

Zymkey App Utils

Generated by Doxygen 1.8.13

Contents

- 1 Intro** **1**

- 2 Class Index** **5**
 - 2.1 Class List 5

- 3 File Index** **7**
 - 3.1 File List 7

- 4 Class Documentation** **9**
 - 4.1 zkAccelAxisDataType Struct Reference 9
 - 4.1.1 Detailed Description 9
 - 4.1.2 Member Data 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

- Index** **13**

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 manufacture 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 capable 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

Class Index

2.1 Class List

Here are the classes, structs, unions and interfaces 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

Class Documentation

4.1 zkAccelAxisDataType Struct Reference

zkGetAccelerometer data output.

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

Public Attributes

- double [g](#)
- int [tapDirection](#)

4.1.1 Detailed Description

zkGetAccelerometer data output.

4.1.2 Member Data 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:

Index

g

zkAccelAxisDataType, [9](#)

tapDirection

zkAccelAxisDataType, [9](#)

zk_app_utils.h, [11](#)

zkAccelAxisDataType, [9](#)

g, [9](#)

tapDirection, [9](#)