

ZKFinger Reader SDK Development Guide C#

Version: 2.0

Date: Sep 2016

ZKFinger Reader SDK Development Guide

Copyright ©ZKTECO CO., LTD.2016 All rights reserved.

Release History

Date	Version	Remarks
May 21, 2016	1.0	Basic version
Sep 18, 2016	2.0	Added 2.0 interface, keep old interface

Contents

1 Overview	4
2 Privacy Policy	4
3 System Requirements.....	4
4 Installation and Deployment	4
5 Description of SDK Interfaces	5
5.1 Referenced Class Library	5
5.2 Description of the Class Library	5
5.3 Interface Description.....	6
5.3.1 Init	6
5.3.2 Terminate.....	6
5.3.3 GetDeviceCount	6
5.3.4 OpenDevice.....	6
5.3.5 CloseDevice	7
5.3.6 SetParameters	7
5.3.7 GetParameters	7
5.3.8 AcquireFingerprint	8
5.3.9 AcquireFingerprintImage	8
5.3.10 DBInit.....	9
5.3.11 DBFree	9
5.3.12 DBMerge.....	9
5.3.13 DBAdd	10
5.3.14 DBDel	10
5.3.15 DBClear	11
5.3.16 DBIdentify	11
5.3.17 DBMatch.....	12
5.3.18 Blob2Base64String	12
5.3.19 Base64String2Blob	12
5.3.20 ByteArray2Int	13
5.3.21 Int2ByteArray	13
5.3.22 ExtractFromImage.....	13
6 Appendixes.....	14
6.1 Parameter Codes	14
6.2 Error Codes	15



1 Overview

Thank you for using ZKFinger Reader SDK. Please read this document carefully before use to fast learn how to use ZKFinger Reader SDK.

2 Privacy Policy

You are authorized to use the software but you must make the following commitment to ZKTeco: You shall not use, copy, modify, lease, or transfer any part of the SDK beyond the clauses of this document.

3 System Requirements

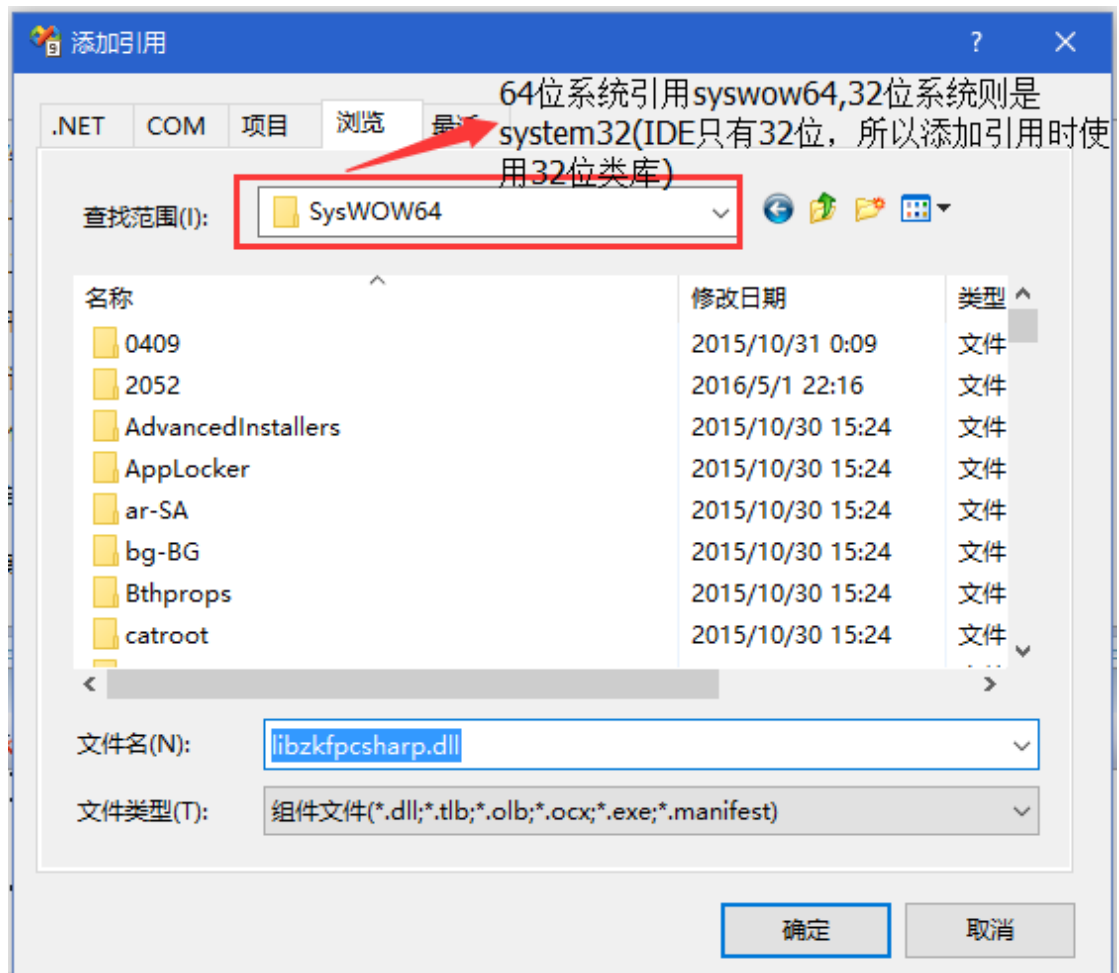
- 1) Operating system: Windows XP or a later version, .net framework 3.5
- 2) Applicable development language: C#

4 Installation and Deployment

- 1) Installation: Install ZKFinger SDK 5.x/ZKOnline SDK 5.x.

5 Description of SDK Interfaces

5.1 Referenced Class Library



5.2 Description of the Class Library

- Dynamic library
Libzkcsharp.dll(system32/syswow64)
- Namespace
libzkcsharp
- Class name
zkfp2

5.3 Interface Description

5.3.1 Init

[Function]

```
public static int Init()
```

[Purpose]

This function is used to initialize the device.

[Parameter Description]

[Return Value]

0 Succeeded

Others Failed (See the error code description.)

5.3.2 Terminate

[Function]

```
public static int Terminate()
```

[Purpose]

This function is used to release library resources.

[Parameter Description]

[Return Value]

0 Succeeded

Others Failed (See the error code description.)

5.3.3 GetDeviceCount

[Function]

```
public static int GetDeviceCount()
```

[Purpose]

This function is used to acquire the number of collected devices.

[Parameter Description]

[Return Value]

Device count

5.3.4 OpenDevice

[Function]

```
public static IntPtr OpenDevice(int index)
```

[Purpose]

This function is used to connect to a device.

[Parameter Description]

Index:

Device index (The values ranges from 0 to n and n indicates the device count minus 1.)

[Return Value]

Device Handle

5.3.5 CloseDevice

[Function]

```
public static int CloseDevice(IntPtr devHandle)
```

[Purpose]

This function is used to shut down a device.

[Parameter Description]

devHandle

Device handle

[Return Value]

0 Succeeded

Others Failed (See the error code description.)

5.3.6 SetParameters

[Function]

```
public static int SetParameters(IntPtr devHandle, int code, byte[] pramValue, int size)
```

[Purpose]

This function is used to set a parameter.

[Parameter Description]

devHandle

Device handle

code

Parameter code (See the Appendixes.)

pramValue

Parameter value

size

Parameter data length

[Return Value]

0 Succeeded

Others Failed (See the error code description.)

5.3.7 GetParameters

[Function]



```
)public static int GetParameters(IntPtr devHandle, int code, byte[] paramValue, ref int size)
```

[Purpose]

This function is used to acquire a parameter.

[Parameter Description]

devHandle

Device handle

code

Parameter code (See the Appendixes.)

paramValue

Parameter value

size

Returned parameter data length

[Return Value]

0 Succeeded

Others Failed (See the error code description.)

5.3.8 AcquireFingerprint

[Function]

```
public static int AcquireFingerprint(IntPtr devHandle, byte[] imgBuffer, byte[] template, ref int size)
```

[Purpose]

This function is used to capture a fingerprint image and template.

[Parameter Description]

devHandle

Device handle

imgBuffer

Returned image (The array size is imageWidth*imageHeight.)

template

Returned fingerprint template (It is recommended that 2048 bytes be pre-allocated.)

size[in/out]

[in] Template array length

[out] Fingerprint template length that is actually returned

[Return Value]

0 Succeeded

Others Failed (See the error code description.)

5.3.9 AcquireFingerprintImage

[Function]



```
public static int AcquireFingerprintImage(IntPtr devHandle, byte[] imgBuffer)
```

[Purpose]

This function is used to capture a fingerprint image.

[Parameter Description]

devHandle

Device handle

imgBuffer

Returned image (The array size is imageWidth*imageHeight.)

[Return Value]

0 Succeeded

Others Failed (See the error code description.)

5.3.10 DBInit

[Function]

```
public static IntPtr DBInit()
```

[Purpose]

This function is used to create an algorithm cache.

[Parameter Description]

[Return Value]

Cache handle

5.3.11 DBFree

[Function]

```
public static int DBFree(IntPtr dbHandle)
```

[Purpose]

This function is used to release an algorithm cache.

[Parameter Description]

dbHandle

Cache handle

[Return Value]

0 Succeeded

Others Failed (See the error code description.)

5.3.12 DBMerge

[Function]

```
public static int DBMerge(IntPtr dbHandle, byte[] temp1, byte[] temp2, byte[] temp3,  
byte[] regTemp, ref int regTempLen)
```

[Purpose]

This function is used to combine three pre-registered fingerprint templates as one registered fingerprint template.

[Parameter Description]

dbHandle

Cache handle

temp1

Pre-registered fingerprint template 1

temp2

Pre-registered fingerprint template 2

temp3

Pre-registered fingerprint template 3

regTemp

Returned registered template

regTempLen[in/out]

[in] regTemp array length

[out] Fingerprint template length that is actually returned

[Return Value]

0 Succeeded

Others Failed (See the error code description.)

5.3.13 DBAdd

[Function]

```
public static int DBAdd(IntPtr dbHandle, int fid, byte[] regTemp)
```

[Purpose]

This function is used to add a registered template to the memory.

[Parameter Description]

dbHandle

Cache handle

fid

Fingerprint ID (The fingerprint ID is returned after 1:N comparison is successfully conducted.)

regTemp

Registered template

[Return Value]

0 Succeeded

Others Failed (See the error code description.)

5.3.14 DBDel

[Function]

```
public static int DBDel (IntPtr dbHandle, int fid)
```

[Purpose]

This function is used to delete a registered fingerprint template from the memory.

[Parameter Description]

dbHandle

Cache handle

fid

Fingerprint ID (The fingerprint ID is returned after 1:N comparison is successfully conducted.)

[Return Value]

0 Succeeded

Others Failed (See the error code description.)

5.3.15 DBClear

[Function]

```
public static int DBClear(IntPtr dbHandle)
```

[Purpose]

This function is used to clear all fingerprint templates in the memory.

[Parameter Description]

dbHandle

Cache handle

[Return Value]

0 Succeeded

Others Failed (See the error code description.)

5.3.16 DBIdentify

[Function]

```
public static int DBIdentify(IntPtr dbHandle, byte[] temp, ref int fid, ref int score)
```

[Purpose]

This function is used to conduct 1:N comparison.

[Parameter Description]

dbHandle

Cache handle

temp

Template used for comparison

fid

Returned fingerprint ID

score

Returned comparison score

[Return Value]

0 Succeeded

Others Failed (See the error code description.)

5.3.17 DBMatch

[Function]

```
public static int Match(IntPtr dbHandle, byte[] temp1, byte[] temp2)
```

[Purpose]

This function is used to conduct 1:1 comparison on two fingerprint templates.

[Parameter Description]

dbHandle

Cache handle

temp1

Template 1 used for comparison

temp2

Template 2 used for comparison

[Return Value]

>=0 Comparison score

Others Failed (See the error code description.)

5.3.18 Blob2Base64String

[Function]

```
static public int Blob2Base64String(byte[] buf, int len, ref String strBase64)
```

[Purpose]

This function is used to convert a byte[] array into a Base64 string.

[Parameter Description]

buf

BLOB data

len

Length

strBase64

Returned Base64 string

[Return Value]

String length

5.3.19 Base64String2Blob

[Function]

```
static public byte[] Base64String2Blob(String strBase64)
```

[Purpose]

This function is used to convert a Base64 string into a byte[] array.

[Parameter Description]

strBase64

Base64 string

[Return Value]

Byte[] array

5.3.20 ByteArray2Int

[Function]

`static public boolean ByteArray2Int(byte[] buf, ref int value)`

[Purpose]

This function is used to convert a 4-byte array into an integer.

[Parameter Description]

buf

Byte array

value

Returned data

[Return Value]

true Succeeded

false Failed

5.3.21 Int2ByteArray

[Function]

`static public boolean Int2ByteArray(int value, byte[] buf)`

[Purpose]

This function is used to convert an integer into a 4-byte array.

[Parameter Description]

value

Data

buf

Byte array

[Return Value]

true Succeeded

false Failed

5.3.22 ExtractFromImage

[Function]

`public static int ExtractFromImage(IntPtr dbHandle, String FileName, int DPI, byte[] template, ref int size)`

[Purpose]

This function is used to extract a template from a BMP or JPG file.

[Parameter Description]

dbHandle

Cache handle

FileName

Full path of a file

DPI

Image DPI

template

Returned fingerprint template (It is recommended that 2048 bytes be pre-allocated.)

size[in/out]

[in] Template array length

[out] Fingerprint template length that is actually returned

[Return Value]

0 Succeeded

Others Failed (See the error code description.)

[Note]

Only the SDK of the standard version supports this function.

6 Appendixes

6.1 Parameter Codes

Parameter Code	Property	Data Type	Description
1	Read-only	Int	Image width
2	Read-only	Int	Image height
3	Read-write (supported only by the LIVEID20R currently)	Int	Image DPI (750/1000 is recommended for children.)
106	Read-only	Int	Image data size
1015	Read-only	4-byte array	VID&PID (The former two bytes indicate VID and the latter two bytes indicate PID.)
2002	Read-write (supported only by the LIVEID20R currently)	Int	Anti-fake function (1: enable; 0: disable)
2004	Read-only	Int	A fingerprint image is true if the lower five bits are all 1's (value&31==31).
1101	Read-only	String	Vendor information

Parameter Code	Property	Data Type	Description
1102	Read-only	String	Product name
1103	Read-only	String	Device SN
101	Write-only (Devices except the LIVE20R need to call a function to disable the parameter.)	Int	1 indicates that the white light blinks; 0 indicates that the parameter is disabled.
102	Write-only (Devices except the LIVE20R need to call a function to disable the parameter.)	Int	1 indicates that the green light blinks; 0 indicates that the parameter is disabled.
103	Write-only (Devices except the LIVE20R need to call a function to disable the parameter.)	Int	1 indicates that the red light blinks; 0 indicates that the parameter is disabled.
104	Write-only (not supported by the LIVE20R)	Int	1 indicates that buzzing is started; 0 indicates that the parameter is disabled.

6.2 Error Codes

classname:zkfp

```

public static int ZKFP_ERR_ALREADY_INIT = 1; /**< Initialized */
public static int ZKFP_ERR_OK = 0; /**< Operation succeeded */
public static int ZKFP_ERR_INITLIB = -1; /**< Failed to initialize the algorithm library */
public static int ZKFP_ERR_INIT = -2; /**< Failed to initialize the capture library */
public static int ZKFP_ERR_NO_DEVICE = -3; /**< No device connected */
public static int ZKFP_ERR_NOT_SUPPORT = -4; /**< Not supported by the interface */
public static int ZKFP_ERR_INVALID_PARAM = -5; /**< Invalid parameter */
public static int ZKFP_ERR_OPEN = -6; /**< Failed to start the device */
public static int ZKFP_ERR_INVALID_HANDLE = -7; /**< Invalid handle */
public static int ZKFP_ERR_CAPTURE = -8; /**< Failed to capture the image */
public static int ZKFP_ERR_EXTRACT_FP = -9; /**< Failed to extract the fingerprint template */
public static int ZKFP_ERR_ABSORT = -10; /**< Suspension */
public static int ZKFP_ERR_MEMORY_NOT_ENOUGH = -11; /**< Insufficient memory */
public static int ZKFP_ERR_BUSY = -12; /**< The fingerprint is being captured */
public static int ZKFP_ERR_ADD_FINGER = -13; /**< Failed to add the fingerprint template */
public static int ZKFP_ERR_DEL_FINGER = -14; /**< Failed to delete the fingerprint template */
public static int ZKFP_ERR_FAIL = -17; /**< Operation failed */
public static int ZKFP_ERR_CANCEL = -18; /**< Capture cancelled */
public static int ZKFP_ERR_VERIFY_FP = -20; /**< Fingerprint comparison failed */
public static int ZKFP_ERR_MERGE = -22; /**< Failed to combine registered fingerprint templates

```



```
*/  
public static int ZKFP_ERR_NOT_OPENED = -23; /**< Device not started */  
public static int ZKFP_ERR_NOT_INIT = -24; /**< Not initialized */  
public static int ZKFP_ERR_ALREADY_OPENED = -25; /**< Device started */
```