ZKFinger SDK

for Java

Version: 1.0

Date: May 2016

Revision Records

Date	Version	Description	Author
2016-05-21	1.0.0	Basic version	Chen Jianxing
2016-06-01	1.0.1	Added external image interfaces.	Chen Jianxing

Contents

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

Privacy Policy

Usage Description

As functions of the ZKFinger SDK software are constantly expanded, ZKFinger SDK documentations will be upgrading. Therefore, please read ZKFinger SDK documents carefully when using the ZKFinger SDK software. We apologize for any inconvenience caused by the preceding reasons. You can also contact the authors of the documentations. Thank you.

Company: ZKTech (Xiamen) Software

Address: Room 403-02, No.32, Guanri Road, Phase 2 of Xiamen Software Park

Telephone: 0592-5961369-8023

Website: www.zkteco.com

Mail: sdksupport@zkteco.com

1. Overview of ZKFinger SDK

ZKFinger SDK is a set of application programming interfaces (APIs) developed by ZKTeco for development engineers. It is capable of managing ZKTeco fingerprint readers in a unified manner. Development engineers can use functions in different classes to develop Java-based applications.

ZKFinger SDK supports the following functions:

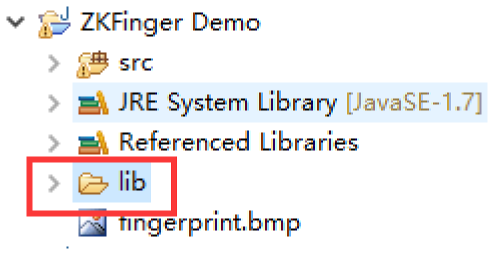
Fingerprint readers: ZKFinger SDK supports fingerprint capture and algorithm operations, including device initialization, device startup, device shutdown, 1:1 comparison, and 1:N comparison.

2. Development Environment Setup

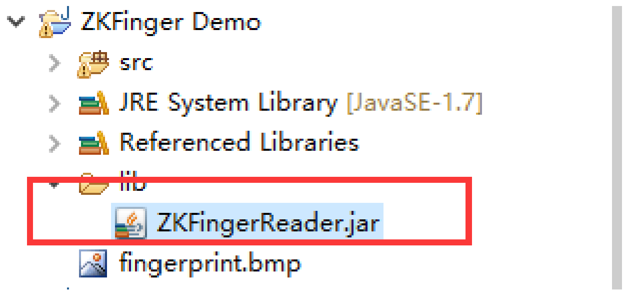
2.1 Importing ZKFingerReader.jar

Open the **SDK** folder and import **ZKFingerReader.jar** in the **java/lib** directory to the application development tool (the following uses Eclipse as an example).

Step 1: Create the **lib** directory in the directory of a project.



Step 2: Copy **ZKFingerReader.jar**, right-click the **lib** directory and choose **Paste** to copy **ZKFingerReader.jar** into the **lib** directory.



2.2 Deploying SDK

Install ZKFinger SDK 5.x/ZKOnline SDK 5.x.

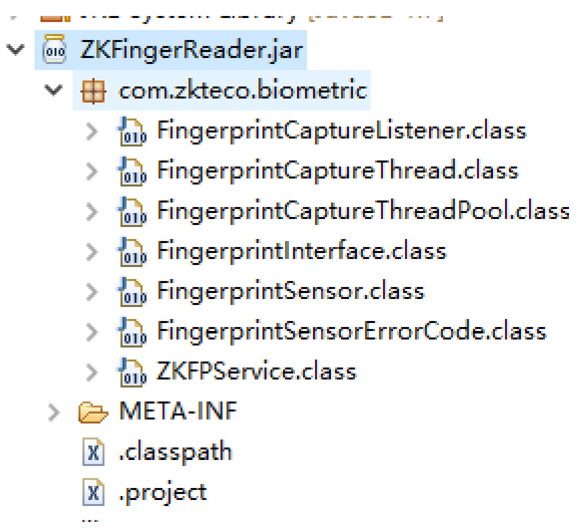
3 ZKFinger SDK

ZKFinger SDK abstracts function modules as classes. Users can call methods in classes to complete underlying hardware operations and processing of the fingerprint algorithm.

ZKFinger SDK includes the fingerprint reader class and algorithm handling class. The following table lists the types of the classes.

Class Name	Type
com.zkteco.biometric.FingerprintSensor	Fingerprint reader class, algorithm handling class
com.zkteco.biometric.FingerprintCaptureListener	Capture listening event class

The following figure shows the structure of the SDK package.



```
ZKFingerReader.jar
  com.zkteco.biometric
    FingerprintCaptureListener.class
    FingerprintCaptureThread.class
    FingerprintCaptureThreadPool.class
    FingerprintInterface.class
    FingerprintSensor.class
    FingerprintSensorErrorCode.class
    ZKFPService.class
  META-INF
  .classpath
  .project
  ...
```

3.1 FingerprintSensor.class

FingerprintSensor.class is a class for controlling fingerprint readers, which can be used to start and shut down a fingerprint reader, start fingerprint capture, and stop fingerprint capture.

3.1.1 getDeviceCount

[Function]

public int getDeviceCount ()

[Purpose]

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

[Parameter Description]

[Return Value]

Device count

[Note]

3.1.2 openDevice

[Function]

public int openDevice (int index)

[Purpose]

This function is used to connect to a device.

[Parameter Description]

index

Device index number. The value is determined based on the total number of connected fingerprint readers.

Example:

When a total of one fingerprint reader is connected, the index is 0.

When a total of two fingerprint readers are connected, the index is 0 or 1.

…

[Return Value]

0	Succeeded
Others	See the error code description.

[Note]

3.1.3 closeDevice

[Function]

public int closeDevice()

[Purpose]

This function is used to shut down a device.

[Parameter Description]

[Return Value]

0	Succeeded
Others	See the error code description.

[Note]

3.1.4 setFingerprintCaptureListener

[Function]

public void setFingerprintCaptureListener(FingerprintCaptureListener listener)

[Purpose]

This function is used to set a fingerprint capture listening event.

[Parameter Description]

Listener

Listening object

[Return Value]

[Note]

3.1.5 startCapture

[Function]

public void startCapture()

[Purpose]

This function is used to capture a fingerprint image.

[Parameter Description]

[Return Value]

[Note]

Images are captured asynchronously and the callback interface set by setFingerprintCaptureListener returns image templates. For details, see the demo.

3.1.6 stopCapture

[Function]

public void stopCapture ()

[Purpose]

This function is used to stop capturing fingerprint images asynchronously.

[Parameter Description]

[Return Value]

[Note]

Asynchronous capture of fingerprint images is stopped.

3.1.7 destroy

[Function]

public static void destroy()

[Purpose]

This function is used to destroy resources.

[Parameter Description]

[Return Value]

[Note]

3.1.8 getImageWidth

[Function]

public static int getImageWidth()

[Purpose]

This function is used to acquire the width of a fingerprint image.

[Parameter Description]

[Return Value]

Width of the fingerprint image

[Note]

3.1.9 getImageHeight

[Function]

public static int getImageHeight()

[Purpose]

This function is used to acquire the height of a fingerprint image.

[Parameter Description]

[Return Value]

Height of the fingerprint image

[Note]

3.1.10 getLastTempLen

[Function]

public static int getLastTempLen ()

[Purpose]

This function is used to acquire the data length of a fingerprint template.

[Parameter Description]

[Return Value]

Data length of the fingerprint template

[Note]

3.1.11 DBAdd

[Function]

public int DBAdd(int fid, byte[] regTemplate)

[Purpose]

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

[Parameter Description]

Fid

Fingerprint ID

regTemplate

Registered template

[Return Value]

0	Succeeded
Others	See the error code description.

[Note]

3.1.12 DBDel

[Function]

public int DBDel (int fid)

[Purpose]

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

[Parameter Description]

Fid

Fingerprint ID

[Return Value]

0	Succeeded
Others	See the error code description.

[Note]

3.1.13 DBCount

[Function]

public int DBCount ()

[Purpose]

This function is used to acquire the number of fingerprint images in the memory.

[Parameter Description]

[Return Value]

>=0	Fingerprint template count
<0	See the error code description.

[Note]

3.1.14 VerifyFPByID

[Function]

public int VerifyFPByID(int fid, byte[] template)

[Purpose]

This function is used to conduct 1:1 comparison based on the fingerprint ID.

[Parameter Description]

Fid

Fingerprint ID

Template

Fingerprint template

[Return Value]

The comparison score is returned.

<0 See the error code description.

[Note]

3.1.15 MatchFP

[Function]

public int MatchFP(byte[] temp1, byte[] temp2)

[Purpose]

This function is used to compare two fingerprint templates.

[Parameter Description]

temp1

Fingerprint template 1

temp2

Fingerprint template 2

[Return Value]

The comparison score is returned.

<0	See the error code description.

[Note]

3.1.16 IdentifyFP

[Function]

public int IdentifyFP(byte[] template, int[] fid, int[] socre)

[Purpose]

This function is used to conduct 1:N comparison.

[Parameter Description]

template

Fingerprint template

Fid

Returned fingerprint ID

Score

Returned comparison score

[Return Value]

0	Succeeded
Others	See the error code description.

[Note]

3.1.17 GenRegFPTemplate

[Function]

public int GenRegFPTemplate(byte[] temp1, byte[] temp2, byte[] temp3, byte[] regTemp, int[] regTempLen)

[Purpose]

This function is used to combine registered fingerprint templates.

[Parameter Description]

temp1

Preregistered template 1

temp2

Preregistered template 2

temp3

Preregistered template 3

regTemp

Returned registered template

regTempLen

Length of the returned registered template

[Return Value]

0	Succeeded
Others	See the error code description.

[Note]

3.1.18 ExtractFromImage

[Function]

public int ExtractFromImage(String filePath, int DPI, byte[] template, int[] size)

[Purpose]

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

[Parameter Description]

FilePath

Full path of a picture file

DPI

Image DPI

Template

Returned fingerprint template

Size

Length of the returned fingerprint template

[Return Value]

0	Succeeded
Others	See the error code description.

[Note]

Only the SDK of the standard version supports this function.

3.1.19 getDevSn

[Function]

public String getDevSn()

[Purpose]

This function is used to acquire the device SN.

[Parameter Description]

[Return Value]

Device SN

[Note]

3.1.20 setFakeFunOn

[Function]

public void setFakeFunOn(int FakeFunOn)

[Purpose]

This function is used to set the anti-fake function.

[Parameter Description]

FakeFunOn

0: disable; 1: enable

[Return Value]

[Note]

Currently, only the live20R supports this function.

3.1.21 getFakeFunOn

[Function]

public int getFakeFunOn()

[Purpose]

This function is used to acquire information about the anti-fake function.

[Parameter Description]

[Return Value]

0: disabled; 1: enabled

[Note]

Currently, only the live20R supports this function.

3.1.22 getFakeStatus

[Function]

public int getFakeStatus ()

[Purpose]

This function is used to set the anti-fake status.

[Parameter Description]

[Return Value]

If the lower five bits are all 1's, a fingerprint image is true (value&31 == 31).
If the lower five bits are not all 1's, a fingerprint image is false.

[Note]

Currently, only the live20R supports this function.

3.1.23 GetParameter

[Function]

public int GetParameter(int code, byte[] value, int[] len)

[Purpose]

This function is used to acquire a parameter.

[Parameter Description]

code

Parameter code (See the Appendixes.)

value

Parameter value

len

Parameter data length

[Return Value]

0	Succeeded
Others	See the error code description.

[Note]

[Example]

```
byte[] value = new byte[4];
int[] len = new int[1];
```

```
len[0] = 4;
int ret = GetParameter(1, value, len);        //image width
if (0 == ret)
{
        //convert byte array to int
}
```

3.1.24 SetParameter

[Function]

public int SetParameter(int code, byte[] value, int len)

[Purpose]

This function is used to set a parameter.

[Parameter Description]

code

Parameter code (See the Appendixes.)

value

Parameter value

len

Parameter data length

[Return Value]

0	Succeeded
Others	See the error code description.

[Note]

[Example]

```
byte[] value = new byte[4];
in len = 4;        //size of int
int FakeFunOn = 1;
value[0] = FakeFunOn & 0xFF;
value[1] = (FakeFunOn & 0xFF00) >> 8;
value[2] = (FakeFunOn & 0xFF0000) >> 16;
value[3] = (FakeFunOn & 0xFF000000) >> 24;
int ret = SetParameter(2002, value, len);    //set FakeFunOn
```

3.1.25 BlobToBase64

[Function]

public static String BlobToBase64(byte[] buf, int cbBuf)

[Purpose]

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

[Parameter Description]

buf

Blob data

cbBuf

Data length

[Return Value]

Base64 string

3.1.26 Base64ToBlob

[Function]

public static int Base64ToBlob(String strBase64, byte[] buf, int cbBuf)

[Purpose]

This function is used to convert Base64 string to byte[]

[Parameter Description]

strBase64

Base64 string

buf

Returned blob data

cbBuf

The length of buf

[Return Value]

The real length of returned data.

3.2 FingerprintCaptureListener.class

FingerprintCaptureListener.class is a fingerprint capture listening event class.

3.2.1 captureOK

[Function]

void captureOK(byte[] fpImage);

[Purpose]

This function is triggered when a fingerprint image is captured successfully.

[Parameter Description]

fpImage

Fingerprint image

[Return Value]

[Note]

3.2.2 captureError

[Function]

void captureError(int errCode)

[Purpose]

This function is triggered when a fingerprint image fails to be captured.

[Parameter Description]

See the error code description.

[Return Value]

None

[Note]

3.2.3 extractOK

[Function]

void extractOK(byte[] fpTemplate);

[Purpose]

This function is triggered when a fingerprint template is extracted successfully.

[Parameter Description]

fpTemplate

Fingerprint template

[Return Value]

None

[Note]

4 Appendixes

4.1 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
1102	Read-only	String	Product name
1103	Read-only	String	Device SN
101	Write-only (For devices other than the LIVE20R, a function needs to be called to turn off the light.)	Int	1 indicates that the white light blinks; 0 indicates that the white light is off.
102	Write-only (For devices other than the LIVE20R, a function needs to be called to turn off the light.)	Int	1 indicates that the green light blinks; 0 indicates that the green light is off.
103	Write-only (For devices other than the LIVE20R, a	Int	1 indicates that the red light blinks; 0

Parameter Code	**Property**	**Data Type**	**Description**
	function needs to be called to turn off the light.)		indicates that the red light is off.
104	Write-only (not supported by the LIVE20R)	Int	1 indicates that buzzing is started; 0 indicates that buzzing is turned off.

4.2 Error Code

Error Code	Description
0	Succeeded
1	Initialized
-1001	Failed
-1002	Failed to connect to the device
-1003	Device not connected
-1	Failed to initialize the algorithm library
-2	Failed to initialize the capture library
-3	No device connected
-4	Not supported by the interface
-5	Invalid parameter
-6	Failed to start the device
-7	Invalid handle
-8	Failed to capture the image
-9	Failed to extract the fingerprint template
-10	Suspension operation
-11	Insufficient memory
-12	The fingerprint is being captured

Error Code	Description
-13	Failed to add the fingerprint template to the memory
-14	Failed to add the fingerprint template
-17	Operation failed
-18	Capture cancelled
-20	Fingerprint comparison failed
-22	Failed to combine registered fingerprint templates
-24	Image processing failed