

ZKFinger SDK

for Android

Version: 1.0.3

Date: Jul, 2020

修订记录

日期	版本	描述	作者
2016-04-13	1.0.0	基础版本	朱龙海
2016-04-27	1.0.1	增加部分接口(3.1.7~3.1.12)	陈建兴
2019-04-11	1.0.2	更新接口及异常回调	陈建兴
2020-7-29	1.0.3	公司名变更	陈建兴

感谢您使用熵基的ZKFinger SDK，在使用前请仔细阅读ZKFinger SDK概述，以便您能更快地掌握并使用ZKFinger SDK。

文档隐私权说明

非经过本公司书面同意，任何单位和个人不得擅自摘抄、复制本手册内容的部分或全部，并不得以任何形式传播。本手册中描述的产品中，可能包含我司及其可能存在的许可人享有版权的软件，除非获得相关权利人的许可，否则，任何人不得以任何形式对前述软件进行复制、分发、修改、摘录、反编译、反汇编、解密、反向工程、出租、转让等侵犯软件版权的行为。

文档使用说明

由于ZKFinger SDK软件功能不断扩充，ZKFinger SDK文档版本也会不断地升级，所以在使用ZKFinger SDK软件时，请详细阅读ZKFinger SDK文档内容。如有上诉原因给您造成的不便，敬请谅解，您也可以联系我们文档编写人，联系信息如下，谢谢！

1 ZKFinger SDK 概述

ZKFinger SDK是熵基提供给开发者的一套应用程序接口，具有统一管理熵基指纹采集器设备模块的功能。开发者可以使用各个类中函数，开发操作熵基Android设备的应用。

ZKFinger SDK包括以下功能：

指纹采集器设备：主要是操作指纹采集器操作，如初始化设备、打开设备，关闭设备等；

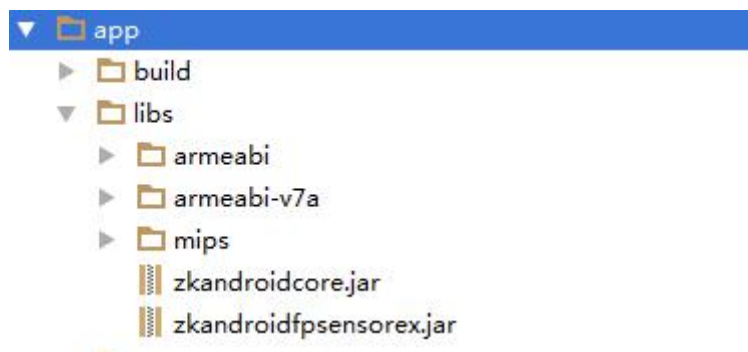
指纹算法：主要有这些功能支持 1:1,1:N 比对等。

2 开发环境搭建

2.1 导入 zkandroidfpreader.jar/zkandroidcore.jar/zkandroidfingerservice.jar

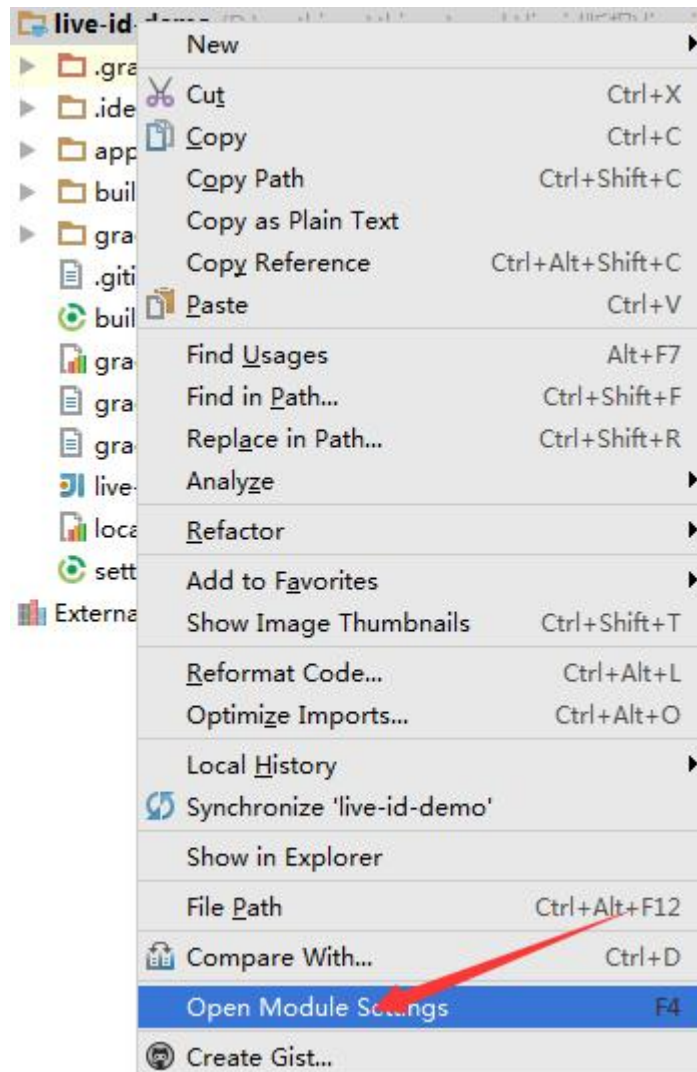
打开 SDK文件夹，将Device目录中的zkandroidfpreader.jar/zkandroidcore.jar/zkandroidfingerservice.jar导入到应用程序开发工具中（以Android Studio为例）

步骤 1：将zkandroidfpreader.jar、zkandroidcore.jar、zkandroidfingerservice.jar拷贝到app/libs目录；

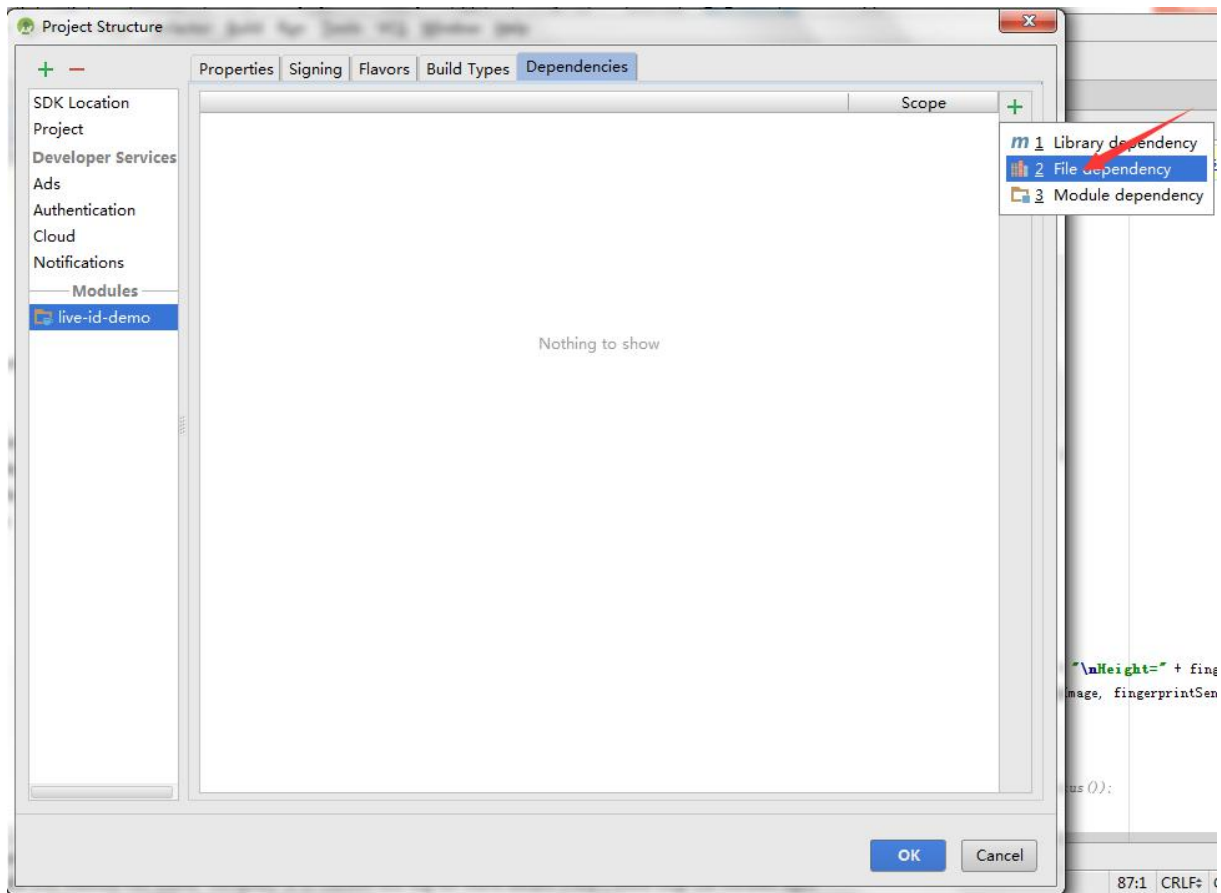


步骤 2：选择工程->Open Modules Setting->Dependencies->+ ->File dependency，选择 libs 文件夹里面的 zkandroidfpreader.jar/ zkandroidcore.jar/zkandroidfingerservice.jar ->点击 OK->导入 zkandroidfpreader.jar/ zkandroidcore.jar/zkandroidfingerservice.jar 成功。

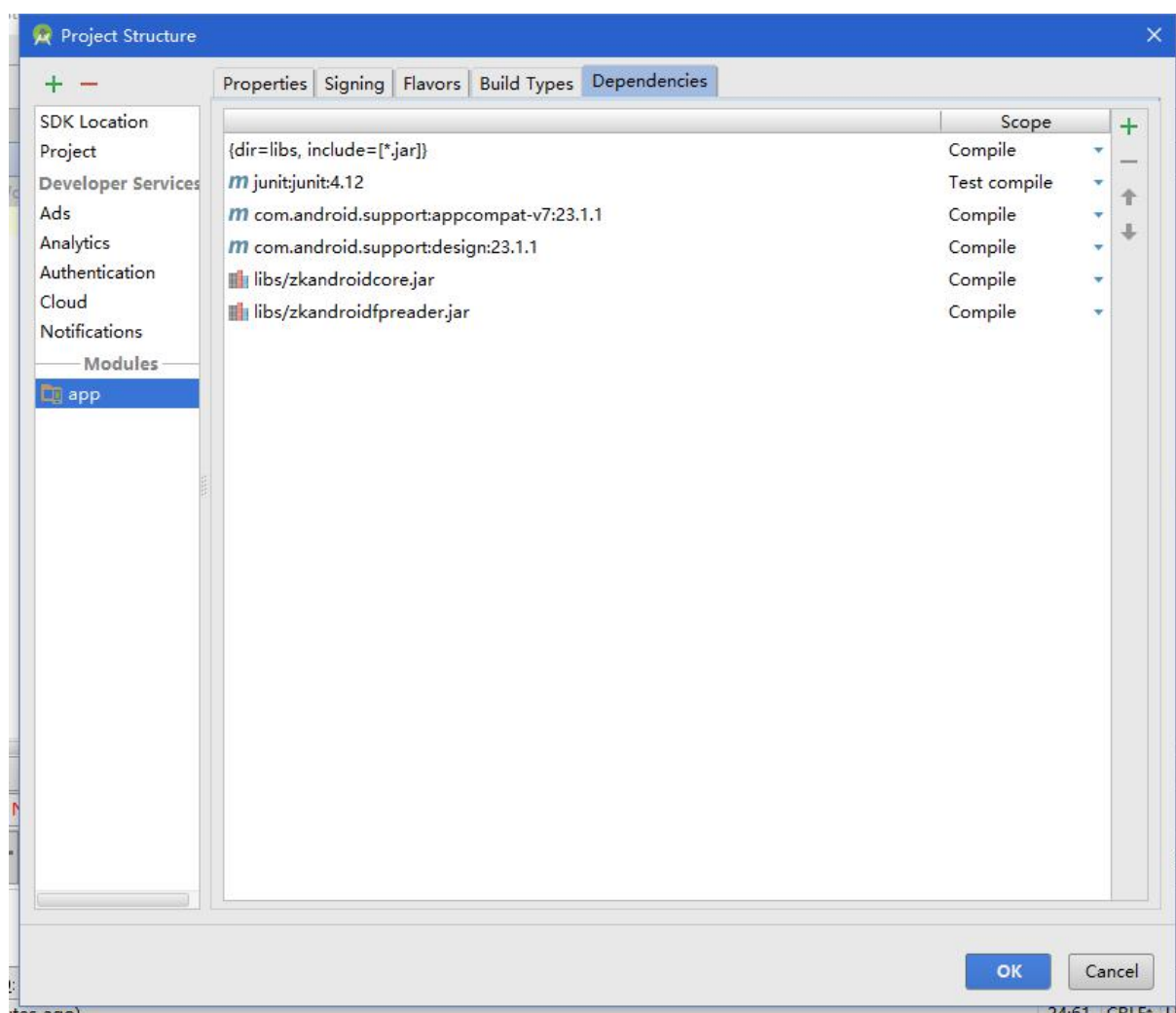
ZKfinger SDK for Android



ZKFinger SDK for Android



ZKFinger SDK for Android



2.2 算法库部署

zkfingerreader\libs目录拷贝到Android Studio 工程对应libs目录。

3 ZKFinger SDK

ZKFinger SDK 将各个功能模块抽象成类，用户通过调用类中方法完成对底层硬件设备的操作，以及对指纹算法的处理。

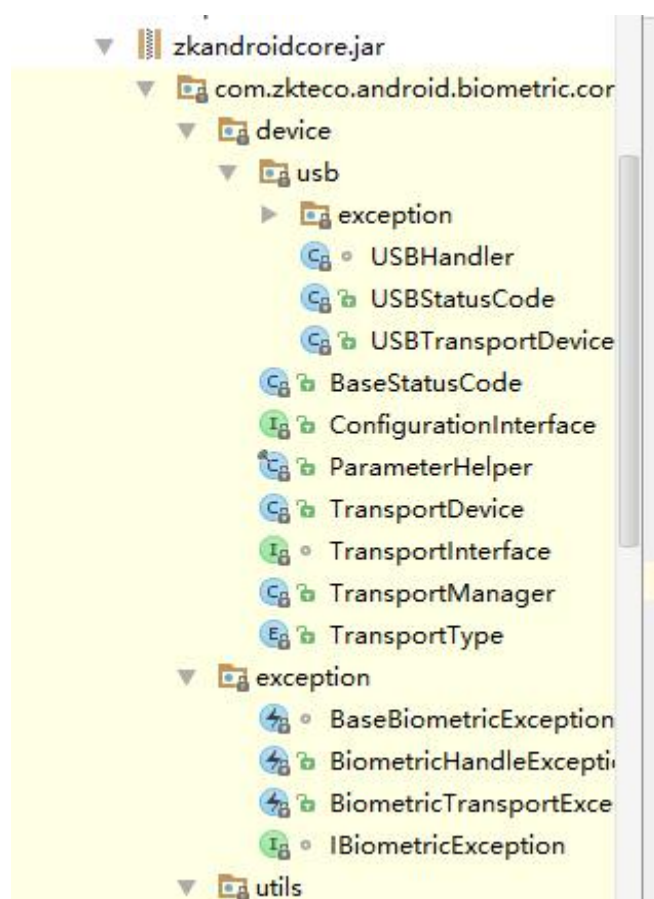
ZKFinger SDK包括指纹采集器设备类、控制设备类、常量类、异常处理类、算法处理类等。各类对应的类型如下图所示：

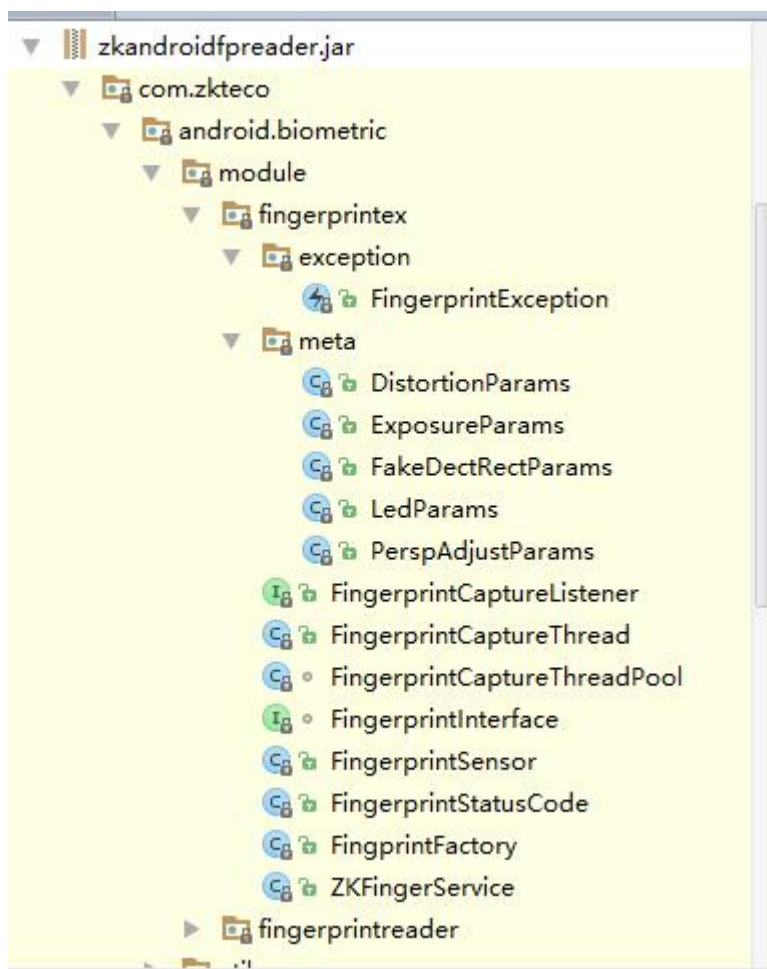
类名	类型
<code>com.zkteco.android.biometric.module.fingerprintre</code>	指静采集器设备类

ZKFinger SDK for Android

ader.FingerprintSensor	
com.zkteco.android.biometric.module.fingerprintrader.FingerprngStatusCode	错误码
com.zkteco.android.biometric.module.fingerprintrader.exception.FingerprintException	异常处理类
com.zkteco.android.biometric.core.utils.LogHelper	日志工具类
com.zkteco.android.biometric.core.utils.ToolUtils	辅助工具类
com.zkteco.android.biometric.module.fingerprintrader.FingerprintCaptureListener	采集监听事件类
com.zkteco.android.biometric.module.fingerprintrader.ZKFingerService	算法处理类
com.zkteco.android.biometric.FingerprintExceptionListener	异常监听事件接口

SDK包结构如下：





3.1 FingerprintSensor.class

FingerprintSensor.class 操作指指纹仪设备类。如打开设备、关闭设备、开始采集、停止采集等。

3.1.1 open

[函数]

public void open(int index)

[功能]

连接设备

[参数]

index

设备索引号，该值是接入采集器总数决定的。

例如:

当采集器总数为 1 时, 则index的值为 0;

当采集器总数为 2 时, index的值为 0 或 1;

.....

[返回值]

[注意]

错误码以异常抛出

3.1.2 close

[函数]

public void close(int index)

[功能]

关闭设备

[参数]

index

设备索引号, 该值是接入采集器总数决定的。

例如:

当采集器总数为 1 时, 则index的值为 0;

当采集器总数为 2 时, index的值为 0 或 1;

.....

[返回值]

[注意]

错误码以异常抛出

3.1.3 setFingerprintCaptureListener

[函数]

**public void setFingerprintCaptureListener(int index,
FingerprintCaptureListener listener)**

[功能]

设置指纹采集监听事件

[参数]

index

设备索引号，该值是接入采集器总数决定的。

例如：

当采集器总数为 1 时，则index的值为 0；

当采集器总数为 2 时，index的值为 0 或 1；

.....

Listener

监听对象

[返回值]

[注意]

3.1.4 startCapture

[函数]

public void startCapture(int index)

[功能]

开始取像

[参数说明]

index

设备索引号，该值是接入采集器总数决定的。

例如：

当采集器总数为 1 时，则index的值为 0；

当采集器总数为 2 时，index的值为 0 或 1；

.....

[返回值]

[注意]

异步取像，通过 *setFingerprintCaptureListener* 设置的回调接口返回图像，模板。(详见 Demo)

3.1.5 stopCapture

[函数]

public void stopCapture (int index)

[功能]

停止取像(异步)。

[参数说明]

index

设备索引号，该值是接入采集器总数决定的。

例如：

当采集器总数为 1 时，则index的值为 0；

当采集器总数为 2 时，index的值为 0 或 1；

.....

[返回值]

[注意]

停止异步取像。

3.1.6 destroy

[函数]

public static void destroy(FingerprintSensor fingerprintSensor)

[功能]

销毁资源。

[参数说明]

fingerprintSensor

设备操作实例对象

[返回值]

[注意]

3.1.7 getImageWidth

[函数]

public static int getImageWidth()

[功能]

获取指纹图像宽。

[参数说明]

[返回值]

指纹图像宽

[注意]

3.1.8 getImageHeight

[函数]

public static int getImageHeight()

[功能]

获取指纹图像高。

[参数说明]

[返回值]

指纹图像高

[注意]

3.1.9 getLastTempLen

[函数]

public static int getLastTempLen ()

[功能]

获取指纹模板数据长度。

[参数说明]

[返回值]

指纹模板数据长度

[注意]

见 **FingerprintCaptureListener.extractOK**

3.1.10 setFakeFunOn

[函数]

```
public static void setFakeFunOn(int fakeFunOn)
```

[功能]

设置防假开关。

[参数说明]

fakeFunOn

0:表示关闭防假

1:表示开启防假

[返回值]

[注意]

仅支持 **SILK20R**

3.1.11 getFakeFunOn

[函数]

```
public static int setFakeFunOn()
```

[功能]

设置防假开关。

[参数说明]

[返回值]

0:表示关闭防假

1:表示开启防假

[注意]

仅支持 **SILK20R**

3.1.12 getFakeStatus

[函数]

```
public static int getFakeStatus()
```

【功能】

获取当前指纹状态

【参数说明】

【返回值】

低 5 位全为 1 表示真指纹: $(value \& 0x1F) == 31$

其他:可疑指纹

【注意】

仅支持 **SILK20R**, 且 **setFakeFunOn(1)**

3.1.13 **getDeviceStatus**

【函数】

public static int getDeviceStatus(int index)

【功能】

获取当前设备状态

【参数说明】

index

设备索引号, 该值是接入采集器总数决定的。

例如:

当采集器总数为 1 时, 则index的值为 0;

当采集器总数为 2 时, index的值为 0 或 1;

.....

【返回值】

1 表示正常

2 表示 bulk 端点异常

其他 通讯异常

【注意】

3.1.14 **rebootDevice**

[函数]

public static void rebootDevice(int index)

[功能]

重启指纹采集器(连接状态下可用)

[参数说明]

index

设备索引号，该值是接入采集器总数决定的。

例如：

当采集器总数为 1 时，则index的值为 0；

当采集器总数为 2 时，index的值为 0 或 1；

.....

[返回值]

[注意]

3.1.15 **openAndReboot**

[函数]

public static void openAndReboot(int index)

[功能]

重启指纹采集器(open-reset-close)

[参数说明]

index

设备索引号，该值是接入采集器总数决定的。

例如：

当采集器总数为 1 时，则index的值为 0；

当采集器总数为 2 时，index的值为 0 或 1；

.....

[返回值]

[注意]

3.1.16 **getStrSerialNumber**

[函数]

public static String getStrSerialNumber()

[功能]

获取设备序列号

[参数说明]

[返回值]

设备序列号

[注意]

3.1.17 **getFirmwareVersion**

[函数]

public static String getFirmwareVersion()

[功能]

获取固件版本

[参数说明]

[返回值]

固件版本

[注意]

3.1.18 SetFingerprintExceptionListener

[函数]

```
public static void  
SetFingerprintExceptionListener(FingerprintExceptionListener listener)
```

[功能]

设置异常监听

[参数说明]

listener

见FingerprintExceptionListener 说明

[返回值]

[注意]

3.2 FingerprintCaptureListener.class

指纹采集监听事件类。

3.2.1 captureOK

[函数]

```
void captureOK(byte[] fpImage);
```

[功能]

采集图像成功

[参数]

fpImage

指纹图像

[返回值]

[注意]

3.2.2 captureError

[函数]

void captureError(FingerprintSensorException e)

[功能]

采集图像失败

[参数]

e,异常对象(见 3.3)

[返回值]

无

[注意]

3.2.3 extractOK

[函数]

void extractOK(byte[] fpTemplate);

[功能]

采集模板成功

[参数]

fpTemplate

指纹模板

[返回值]

无

[注意]

3.2.4 extractError

[函数]

void extractError(int errno)

[功能]

提取模板失败

[参数]

错误码说明请参考“[附录一](#)”

[返回值]

[注意]

3.3 ZKFingerService.class

指纹算法处理类

3.3.1 verify

[函数]

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

[功能]

比对两枚指纹模板

[参数]

temp1

第一个指纹模板数据数组

temp2

第二个指纹模板数据数组

[返回值]

指纹模板的匹配分数，返回值范围：0~100。

建议：返回值大于所设置的阈值或默认值(23)时，可认为两个指纹模板匹配成功。

[注意]

错误码说明请参考“[附录一](#)”

3.3.2 verifyId

[函数]

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

[功能]

ZKFinger SDK for Android

将指纹模板与缓存中指定id的指纹模板进行匹配（即1:1比对），返回匹配分数。

[参数]

temp

指纹模板数据数组

id

被匹配的目标指纹模板id。id类型为字符串，字符串长度不能超过20个字节。

例如：String id = "ZKTeco_20_01"。

[返回值]

指纹模板的匹配分数，返回值范围：0~100。

返回值大于所设置的阈值大小时或默认值(23)，表示两个指纹模板匹配成功。

3.3.3 identify

[函数]

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

[功能]

将指纹模板数据与缓存中的指纹模板数据进行匹配（即1:N比对），返回匹配分数大于threshold的count个结果，若实际结果个数不足count，则返回实际结果个数。例如：count设置为2，返回值可能是0,1,2。

[参数]

temp

指纹模板数据数组

idstr

返回比对成功后的id号和分数的数组。

建议数组大小设置为50*count。

多个结果按照匹配分数从高到低的顺序排列,每个结果包括id和匹配分数，id和匹配分数之间通过' /t' 分隔，结果之间通过' /n' 分隔。

例如：String resultStrs= new String(idstr);

ZKFinger SDK for Android

如resultStrs的返回值为：“1001’ \t’ 95’ \n’ 1002’ \t’ 85’ \n’ 1003’ \t’ 75”，则表示返回三个比对结果：

结果1： id: 1001, score: 95;

结果2： id: 1002, score: 85;

结果3： id: 1003, score: 75;

详细代码请查看[示例代码]。

threshold

匹配阈值，取值范围为0~100（注:此值会覆盖先前所设置的阈值）。

匹配阈值是判断两个指纹特征模板相似程度的分界值，匹配阈值定得越高，表示对相似度的要求越严。两个指纹模板进行比对，返回的数值大于设置的阈值，表示比对成功，否则比对失败。通常在具体应用时可以根据情况灵活调整。

匹配阈值说明

拒判率	误判率	匹配阈值	
		1:N	1:1
高	低	80	45
中	中	75	40
低	高	70	35

count

期望返回的结果个数，建议设置为1。

例如：

threshold 设置为 50，count 设置为 3。如果指纹模板比对过程中只有 2 个模板符合匹配阈值，则函数返回值为 2。

【返回值】

成功 返回实际的比对结果个数。返回值可能小于count。1：表示1个指纹模板匹配成功。0：表示没有指纹模板匹配成功。

否则 返回一个错误码

【注意】

错误码说明请参考“[附录一](#)”

[示例代码]

```
int ret = 0;
byte[] idstr = new byte[50];
ret = ZKFingerServer.identify(tempIdentify, idstr, 50, 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!");
}
```

3.3.4merge

[函数]

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

[功能]

合并 3 个指纹模板为登记模板

[参数]

temp1

待登记指纹模板 1

temp2

待登记指纹模板 2

temp3

待登记指纹模板 3

temp

输出登记指纹模板

[返回值]

>0 登记模板长度

其他 失败

[注意]

错误码说明请参考“[附录一](#)”

3.3.5 save

[函数]

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

[功能]

将指纹模板数据保存到缓存。用户将自己数据库中需要用到的指纹数据，通过调用**save**方法保存到缓存中。

[注意]

调用**save**须确保指纹算法库已初始化（init）成功。

[参数]

temp

传入的指纹模板数据数组

id

指纹模板id。

id类型为字符串，字符串长度不能超过20个字节。

例如：String id = “ZKTeco_20_01”。

[返回值]

0 成功

否则 返回一个错误码

[注意]

错误码说明请参考“[附录一](#)”

3.3.6 get

[函数]

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

[功能]

获取缓存中指定 id 的指纹模板数据。

[参数]

temp

指定 id 的指纹模板数据的数组。建议将该数组大小设置为 1024*3，以确保有足够的存储空间储存指纹模板数据。

id

需要获取的指纹模板id。

id类型为字符串，字符串长度不能超过20个字节。

例如：String id = “ZKTeco_20_01”。

[返回值]

>0 成功，值为指纹模板长度

否则 返回一个错误码

[注意]

错误码说明请参考“[附录一](#)”

3.3.7 del

[函数]

static public int del(String id)

[功能]

从内存中删除登记的指纹模板

[参数]

id

需要删除的指纹模板的id。

id类型为字符串，字符串长度不能超过20个字节。

例如：String id = “ZKTeco_20_01”。

[返回值]

0 成功
否则 返回一个错误码

[注意]

错误码说明请参考“[附录一](#)”

3.3.8 clear

[函数]

static public int clear ()

[功能]

清空缓存中的所有指纹数据

[参数]

无

[返回值]

0 成功
否则 返回一个错误码

[注意]

错误码说明请参考“[附录一](#)”

3.3.9 count

[函数]

static public int count()

[功能]

获取缓存中当前存储的指纹模板数

[参数]

无

[返回值]

当前缓存中保存的指纹模板数

3.4 FingerprintExceptionListener.class

通讯异常监听事件类。

3.2.1 captureOK

[函数]

void onDeviceException();

[功能]

通讯异常时触发

[参数]

[返回值]

[注意]

3.5 实例代码(详见 Demo-Android Studio)

3.5.1 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;
```

ZKFinger SDK for Android

```
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.FingerprintFactory;
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;
```

ZKFinger SDK for Android

```
@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_
H_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 = FingerprintFactory.createFingerprintSensor(this, Transpo
```

ZKFinger SDK for Android

```
rtType.USB, fingerprintParams);
    }

    @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();
        }
```

ZKFinger SDK for Android

```
        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 FingerprintCaptureList
ener() {
            @Override
            public void captureOK(final byte[] fpImage) {
                runOnUiThread(new Runnable() {
                    @Override
                    public void run() {
                        if (null != fpImage)
                        {
                            ToolUtils.outputHexString(fpImage);
                            LogHelper.i("width=" + fingerprintSensor.getImageW
idth() + "\nHeight=" + fingerprintSensor.getImageHeight());

                            Bitmap bitmapFp = ToolUtils.renderCroppedGreyScale
Bitmap(fpImage, fingerprintSensor.getImageWidth(), fingerprintSensor.getImageHeigh
t());

                            //saveBitmap(bitmapFp);
                            imageView.setImageBitmap(bitmapFp);
                        }
                        //textView.setText("FakeStatus:" + fingerprintSensor.g
etFakeStatus());
                    }
                });
            }
        });
    }
```

ZKFinger SDK for Android

```
    }

    @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.getInternalError
rCode() + ", message=" + 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];
```

ZKFinger SDK for Android

```
int ret = ZKFingerService.identify(tmpBuffer, bufi
ds, 55, 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(regtem
parray[enrollidx-1], tmpBuffer) <= 0)
{
    textView.setText("please press the same finger
3 times for the enrollment");
    return;
}
System.arraycopy(tmpBuffer, 0, regtempparray[enroll
idx], 0, 2048);

enrollidx++;
if (enrollidx == 3) {
    byte[] regTemp = new byte[2048];
    if (0 < (ret = ZKFingerService.merge(regtempar
ray[0], regtempparray[1], regtempparray[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;
}
```

ZKFinger SDK for Android

```
        } else {
            textView.setText("You need to press the " + (3
- enrollidx) + "time fingerprint");
        }
    } else {
        byte[] bufids = new byte[256];
        int ret = ZKFingerService.identify(tmpBuffer, bufi
ds, 55, 1);

        if (ret > 0) {
            String strRes[] = new String(bufids).split("\t
");

            textView.setText("identify succ, userid:" + str
rRes[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.getMessage() + "inner code:" + e.getInternalErrorCode());
}
}
```

ZKFinger SDK for Android

```
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) {
```

ZKFinger SDK for Android

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

附录一

错误码	备注
-20001	打开设备失败
-20002	关闭设备失败
-20003	获取 GPIO 失败
-20004	设置 GPIO 失败
-20005	读 EEPROM 失败
-20006	从 USB 获取图像失败
-20007	探测 USB 图像失败
-20008	输入缓存不够
-20009	读取数据异常
-20010	采集指纹失败
-20011	解密图像数据失败
-20012	启动采集线程失败
-20013	停止采集线程失败
-20014	初始化指纹设备失败
-20015	设置校正参数失败
ERR_NOT_FOUND -5000	没有找到制定 id 指纹模板

ZKFinger SDK for Android

ERR_PARAM -5002	参数错误
ERR_TEMPLATE -5003	指纹模板错误
ERR_METHOD -5004	方法错误