User Manual

ZKFinger Reader SDK for Android

Version: 1.0.3

Date: October 2019

Important Statement

Thank you for choosing the ZKFinger SDK of the ZKTeco. Please read the ZKFinger SDK user manual carefully before use so that you can master and use the ZKFinger SDK faster.

Unless authorized by our company, no group or individual shall take excerpts of or copy all or part of these instructions nor transmit the contents of these instructions by any means.

The products described in this manual may include software that is copyrighted by our company and its possible licensors. No one may copy, publish, edit, take excerpts of, decompile, decode, reverse-engineer, rent, transfer, sublicense, or otherwise infringe upon the software's copyright unless authorized by the copyright holder(s). This is subject to relevant laws prohibiting such restrictions.

As this product is regularly updated, we cannot guarantee exact consistency between the actual product and the written information in this manual. Our company claims no responsibility for any disputes that arise due to differences between the actual technical parameters and the descriptions in this document. The manual is subject to change without prior notice.

Table of Contents

1 ZKFinger SDK Introduction

The ZKFinger SDK is a set of application interfaces provided by the ZKTeco to the developer, and has the function of uniformly managing the central control fingerprint collector device module. Developers can use the functions in each class to develop applications that control Android devices.

The ZKFinger SDK includes the following features:

- **Fingerprint collector device:** mainly operates the fingerprint collector operation, such as initializing the device, opening the device, turning off the device, etc.
- **Fingerprint algorithm:** These functions mainly support 1:1, 1:N comparison and so on.

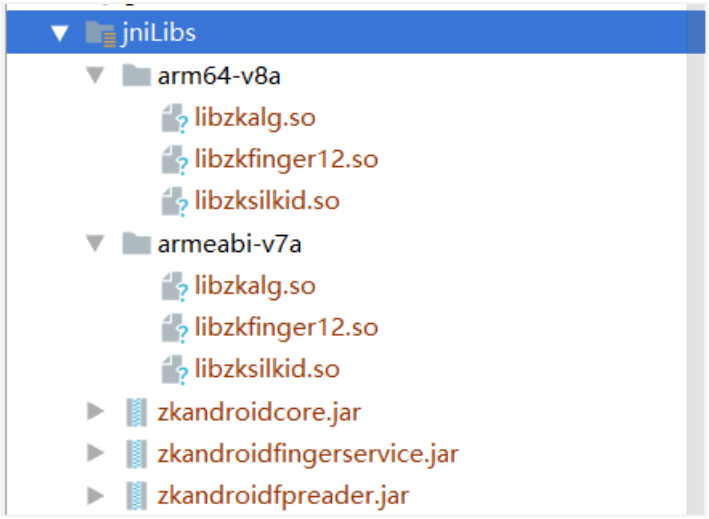
2 Building an Environment of Development

2.1 Import File

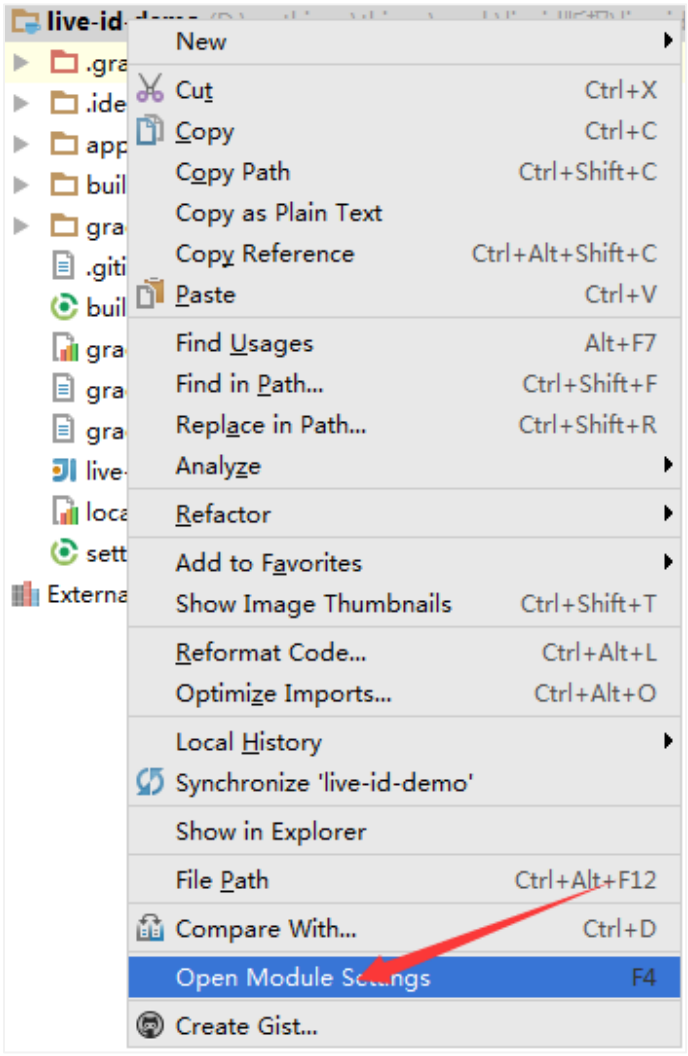
To import file of zkandroidfpreader.jar/zkandroidcore.jar/zkandroidfingerservice.jar

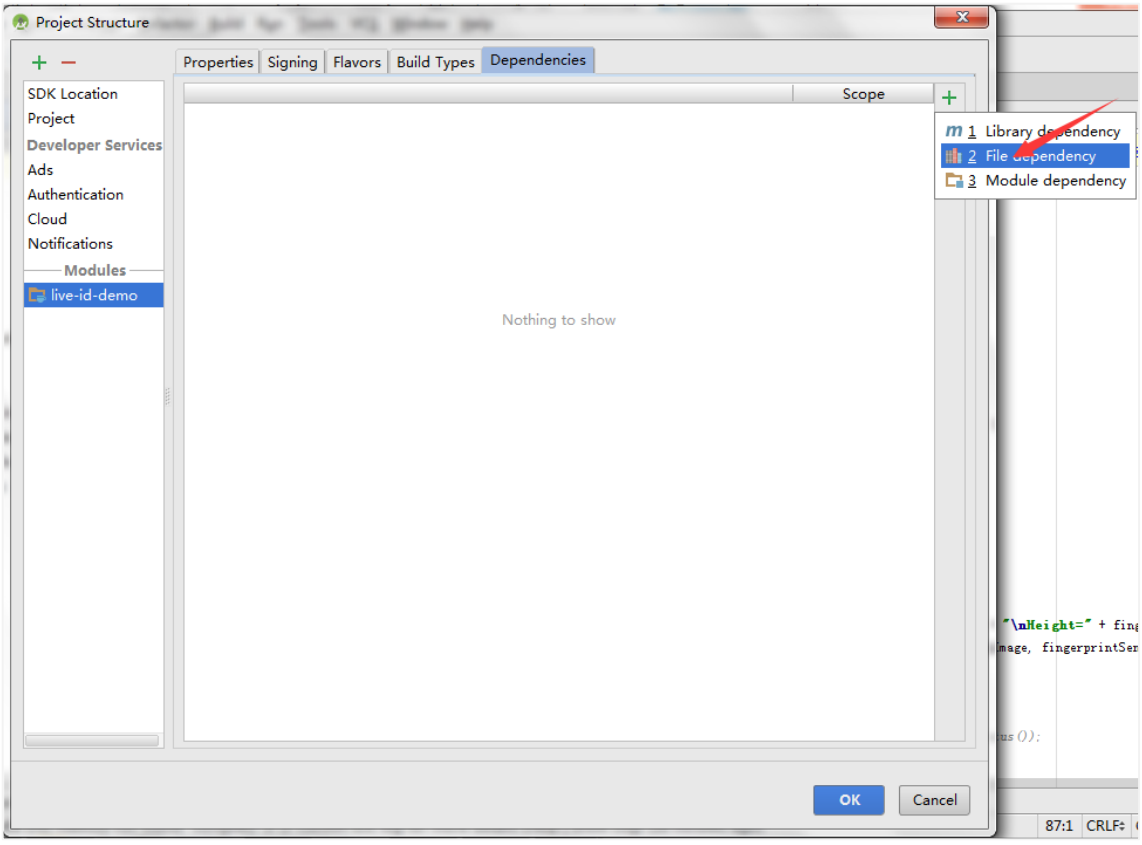
Open the ZKFinger SDK folder and import **zkandroidfpreader.jar/zkandroidcore.jar/zkandroidfingerservice.jar** from the **Device** directory into the application development tool (using Android Studio as an example).

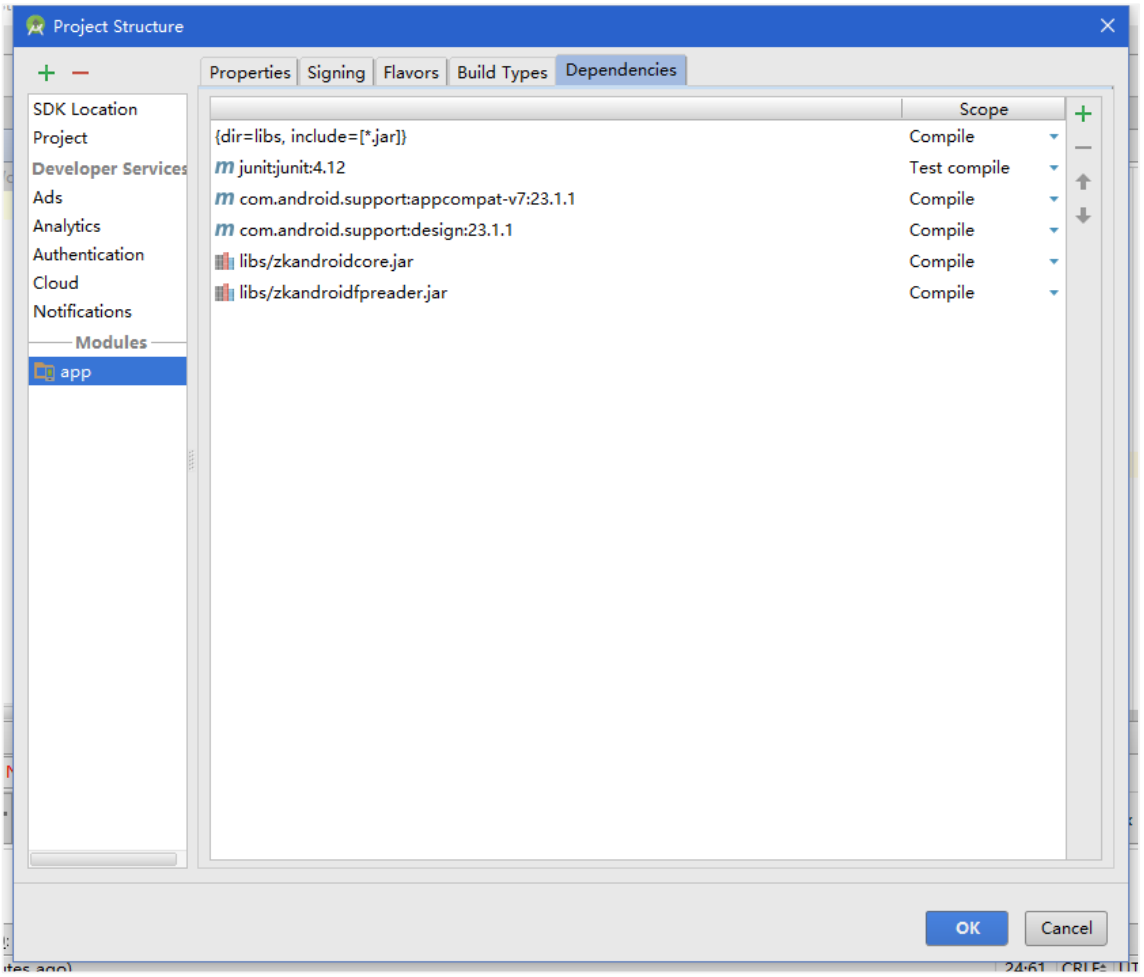
Step1: Copy **zkandroidfpreader.jar, zkandroidcore.jar and zkandroidfingerservice.jar** to **app/libs** directory.



Step 2: Choose engineering->Open Modules Setting->Dependenices->+ ->Filedependency, choose **zkandroidfpreader.jar/ zkandroidcore.jar** of **libs** folder ->click OK->Import **zkandroidfpreader.jar/ zkandroidcore.jar** successfully.


New
Cut Ctrl+X
Copy Ctrl+C
Copy Path Ctrl+Shift+C
Copy as Plain Text
Copy Reference Ctrl+Alt+Shift+C
Paste Ctrl+V
Find Usages Alt+F7
Find in Path... Ctrl+Shift+F
Replace in Path... Ctrl+Shift+R
Analyze
Refactor
Add to Favorites
Show Image Thumbnails Ctrl+Shift+T
Reformat Code... Ctrl+Alt+L
Optimize Imports... Ctrl+Alt+O
Local History
Synchronize 'live-id-demo'
Show in Explorer
File Path Ctrl+Alt+F12
Compare With... Ctrl+D
Open Module Settings F4
Create Gist...


Project Structure
Properties
Signing
Flavors
Build Types
Dependencies
SDK Location
Project
Developer Services
Ads
Authentication
Cloud
Notifications
Modules
live-id-demo
Scope
Nothing to show
1 Library dependency
2 File dependency
3 Module dependency
OK
Cancel



2.2 Algorithm library deployment

Copy the **zkfingerreader\libs directory** to the **libs** directory of the Android Studio project.

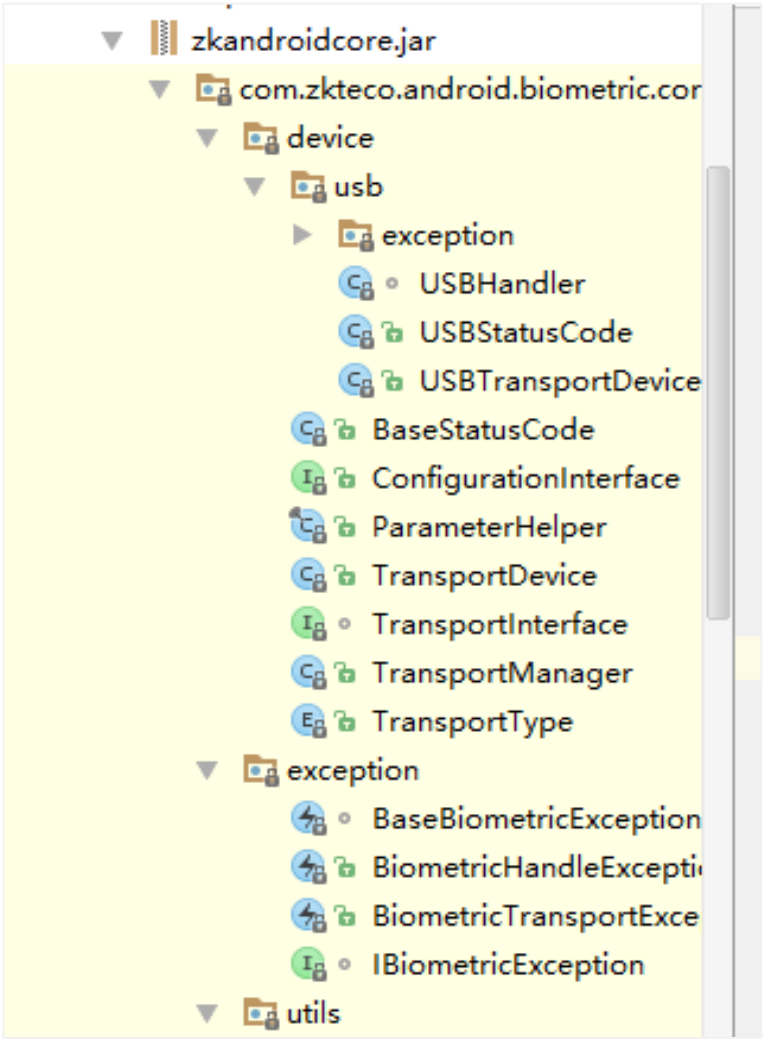
3 ZKFinger SDK

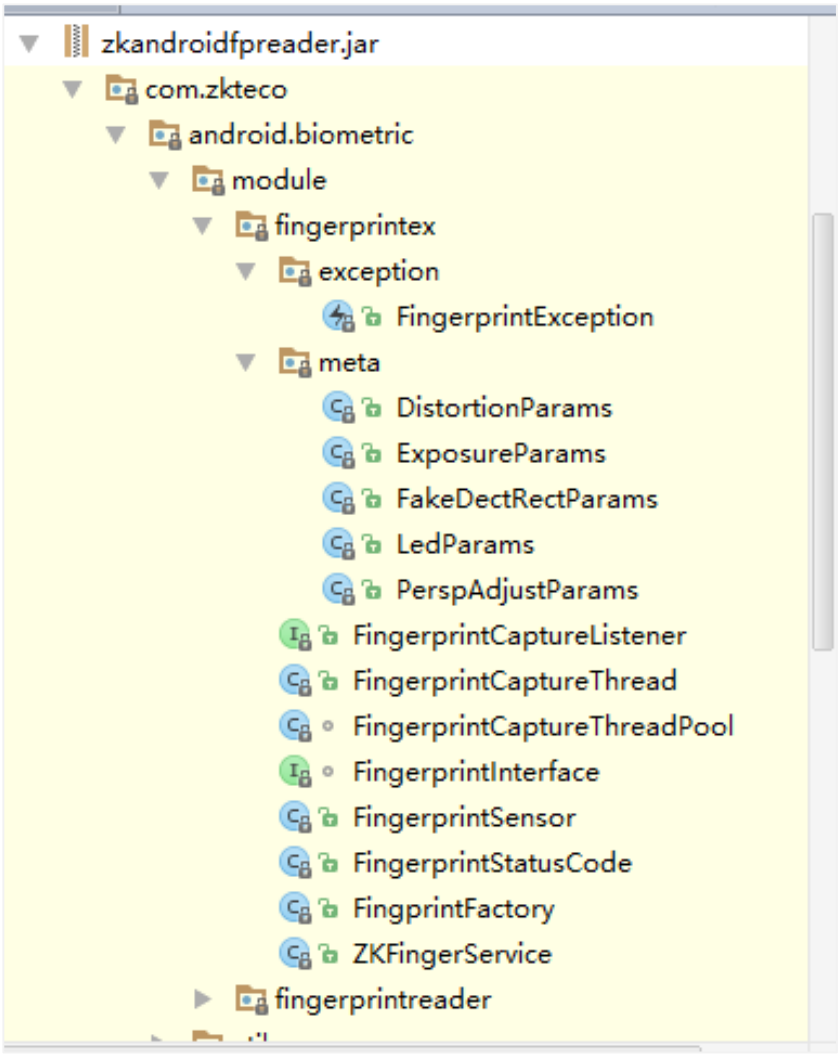
The ZKFinger SDK abstracts each functional module into a class. The user completes the operation of the underlying hardware device and the processing of the fingerprint algorithm by calling methods in the class. The ZKFinger SDK includes a fingerprint sensor device class, a control device class, a constant class, an exception handling class, and an algorithm processing class. The corresponding types of the types are as follows:

Class Name	Class
com.zkteco.android.biometric.module.fingerprintreader.FingerprintSensor	Fingerprint sensor device class
com.zkteco.android.biometric.module.fingerprintreader.FingerpringStatusCode	Error code

com.zkteco.android.biometric.module.fingerprintreader.exception.FingerprintException	Exception handling class
com.zkteco.android.biometric.core.utils.LogHelper	Log tool
com.zkteco.android.biometric.core.utils.ToolUtils	Auxiliary tool
com.zkteco.android.biometric.module.fingerprintreader.FingerprintCaptureListener	Capture listener event class
com.zkteco.android.biometric.module.fingerprintreader.ZKFingerService	Algorithm processing class

The SDK package structure is as follows:





3.1 FingerprintSensor.class

FingerprintSensor.class operates the fingerprint device class. Such as turning on the device, turning off the device, starting the acquisition, stopping the acquisition, and so on.

3.1.1 open

[Function]

public void open(int index)

[Features]

Connect device

[Parameter]

index

Device index number, which is determined by the total number of collectors connected.

For example:

When the total number of collectors is 1, the value of index is 0.

When the total number of collectors is 2, the value of index is 0 or 1.

…… ……

[Return value]

[Note]

The error code is thrown by an exception.

3.1.2 close

[Function]

public void close(int index)

[Features]

Close device

[Parameter]

index

Device index number, which is determined by the total number of collectors connected.

For example:

When the total number of collectors is 1, the value of index is 0.

When the total number of collectors is 2, the value of index is 0 or 1.

…… ……

[Return value]

[Note]

The error code is thrown by an exception.

3.1.3 setFingerprintfCaptureListener

[Function]

public void setFingerprintCaptureListener(int index, FingerprintCaptureListener listener)

[Features]

Set the fingerprint capture listener event

[Parameter description]

index

Device index number, which is determined by the total number of collectors connected.

For example:

When the total number of collectors is 1, the value of index is 0.

When the total number of collectors is 2, the value of index is 0 or 1.

…… ……

Listener

Listener

[Return value]

[Note]

3.1.4 startCapture

[Function]

public void startCapture(int index)

[Features]

Start taking images

[Parameter description]

index

Device index number, which is determined by the total number of collectors connected.

For example:

When the total number of collectors is 1, the value of index is 0.

When the total number of collectors is 2, the value of index is 0 or 1.

…… ……

[Return value]

[Note]

Asynchronous imaging, returning images, templates via the callback interface set by **setFingerprintCaptureListener** (see Demo for details).

3.1.5 stopCapture

[Function]

public void stopCapture (int index)

[Features]

Stop capturing (asynchronous).

[Parameter description]

index

Device index number, which is determined by the total number of collectors connected.

For example:

When the total number of collectors is 1, the value of index is 0.

When the total number of collectors is 2, the value of index is 0 or 1.

…… ……

[Return value]

[Note]

Stop asynchronous image capture.

3.1.6 destroy

[Function]

public void destroy(FingerprintSensor fingerprintSensor)

[Features]

Destroy resources.

[Parameter description]

fingerprintSensor

Device operation instance object

[Return value]

[Note]

3.1.7 getImageWidth

[Function]

public int getImageWidth()

[Features]

Get fingerprint image width

[Parameter description]

[Return value]

Fingerprint image width

[Note]

3.1.8 getImageHeight

[Function]

public int getImageHeight()

[Features]

Get fingerprint image high.

[Parameter description]

[Return value]

High fingerprint image

[Note]

3.1.9 getLastTempLen

[Function]

public int getLastTempLen ()

[Features]

Get the fingerprint template data length.

[Parameter description]

[Return value]

Fingerprint template data length

[Note]

See FingerprintCaptureListener. extractOK

3.1.10 setFakeFunOn

[Function]

public void setFakeFunOn(int fakeFunOn)

[Features]

Set anti-false switch.

[Parameter description]

fakeFunOn

0: means to turn off anti-false

1: means to turn on anti-false

[Return value]

[Note]

Only supports SILK20R

3.1.11 getFakeStatus

[Function]

public int getFakeStatus()

[Features]

Get the current fingerprint status

[Parameter description]

[Return value]

The lower 5 bits are all 1 for the true fingerprint: (value&0x1F) == 31

Other: Suspicious fingerprint

[Note]

Only SILK20R is supported, and setFakeFunOn(1)

3.1.12 rebootDevice

[Function]

public void rebootDevice(int index)

[Features]

Restart the fingerprint collector

[Parameter description]

index

Device index number, which is determined by the total number of collectors connected.

For example:

When the total number of collectors is 1, the value of index is 0.

When the total number of collectors is 2, the value of index is 0 or 1.

…… ……

[Return value]

[Note]

If the command cannot be sent to the collector after the communication is abnormal, you can call [openAndReboot] to open the device internally and issue a restart command to shut down the device.

3.1.13 openAndReboot

[Function]

public void openAndReboot(int index)

[Features]

Restart the fingerprint collector

[Parameter description]

index

Device index number, which is determined by the total number of collectors connected.

For example:

When the total number of collectors is 1, the value of index is 0.

When the total number of collectors is 2, the value of index is 0 or 1.

…… ……

[Return value]

[Note]

This operation involves opening the device, issuing a restart command, and shutting down the device.

3.1.14 getStrSerialNumber

[Function]

public String getStrSerialNumber()

[Features]

Get device SN

[Parameter description]

[Return value]

Collector device SN

[Note]

3.1.15 getFirmwareVersion

[Function]

public String getFirmwareVersion()

[Features]

Get the collector firmware version

[Parameter description]

[Return value]

Collector firmware version

[Note]

3.2 FingerprintCaptureListener.class

Fingerprint capture listener event class.

3.2.1 captureOK

[Function]

void captureOK(byte[] fpImage);

[Features]

Successful image acquisition

[Parameter]

fpImage

Fingerprint image

[Return value]

[Note]

3.2.2 captureError

[Function]

void captureError(FingerprintSensorException e)

[Features]

Failed to capture images

[Parameter]

e, Anomalous object (see 3.3)

[Return value]

No

[Note]

3.2.3 extractOK

[Function]

void extractOK(byte[] fpTemplate);

[Features]

Successful collection template

[Parameter]

fpTemplate

Fingerprint template

[Return value]

No

[Note]

3.2.4 extractError

[Function]

void extractError(int errno)

[Features]

Failed to extract template

[Parameter]

Please refer to "Appendix 1" for the error code description.

[Return value]

[Note]

3.3 ZKFingerService.class

Fingerprint algorithm processing class.

3.3.1 verify

[Function]

static public int verify(byte[] temp1, byte[] temp2)

[Features]

Compare two fingerprint templates

[Parameter]

temp1

First fingerprint template data array

temp2

Second fingerprint template data array

[Return value]

The matching score of the fingerprint template, the return value range: 0~100.

Recommendation: When the return value is greater than the set threshold or the default value (50), the two fingerprint templates can be considered to match successfully.

[Note]

Please refer to "Appendix 1" for the error code description.

3.3.2 verifyId

[Function]

static public int verifyId(byte[] temp, String id)

[Features]

Take the fingerprint template to matched with the fingerprint template of the specified id in the cache (that is 1:1 comparison), and the matching score is returned.

[Parameter]

temp

Fingerprint template data array

id

The target fingerprint template id being matched. The id type is a string, and the string length cannot exceed 20 bytes.

For example: String id = "ZKTeco_20_01".

[Return value]

The matching score of the fingerprint template, the return value range: 0~1000.

When the return value is greater than the set threshold size or the default value (50), it means that the two fingerprint templates match successfully.

3.3.3 identify

[Function]

static public int identify(byte[] temp, byte []idstr, int threshold, int count)

[Features]

Take the fingerprint template data to matched with the fingerprint template data in the cache (that is 1:N comparison), and the count results whose matching score is greater than threshold are returned. If the actual number of results is less than count, the actual number of results is returned. For example: count is set to 2, and the return value may be 0, 1, 2.

[Parameter]

temp

Fingerprint template data array

idstr

Returns an array of id numbers and scores after successful comparison

It is recommended that the array size be set to 50*count.

The multiple results are arranged in descending order of matching scores, each result including an id and a matching score, and the id and the matching score are separated by '/t', and the results are separated by '/n'.

For example: String resultStrs= new String(idstr);

If the return value of resultStrs is:"1001'\t'95'\n'1002'\t'85'\n'1003'\t'75", means returning three comparison results:

Result 1: id: 1001, score: 95;

Result 2: id: 1002, score: 85;

Result 3: id: 1003, score: 75;

For more detailed coding, please refer to the [Sample code].

threshold

Match the threshold, with the value ranging from 0 to 1000 (note: this value will override the previously set threshold).

The matching threshold is the boundary value to judge the similarity between two fingerprint feature templates. The higher the value is, the stricter the requirement for similarity is. Comparing the two fingerprint templates, the returned value is greater than the set threshold, indicating that the comparison is successful, otherwise the comparison fails. Usually, it can be flexibly adjusted according to the specific application.

Description of matching threshold

FRR	FAR	Recommended matching thresholds	
		1:N	1:1
High	Low	80	60
Medium	Medium	75	55
Low	High	70	50

count

The number of results expected to be returned is recommended to be set to 1.

For example:

Threshold is set to 70 and count is set to 3. If only 2 templates meet the matching threshold during fingerprint template matching, the function returns a value of 2.

[Return value]

Success Returns the actual number of comparison results. The return value may be less than **count**. **1**: Indicates that 1 fingerprint template was successfully matched. **0**: Indicates that no fingerprint template matches successfully.

Otherwise Returns an error code

[Note]

Please refer to "Appendix 1" for error code description.

[Sample code]

```
int ret = 0;
byte[] idstr = new byte[50];
ret = ZKFingerServer.identify(tempIdentify, idstr, 70, 1);
if( ret == 1 ){
String resultString = new String(idstr);
textView.setText("id: " + resultString.split("\t")[0] + ", score: " + resultString.split("\t")[1] );
}else{
textView.setText("Please try again!");
}
```

3.3.4 merge

[Function]

static public int merge(byte[] temp1, byte[] temp2, byte[] temp3, byte[] temp)

[Features]

Merge 3 fingerprint templates as registration templates

[Parameter]

temp1

Fingerprint template to be registered 1

temp2

Fingerprint template to be registered 2

temp3

Fingerprint template to be registered 3

temp

Output registered fingerprint template

[Return value]

>0 Registration template length

Otherwise Failure

[Note]

Please refer to "Appendix 1" for error code description.

3.3.5 save

[Function]

static public save (byte[] temp, String id)

[Features]

Save fingerprint template data to cache. The user saves the fingerprint data needed in his database to the cache by calling the save method.

[Note]

Calling save must ensure that the fingerprint algorithm library has been initialized (init) successfully.

[Parameter]

temp

Incoming Fingerprint Template Data Array

id

Fingerprint template id.

Id type is string, string length cannot exceed 20 bytes.

For example: String id = "ZKTeco_20_01".

[Return value]

0 Success

Otherwise Returns an error code

[Note]

Please refer to "Appendix 1" for error code description.

3.3.6 get

[Function]

static public int get (byte[] temp, String id)

[Features]

Gets the fingerprint template data of the specified id in the cache.

[Parameter]

temp

An array of fingerprint template data specifying the id. It is recommended to set the array size to 1024*3

to ensure sufficient storage space to store fingerprint template data.

id

Fingerprint template id to obtain.

Id type is string, string length cannot exceed 20 bytes.

For example: String id = "ZKTeco_20_01".

[Return value]

>0 Success, The value is the fingerprint template length

Otherwise Returns an error code

[Note]

Please refer to "Appendix 1" for error code description.

3.3.7 del

[Function]

static public int del(String id)

[Features]

Delete registered fingerprint template from memory

[Parameter]

id

Id of fingerprint template to be deleted.

Id type is string, string length cannot exceed 20 bytes.

For example: String id = "ZKTeco_20_01".

[Return value]

0 Success

Otherwise Returns an error code

[Note]

Please refer to "Appendix 1" for error code description.

3.3.8 clear

[Function]

static public int clear ()

[Features]

Empty all fingerprint data in cache

[Parameter]

No

[Return value]

0 Success

Otherwise Returns an error code

[Note]

Please refer to "Appendix 1" for error code description.

3.3.9 count

[Function]

static public int count()

[Features]

Gets the number of fingerprint templates currently stored in the cache

[Parameter]

No

[Return value]

The number of fingerprint templates currently stored in the cache

3.3.10 version

[Function]

static public void version(byte[] version)

[Features]

Gets the algorithm version number

[Parameter]

version

Returns the algorithm version number

[Return value]

3.3.11 setParameter

[Function]

static public int setParameter(int opCode, int value)

[Features]

Setting algorithm parameters

[Parameter]

opCode

Parameter code (1 means 1:1 threshold)

vaule

Parameter value (when opCode=1 means setting 1:1 threshold, default 50)

[Return value]

0 means success

Other means failure

[Remarks]

Please do not set it at will except OpCode=1 unless SDK developers specify otherwise.

3.3.12 getTemplateQuality

[Function]

static public int getTemplateQuality(byte[] template)

[Features]

Get template quality

[Parameter]

template

Fingerprint template

[Return value]

Template quality

3.3.13 getTemplateLength

[Function]

static public int getTemplateLength(byte[] template)

[Features]

Get template length

[Parameter]

template

Fingerprint template

[Return value]

Effective length of template data

3.3.14 convertTemplate

[Function]

static public int convertTemplate(byte[] inTemp, byte[] outTemp)

[Features]

Template conversion(12.0->10.0)

[Parameter]

inTemp

12.0 template

outTemp

10.0 template

[Return value]

> 0 Indicates that the conversion was successful, with a value of 10.0 template data length returned.

3.3.15 getFPLimitCount

[Function]

static public int getFPLimitCount()

[Features]

Get 1:N fingerprint capacity

[Parameter]

No

[Return value]

1:N fingerprint capacity

3.4 Code Samples (see Demo-Android Studio)

3.4.1 MainActivity.java

```
package  com.zkteco.live_id_demo;

import  android.graphics.Bitmap;
import  android.os.Bundle;
import  android.support.design.widget.FloatingActionButton;
import  android.support.design.widget.Snackbar;
import  android.support.v7.app.AppCompatActivity;
import  android.support.v7.widget.Toolbar;
import  android.util.Log;
import  android.view.View;
import  android.view.Menu;
import  android.view.MenuItem;
import  android.widget.ImageView;
import  android.widget.TextView;

import  com.zkteco.android.biometric.core.device.ParameterHelper;
import  com.zkteco.android.biometric.core.device.TransportType;
import  com.zkteco.android.biometric.core.utils.LogHelper;
import  com.zkteco.android.biometric.core.utils.ToolUtils;
import  com.zkteco.android.biometric.module.fingerprintreader.FingerprintCaptureListener;
```

```
import com.zkteco.android.biometric.module.fingerprintreader.FingerprintSensor;
import com.zkteco.android.biometric.module.fingerprintreader.FingprintFactory;
import com.zkteco.android.biometric.module.fingerprintreader.ZKFingerService;
import com.zkteco.android.biometric.module.fingerprintreader.exception.FingerprintException;

import java.io.File;
import java.io.FileNotFoundException;
import java.io.FileOutputStream;
import java.io.IOException;
import java.util.HashMap;
import java.util.Map;

public class MainActivity extends AppCompatActivity {
    private static final int VID = 6997;
    private static final int PID = 288;
    private TextView textView = null;
    private ImageView imageView = null;
    private boolean bstart = false;
    private boolean isRegister = false;
    private int uid = 1;
    private byte[][] regtemparray = new byte[3][2048];  //register template buffer array
    private int enrollidx = 0;
    private byte[] lastRegTemp = new byte[2048];

    private FingerprintSensor fingerprintSensor = null;
    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);
        Toolbar toolbar = (Toolbar) findViewById(R.id.toolbar);
        setSupportActionBar(toolbar);

        FloatingActionButton fab = (FloatingActionButton) findViewById(R.id.fab);
        fab.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View view) {
                Snackbar.make(view, "Replace with your own action", Snackbar.LENGTH_LONG)
                        .setAction("Action", null).show();
            }
        });

        textView = (TextView)findViewById(R.id.textView);
        imageView = (ImageView)findViewById(R.id.imageView);
```

```
            startFingerprintSensor();
        }


        private void startFingerprintSensor() {
            // Define output log level
            LogHelper.setLevel(Log.WARN);
            // Start fingerprint sensor
            Map fingerprintParams = new HashMap();
            //set vid
            fingerprintParams.put(ParameterHelper.PARAM_KEY_VID, VID);
            //set pid
            fingerprintParams.put(ParameterHelper.PARAM_KEY_PID, PID);
            fingerprintSensor = FingprintFactory.createFingerprintSensor(this, TransportType.USB, fingerprint
Params);
        }
        @Override
        public boolean onCreateOptionsMenu(Menu menu) {
            // Inflate the menu; this adds items to the action bar if it is present.
            getMenuInflater().inflate(R.menu.menu_main, menu);
            return true;
        }

        @Override
        public boolean onOptionsItemSelected(MenuItem item) {
            // Handle action bar item clicks here. The action bar will
            // automatically handle clicks on the Home/Up button, so long
            // as you specify a parent activity in AndroidManifest.xml.
            int id = item.getItemId();

            //noinspection SimplifiableIfStatement
            if (id == R.id.action_settings) {
                return true;
            }

            return super.onOptionsItemSelected(item);
        }

        public void saveBitmap(Bitmap bm) {
            File f = new File("/sdcard/fingerprint", "test.bmp");
            if (f.exists()) {
                f.delete();
            }
                FileOutputStream out = null;
```

```
            try {
                out = new FileOutputStream(f);
                bm.compress(Bitmap.CompressFormat.PNG, 90, out);
                out.flush();
                out.close();
            } catch (FileNotFoundException e) {
                e.printStackTrace();
            }catch (IOException e) {
                e.printStackTrace();
            }
    }
    public void OnBnBegin(View view) throws FingerprintException
    {
        try {
            if (bstart) return;
            fingerprintSensor.open(0);
            final FingerprintCaptureListener listener = new FingerprintCaptureListener() {
                @Override
                public void captureOK(final byte[] fpImage) {
                    runOnUiThread(new Runnable() {
                        @Override
                        public void run() {
                            if(null != fpImage)
                            {
                                ToolUtils.outputHexString(fpImage);
                                LogHelper.i("width=" + fingerprintSensor.getImageWidth() + "\nHe
ight=" + fingerprintSensor.getImageHeight());
                                Bitmap bitmapFp = ToolUtils.renderCroppedGreyScaleBitmap(fpIm
age, fingerprintSensor.getImageWidth(), fingerprintSensor.getImageHeight());
                                //saveBitmap(bitmapFp);
                                imageView.setImageBitmap(bitmapFp);
                            }
                            //textView.setText("FakeStatus:" + fingerprintSensor.getFakeStatus());
                        }
                    });
                }
                @Override
                public void captureError(FingerprintException e) {
                    final FingerprintException exp = e;
                    runOnUiThread(new Runnable() {
                        @Override
                        public void run() {
                            LogHelper.d("captureError   errno=" + exp.getErrorCode() +
                                    ",Internal error code: " + exp.getInternalErrorCode() + ",mess
```

```
age=" + exp.getMessage());
                                }
                            });
                        }
                        @Override
                        public void extractError(final int err)
                        {
                            runOnUiThread(new Runnable() {
                                @Override
                                public void run() {
                                    textView.setText("extract fail, errorcode:" + err);
                                }
                            });
                        }

                        @Override
                        public void extractOK(final byte[] fpTemplate)
                        {
                            final byte[] tmpBuffer = fpTemplate;
                            runOnUiThread(new Runnable() {
                                @Override
                                public void run() {
                                    if (isRegister) {
                                        byte[] bufids = new byte[256];
                                        int ret = ZKFingerService.identify(tmpBuffer, bufids, 70, 1);
                                        if (ret > 0)
                                        {
                                            String strRes[] = new String(bufids).split("\t");
                                            textView.setText("the finger already enroll by " + strRes[0] +
 ",cancel enroll");
                                            isRegister = false;
                                            enrollidx = 0;
                                            return;
                                        }
                                        if (enrollidx > 0 && ZKFingerService.verify(regtemparray[enrollidx-
1], tmpBuffer) <= 0)
                                        {
                                            textView.setText("please press the same finger 3 times for t
he enrollment");
                                            return;
                                        }
                                        System.arraycopy(tmpBuffer, 0, regtemparray[enrollidx], 0, 2048);
                                        enrollidx++;
                                        if (enrollidx == 3) {
```

```
                                        byte[] regTemp = new byte[2048];
                                        if (0 < (ret = ZKFingerService.merge(regtemparray[0], regtem
parray[1], regtemparray[2], regTemp))) {
                                            ZKFingerService.save(regTemp, "test" + uid++);
                                            System.arraycopy(regTemp, 0, lastRegTemp, 0, ret);
                                            textView.setText("enroll succ");
                                        } else {
                                            textView.setText("enroll fail");
                                        }
                                        isRegister = false;
                                    } else {
                                        textView.setText("You need to press the " + (3 - enrollidx)
+ "time fingerprint");
                                    }
                                } else {
                                    byte[] bufids = new byte[256];
                                    int ret = ZKFingerService.identify(tmpBuffer, bufids, 70, 1);
                                    if (ret > 0) {
                                        String strRes[] = new String(bufids).split("\t");
                                        textView.setText("identify succ, userid:" + strRes[0] + ", score:
" + strRes[1]);
                                    } else {
                                        textView.setText("identify fail");
                                    }
                                }
                            }
                        });
                    }


                };
                fingerprintSensor.setFingerprintCaptureListener(0, listener);
                fingerprintSensor.startCapture(0);
                bstart = true;
                textView.setText("start capture succ");
            }catch (FingerprintException e)
            {
                textView.setText("begin capture fail.errorcode:"+ e.getErrorCode() + "err message:" + e.get
Message() + "inner code:" + e.getInternalErrorCode());
            }
        }

        public void OnBnStop(View view) throws FingerprintException
        {
```

```
        try {
            if (bstart)
            {
                //stop capture
                fingerprintSensor.stopCapture(0);
                bstart = false;
                fingerprintSensor.close(0);
                textView.setText("stop capture succ");
            }
            else
            {
                textView.setText("already stop");
            }
        } catch (FingerprintException e) {
            textView.setText("stop fail, errno=" + e.getErrorCode() + "\nmessage=" + e.getMessage());
        }
    }

    public void OnBnEnroll(View view) {
        if (bstart) {
            isRegister = true;
            enrollidx = 0;
            textView.setText("You need to press the 3 time fingerprint");
        }
        else
        {
            textView.setText("please begin capture first");
        }
    }

    public void OnBnVerify(View view) {
        if (bstart) {
            isRegister = false;
            enrollidx = 0;
        }else {
            textView.setText("please begin capture first");
        }
    }
}
```

Appendix I

Error code	Remarks
-20001	Open device failed
-20002	Device shutdown failed
-20003	Get GPIO failed
-20004	Setting GPIO failed
-20005	Read EEPROM failed
-20006	Get image from USB failed
-20007	Detect USB image failed
-20008	Input buffer is not enough
-20009	Read data abnormal
-20010	Collect fingerprint failed
-20011	Decrypting image data failed
-20012	Start the acquisition thread failed
-20013	Stop collecting thread failed
-20014	Initialization of fingerprint device failed
-20015	Setting calibration parameters failed
ERR_NOT_FOUND -5000	No id fingerprint template found
ERR_PARAM -5002	Parameter error
ERR_TEMPLATE -5003	Fingerprint template error
ERR_METHOD -5004	Method error

ZK Building, Wuhe Road, Gangtou, Bantian, Buji Town,
Longgang District, Shenzhen China 518129

Tel: +86 755-89602345

Fax: +86 755-89602394

www.zkteco.com