTX ctx, const char ∗src_ct_filename, const char ∗dst_pt_filename, bool use_↩ shared_key)

  Unlock source (ciphertext) data from a file and store the results (plaintext) in a destination file.
- int zkUnlockDataB2F (zkCTX ctx, const uint8_t ∗src_ct, int src_ct_sz, const char ∗dst_pt_filename, bool use_shared_key)

  Unlock source (ciphertext) data from a byte array and store the results (plaintext) in a destination file.
- int zkUnlockDataF2B (zkCTX ctx, const char ∗src_ct_filename, uint8_t ∗∗dst_pt, int ∗dst_pt_sz, bool use_↩ shared_key)

  Unlock source (ciphertext) data from a file and store the results (plaintext) in a destination byte array.

- int zkUnlockDataB2B (zkCTX ctx, const uint8_t ∗src_ct, int src_ct_sz, uint8_t ∗∗dst_pt, int ∗dst_pt_sz, bool use_shared_key)

  Unlock source (ciphertext) data from a byte array and store the results (plaintext) in a destination byte array.
- int zkGenECDSASigFromDigest (zkCTX ctx, const uint8_t ∗digest, int slot, uint8_t ∗∗sig, int ∗sig_sz)

  Generate a signature using the Zymkey's ECDSA private key.
- int zkVerifyECDSASigFromDigest (zkCTX ctx, const uint8_t ∗digest, int slot, const uint8_t ∗sig, int sig_sz)

  Verify a signature using the Zymkey's ECDSA public key. The public is not specified in the parameter list to insure that the public key that matches the Zymkey's ECDSA private key is used.
- int zkVerifyECDSASigFromDigestWithForeignKey (zkCTX ctx, const uint8_t ∗digest, const uint8_t ∗foreign←_pubkey, int foreign_pubkey_sz, const uint8_t ∗sig, int sig_sz, bool sig_is_der, int ec_curve_type)

  Verify a signature using a foreign ECDSA public key.
- int zkSaveECDSAPubKey2File (zkCTX ctx, const char ∗filename, int slot)

  Store the ECDSA public key to a file in PEM format.
- int zkGetECDSAPubKey (zkCTX ctx, uint8_t ∗∗pk, int ∗pk_sz, int slot)

  Gets the ECDSA public key and stores in a byte array created by this function.
- int zkLEDOff (zkCTX ctx)

  Turns the LED off.
- int zkLEDOn (zkCTX ctx)

  Turns the LED on.
- int zkLEDFlash (zkCTX ctx, uint32_t on_ms, uint32_t off_ms, uint32_t num_flashes)

  Flashes the LED.
- int zkSetI2CAddr (zkCTX ctx, int addr)

  Sets the i2c address of the Zymkey (i2c versions only)
- int zkGetTime (zkCTX ctx, uint32_t ∗epoch_time_sec, bool precise_time)

  Get current GMT time.
- int zkSetTapSensitivity (zkCTX ctx, int axis, float pct)

  Sets the sensitivity of tap operations.
- int zkWaitForTap (zkCTX ctx, uint32_t timeout_ms)

  Wait for a tap event to be detected.
- int zkGetAccelerometerData (zkCTX ctx, zkAccelAxisDataType ∗x, zkAccelAxisDataType ∗y, zkAccelAxis←DataType ∗z)

  Get current accelerometer data and tap info.
- int zkWaitForPerimeterEvent (zkCTX ctx, uint32_t timeout_ms)

  Wait for a perimeter breach event to be detected.
- int zkGetPerimeterDetectInfo (zkCTX ctx, uint32_t ∗∗timestamps_sec, int ∗num_timestamps)

  Get current perimeter detect info.
- int zkClearPerimeterDetectEvents (zkCTX ctx)

  Clear perimeter detect events.
- int zkSetPerimeterEventAction (zkCTX ctx, int channel, uint32_t action_flags)

  Set perimeter breach action.

5.1.1 Detailed Description

C interface to Zymkey Application Utilities Library.

Author

Scott Miller

Version

1.0

Date

November 17, 2016

Copyright

Zymbit, Inc.

This file contains the C API to the the Zymkey Application Utilities library. This API facilitates writing user space applications which use Zymkey to perform cryptographic operations, such as:

1. Signing of payloads using ECDSA
2. Verification of payloads that were signed using Zymkey
3. Exporting the public key that matches Zymkey's private key
4. "Locking" and "unlocking" data objects
5. Generating random data Additionally, there are functions for changing the i2c address (i2c units only), setting tap sensitivity and controlling the LED.

5.1.2 Function Documentation

5.1.2.1 zkClearPerimeterDetectEvents()

```
int zkClearPerimeterDetectEvents (
            zkCTX ctx )
```

Clear perimeter detect events.

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

Returns

0 for success, less than 0 for failure.

5.1.2.2 zkClose()

```
int zkClose (
            zkCTX ctx )
```

Close a Zymkey context.

Parameters

ctx	(input) The Zymkey context to close

Returns

0 for success, less than 0 for failure.

5.1.2.3 zkCreateRandDataFile()

```
int zkCreateRandDataFile (
             zkCTX ctx,
             const char * dst_filename,
             int rdata_sz )
```

Fill a file with random numbers.

Parameters

ctx	(input) Zymkey context.
dst_filename	(input) Absolute path name for the destination file.
rdata_sz	(input) The number of random bytes to generate.

Returns

0 for success, less than 0 for failure.

5.1.2.4 zkGenECDSASigFromDigest()

```
int zkGenECDSASigFromDigest (
             zkCTX ctx,
             const uint8_t * digest,
             int slot,
             uint8_t ** sig,
             int * sig_sz )
```

Generate a signature using the Zymkey's ECDSA private key.

Parameters

ctx	(input) Zymkey context.
digest	(input) This parameter contains the digest of the data that will be used to generate the signature.
slot	(input) The key slot to generate a signature from. This parameter is only valid for Zymkey models 4i and beyond.
sig	(output) A pointer to a pointer to an array of unsigned bytes which contains the generated signature. This pointer is created by this function and must be freed by the application when no longer needed.
sig_sz	(output) A pointer to an integer which contains the size of the signature.

Returns

0 for success, less than 0 for failure.

5.1.2.5 zkGetAccelerometerData()

```
int zkGetAccelerometerData (
            zkCTX ctx,
            zkAccelAxisDataType * x,
            zkAccelAxisDataType * y,
            zkAccelAxisDataType * z )
```

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.

Parameters

x	(output) x axis accelerometer information y (output) y axis accelerometer information z (output) z axis accelerometer information

Returns

0 for success, less than 0 for failure.

5.1.2.6 zkGetECDSAPubKey()

```
int zkGetECDSAPubKey (
            zkCTX ctx,
            uint8_t ** pk,
            int * pk_sz,
            int slot )
```

Gets the ECDSA public key and stores in a byte array created by this function.

Parameters

ctx	(input) Zymkey context.
pk	(output) Pointer to a pointer created by this function which contains the public key.
pk_sz	(output) Pointer to an integer which contains the size of the public key.
slot	(input) The key slot to retrieve. Only valid for model 4i and above.

Returns

0 for success, less than 0 for failure.

5.1.2.7 zkGetPerimeterDetectInfo()

```
int zkGetPerimeterDetectInfo (
            zkCTX ctx,
            uint32_t ** timestamps_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 the event occurred. 0 if no events have occurred. num_timestamps (output) The number of timestamps in the returned array

Returns

0 for success, less than 0 for failure.

5.1.2.8 zkGetRandBytes()

```
int zkGetRandBytes (
            zkCTX ctx,
            uint8_t ** rdata,
            int rdata_sz )
```

Get an array of random bytes.

Parameters

ctx	(input) Zymkey context.
rdata	(input) Pointer to a pointer of bytes.
rdata_sz	(input) The number of random bytes to generate.

Returns

0 for success, less than 0 for failure.

5.1.2.9 zkGetTime()

```
int zkGetTime (
            zkCTX ctx,
            uint32_t * epoch_time_sec,
            bool precise_time )
```

Get current GMT time.

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

Parameters

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

0 for success, less than 0 for failure.

5.1.2.10 zkLEDFlash()

```
int zkLEDFlash (
            zkCTX ctx,
            uint32_t on_ms,
            uint32_t off_ms,
            uint32_t num_flashes )
```

Flashes the LED.

Parameters

ctx	(input) Zymkey context.
on_ms	(input) The amount of time, in milliseconds, that the LED will stay on during a flash cycle.
off_ms	(input) The amount of time, in milliseconds, that the LED will stay off during a flash cycle.
num_flashes	(input) The number of on/off flash cycles to complete. If this parameter is 0, then the LED will flash indefinitely.

Returns

0 for success, less than 0 for failure.

5.1.2.11 zkLEDOff()

```
int zkLEDOff (
            zkCTX ctx )
```

Turns the LED off.

Parameters

ctx	(input) Zymkey context.

Returns

0 for success, less than 0 for failure.

5.1.2.12 zkLEDOn()

```
int zkLEDOn (
             zkCTX ctx )
```

Turns the LED on.

Parameters

ctx	(input) Zymkey context.

Returns

0 for success, less than 0 for failure.

5.1.2.13 zkLockDataB2B()

```
int zkLockDataB2B (
             zkCTX ctx,
             const uint8_t * src_pt,
             int src_pt_sz,
             uint8_t ** dst_ct,
             int * dst_ct_sz,
             bool use_shared_key )
```

Lock up source (plaintext) data from a byte array and store the results (ciphertext) in a destination byte array.

This function encrypts and signs a block of plaintext data and stores the result in a binary byte array.

Note

(See zkLockDataF2F for notes about keys)

Parameters

ctx	(input) Zymkey context.
src_pt	(input) Binary plaintext source byte array.
src_pt_sz	(input) Size of plaintext source data.
dst_ct	(output) A pointer to a pointer to an array of unsigned bytes created by this function. This pointer must be freed by the application when no longer needed.
dst_ct_sz	(output) A pointer to an integer which contains the size of the destination array.
use_shared_key	(input) Specifies if shared key is to be used. See zkLockDataF2F.

Returns

0 for success, less than 0 for failure.

5.1.2.14 zkLockDataB2F()

```
int zkLockDataB2F (
            zkCTX ctx,
            const uint8_t * src_pt,
            int src_pt_sz,
            const char * dst_ct_filename,
            bool use_shared_key )
```

Lock up source (plaintext) data from a byte array and store the results (ciphertext) in a destination file.

This function encrypts and signs a block of binary plaintext data and stores the result in a destination file.

Note

(See zkLockDataF2F for notes about keys)

Parameters

ctx	(input) Zymkey context.
src_pt	(input) Binary plaintext source byte array.
src_pt_sz	(input) Size of plaintext source data.
dst_ct_filename	(input) The absolute path to the file where the destination (ciphertext) data should be deposited.
use_shared_key	(input) Specifies if shared key is to be used. See zkLockDataF2F.

Returns

0 for success, less than 0 for failure.

5.1.2.15 zkLockDataF2B()

```
int zkLockDataF2B (
            zkCTX ctx,
            const char * src_pt_filename,
            uint8_t ** dst_ct,
            int * dst_ct_sz,
            bool use_shared_key )
```

Lock up source (plaintext) data from a file and store the results (ciphertext) in a destination byte array.

This function encrypts and signs a block of plaintext data from a file and stores the result in a binary byte array.

Note

(See zkLockDataF2F for notes about keys)

Parameters

ctx	(input) Zymkey context.
src_pt_filename	(input) The absolute path to the file where the source (plaintext) data is located.
dst_ct	(output) A pointer to a pointer to an array of unsigned bytes created by this function. This pointer must be freed by the application when no longer needed.
dst_ct_sz	(output) A pointer to an integer which contains the size of the destination array.
use_shared_key	(input) Specifies if shared key is to be used. See zkLockDataF2F.

Returns

0 for success, less than 0 for failure.

5.1.2.16 zkLockDataF2F()

```
int zkLockDataF2F (
            zkCTX ctx,
            const char * src_pt_filename,
            const char * dst_ct_filename,
            bool use_shared_key )
```

Lock up source (plaintext) data from a file and store the results (ciphertext) in a destination file.

This function encrypts and signs a block of plaintext data from a file and stores the result in a destination file.

Note

The zymkey has two keys that can be used for locking/unlocking operations, designated as 'shared' and 'one-way'.

1. The one-way key is meant to lock up data only on the local host computer. Data encrypted using this key cannot be exported and deciphered anywhere else.
2. The shared key is meant for publishing data to other sources that have the capability to generate the shared key, such as the Zymbit cloud server.

Parameters

ctx	(input) Zymkey context.
src_pt_filename	(input) The absolute path to the file where the source (plaintext) data is located.
dst_ct_filename	(input) The absolute path to the file where the destination (ciphertext) data should be deposited.
use_shared_key	(input) This parameter specifies which key will be used to used to lock the data up. A value of 'false' specifies that the Zymkey will use the one-way key whereas 'true' specifies that the shared key will be used. Specify 'true' for publishing data to another that has the shared key (e.g. Zymbit cloud) and 'False' when the data is meant to reside exclusively withing the host computer.

Returns

0 for success, less than 0 for failure.

5.1.2.17 zkOpen()

```
int zkOpen (
            zkCTX * ctx )
```

Open a Zymkey context.

Parameters

ctx	(output) returns a pointer to a Zymkey context.

Returns

0 for success, less than 0 for failure.

5.1.2.18 zkSaveECDSAPubKey2File()

```
int zkSaveECDSAPubKey2File (
            zkCTX ctx,
            const char * filename,
            int slot )
```

Store the ECDSA public key to a file in PEM format.

This function is useful for generating Certificate Signing Requests (CSR).

Parameters

ctx	(input) Zymkey context.
filename	(input) Filename where PEM formatted public key is to be stored.
slot	(input) The key slot to retrieve. Only valid for model 4i and above.

Returns

0 for success, less than 0 for failure.

5.1.2.19 zkSetI2CAddr()

```
int zkSetI2CAddr (
            zkCTX ctx,
            int addr )
```

Sets the i2c address of the Zymkey (i2c versions only)

This method should be called if the i2c address of the Zymkey is shared with another i2c device on the same i2c bus. The default i2c address for Zymkey units is 0x30. Currently, the address may be set in the ranges of 0x30 - 0x37 and 0x60 - 0x67. After successful completion of this command, the Zymkey will reset itself.

Parameters

addr	(input) The i2c address that the Zymkey will set itself to.

Returns

0 for success, less than 0 for failure.

5.1.2.20 zkSetPerimeterEventAction()

```
int zkSetPerimeterEventAction (
             zkCTX ctx,
             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)

Returns

0 for success, less than 0 for failure.

5.1.2.21 zkSetTapSensitivity()

```
int zkSetTapSensitivity (
             zkCTX ctx,
             int axis,
             float pct )
```

Sets the sensitivity of tap operations.

This method permits setting the sensitivity of the tap detection feature. Each axis may be individually configured or all at once.

Parameters

axis	(input) The axis to configure. This parameter should contain one of the values in the enum typedef ACCEL_AXIS_TYPE.
pct	(input) The sensitivity expressed as percentage. 1. 0% = Shut down: Tap detection should not occur along the axis. 2. 100% = Maximum sensitivity.

Returns

0 for success, less than 0 for failure.

5.1.2.22 zkUnlockDataB2B()

```
int zkUnlockDataB2B (
            zkCTX ctx,
            const uint8_t * src_ct,
            int src_ct_sz,
            uint8_t ** dst_pt,
            int * dst_pt_sz,
            bool use_shared_key )
```

Unlock source (ciphertext) data from a byte array and store the results (plaintext) in a destination byte array.

This function verifies a locked object signature and decrypts the associated ciphertext data.

Note

(See zkLockDataF2F for notes about keys)

Parameters

ctx	(input) Zymkey context.
src_ct	(input) Binary ciphertext source byte array.
src_ct_sz	(input) Size of ciphertext source data.
dst_pt	(output) A pointer to a pointer to an array of unsigned bytes created by this function. This pointer must be freed by the application when no longer needed.
dst_pt_sz	(output) A pointer to an integer which contains the size of the destination array.
use_shared_key	(input) Specifies if shared key is to be used. See zkLockDataF2F.

Returns

0 for success, less than 0 for failure.

5.1.2.23 zkUnlockDataB2F()

```
int zkUnlockDataB2F (
            zkCTX ctx,
            const uint8_t * src_ct,
            int src_ct_sz,
            const char * dst_pt_filename,
            bool use_shared_key )
```

Unlock source (ciphertext) data from a byte array and store the results (plaintext) in a destination file.

This function verifies a locked object signature and decrypts the associated ciphertext data.

Note

(See zkLockDataF2F for notes about keys)

Parameters

ctx	(input) Zymkey context.
src_ct	(input) Binary ciphertext source byte array.
src_ct_sz	(input) Size of ciphertext source data.
dst_pt_filename	(input) The absolute path to the file where the destination (plaintext) data should be deposited.
use_shared_key	(input) Specifies if shared key is to be used. See zkLockDataF2F.

Returns

0 for success, less than 0 for failure.

5.1.2.24 zkUnlockDataF2B()

```
int zkUnlockDataF2B (
            zkCTX ctx,
            const char * src_ct_filename,
            uint8_t ** dst_pt,
            int * dst_pt_sz,
            bool use_shared_key )
```

Unlock source (ciphertext) data from a file and store the results (plaintext) in a destination byte array.

This function verifies a locked object signature and decrypts the associated ciphertext data.

Note

(See zkLockDataF2F for notes about keys)

Parameters

ctx	(input) Zymkey context.
src_ct_filename	(input) The absolute path to the file where the source (ciphertext) data is located.
dst_pt	(output) A pointer to a pointer to an array of unsigned bytes created by this function. This pointer must be freed by the application when no longer needed.
dst_pt_sz	(output) A pointer to an integer which contains the size of the destination array.
use_shared_key	(input) Specifies if shared key is to be used. See zkLockDataF2F.

Returns

0 for success, less than 0 for failure.

5.1.2.25 zkUnlockDataF2F()

```
int zkUnlockDataF2F (
            zkCTX ctx,
            const char * src_ct_filename,
            const char * dst_pt_filename,
            bool use_shared_key )
```

Unlock source (ciphertext) data from a file and store the results (plaintext) in a destination file.

This function verifies a locked object signature and decrypts the associated ciphertext data.

Note

(See zkLockDataF2F for notes about keys)

Parameters

ctx	(input) Zymkey context.
src_ct_filename	(input) The absolute path to the file where the source (ciphertext) data is located.
dst_pt_filename	(input) The absolute path to the file where the destination (plaintext) data should be deposited.
use_shared_key	(input) This parameter specifies which key will be used to used to lock the data up. A value of 'false' specifies that the Zymkey will use the one-way key whereas 'true' specifies that the shared key will be used. Specify 'true' for publishing data to another that has the shared key (e.g. Zymbit cloud) and 'False' when the data is meant to reside exclusively withing the host computer.

Returns

0 for success, less than 0 for failure.

5.1.2.26 zkVerifyECDSASigFromDigest()

```
int zkVerifyECDSASigFromDigest (
            zkCTX ctx,
            const uint8_t * digest,
            int slot,
            const uint8_t * sig,
            int sig_sz )
```

Verify a signature using the Zymkey's ECDSA public key. The public is not specified in the parameter list to insure that the public key that matches the Zymkey's ECDSA private key is used.

Parameters

ctx	(input) Zymkey context.
digest	(input) This parameter contains the digest of the data that will be used to generate the signature.
slot	(input) The key slot to generate a signature from. This parameter is only valid for Zymkey models 4i and beyond.
sig	(input) Array of bytes which contains the signature.
sig_sz	(input) Size of signature.

Returns

0 for signature verification failed, 1 for signature verification passed, less than 0 for general failure.

5.1.2.27 zkVerifyECDSASigFromDigestWithForeignKey()

```
int zkVerifyECDSASigFromDigestWithForeignKey (
            zkCTX ctx,
            const uint8_t * digest,
            const uint8_t * foreign_pubkey,
            int foreign_pubkey_sz,
            const uint8_t * sig,
            int sig_sz,
            bool sig_is_der,
            int ec_curve_type )
```

Verify a signature using a foreign ECDSA public key.

Parameters

ctx	(input) Zymkey context.
digest	(input) This parameter contains the digest of the data that will be used to generate the signature.
foreign_pubkey	(input) A byte array in binary representation of the foreign public key which is paired to the private key which was used to generate the signature. This array must contain the uncompressed specifier which means that the preamble byte of '0x04' must be in the first byte. For example, a public key corresponding to a 256 bit EC curve would be 65 bytes long.
foreign_pubkey_size	(input) The foreign public key size
sig	(input) Array of bytes which contains the signature.
sig_sz	(input) Size of signature.
sig_is_der	(input) If the input signature is in DER format, set to true.
ec_curve_type	(input) Type of curve to verify against. Maps to ZK_FOREIGN_PUBKEY_TYPE.

Returns

0 for signature verification failed, 1 for signature verification passed, less than 0 for general failure.

5.1.2.28 zkWaitForPerimeterEvent()

```
int zkWaitForPerimeterEvent (
            zkCTX ctx,
            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.

Returns

0 for success, less than 0 for failure, -ETIMEDOUT when no perimeter events detected within the specified timeout

5.1.2.29 zkWaitForTap()

```
int zkWaitForTap (
            zkCTX ctx,
            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.

Returns

0 for success, less than 0 for failure, -ETIMEDOUT when no tap events detected within the specified timeout

Index

g
    zkAccelAxisDataType, 9

tapDirection
    zkAccelAxisDataType, 9

zk_app_utils.h, 11
    zkClearPerimeterDetectEvents, 14
    zkClose, 14
    zkCreateRandDataFile, 15
    zkGenECDSASigFromDigest, 15
    zkGetAccelerometerData, 16
    zkGetECDSAPubKey, 16
    zkGetPerimeterDetectInfo, 16
    zkGetRandBytes, 17
    zkGetTime, 17
    zkLEDFlash, 18
    zkLEDOff, 18
    zkLEDOn, 19
    zkLockDataB2B, 19
    zkLockDataB2F, 20
    zkLockDataF2B, 20
    zkLockDataF2F, 21
    zkOpen, 22
    zkSaveECDSAPubKey2File, 22
    zkSetI2CAddr, 22
    zkSetPerimeterEventAction, 23
    zkSetTapSensitivity, 23
    zkUnlockDataB2B, 24
    zkUnlockDataB2F, 25
    zkUnlockDataF2B, 25
    zkUnlockDataF2F, 26
    zkVerifyECDSASigFromDigest, 26
    zkVerifyECDSASigFromDigestWithForeignKey, 27
    zkWaitForPerimeterEvent, 28
    zkWaitForTap, 28
zkAccelAxisDataType, 9
    g, 9
    tapDirection, 9
zkClearPerimeterDetectEvents
    zk_app_utils.h, 14
zkClose
    zk_app_utils.h, 14
zkCreateRandDataFile
    zk_app_utils.h, 15
zkGenECDSASigFromDigest
    zk_app_utils.h, 15
zkGetAccelerometerData
    zk_app_utils.h, 16
zkGetECDSAPubKey
    zk_app_utils.h, 16
zkGetPerimeterDetectInfo
    zk_app_utils.h, 16
zkGetRandBytes
    zk_app_utils.h, 17
zkGetTime
    zk_app_utils.h, 17
zkLEDFlash
    zk_app_utils.h, 18
zkLEDOff
    zk_app_utils.h, 18
zkLEDOn
    zk_app_utils.h, 19
zkLockDataB2B
    zk_app_utils.h, 19
zkLockDataB2F
    zk_app_utils.h, 20
zkLockDataF2B
    zk_app_utils.h, 20
zkLockDataF2F
    zk_app_utils.h, 21
zkOpen
    zk_app_utils.h, 22
zkSaveECDSAPubKey2File
    zk_app_utils.h, 22
zkSetI2CAddr
    zk_app_utils.h, 22
zkSetPerimeterEventAction
    zk_app_utils.h, 23
zkSetTapSensitivity
    zk_app_utils.h, 23
zkUnlockDataB2B
    zk_app_utils.h, 24
zkUnlockDataB2F
    zk_app_utils.h, 25
zkUnlockDataF2B
    zk_app_utils.h, 25
zkUnlockDataF2F
    zk_app_utils.h, 26
zkVerifyECDSASigFromDigest
    zk_app_utils.h, 26
zkVerifyECDSASigFromDigestWithForeignKey
    zk_app_utils.h, 27
zkWaitForPerimeterEvent
    zk_app_utils.h, 28
zkWaitForTap
    zk_app_utils.h, 28