1 概述

1.1 目的

本文档针对说明第三方应用如何使用魅族手机的指纹服务。

1.2 架构

```
APP => FingerprintManager => FingerprintService => Fingerprint JNI
```

2 指纹API

2.1 open

函数原型：`public static FingerprintManager open()`
功能描述：创建`FingerprintManager`实例化对象，从而使用指纹服务。获取的实例化对象需要由调用者主动调用`release`接口将其释放。
参数描述：无
返回值：

- FingerprintManager：返回实例化对象表示成功，后续可通过该对象的方法进行其他操作；
- null：失败。

2.2 release

函数原型：`void release();`
功能说明：释放调用者通过`open`获取的实例化对象。当应用完成所需要的操作后，无需进行其他指纹识别操作，必须调用该接口进行释放，应与`open`成对出现。
参数说明：无
返回值：无

2.3 abort

函数原型：`void abort() ;`
功能说明：取消正在进行的指纹操作。
参数说明：无
返回值：无

2.4 setCaptureCallback

函数原型：`public void setCaptureCallback(CaptureCallback captureCallback) ;`
功能说明：设置图像捕获的回调函数。
参数说明：

- captureCallback：图像捕获回调函数，其接口如下

```
        public static interface CaptureCallback {
            /**
             * Called when the sensor is waiting for the user to touch the sensor with the finger.
             */
            void onWaitingForInput();
            /**
             * Called when the user has put down the finger and the image capture procedure has started.
             */
            void onInput();
            /**
             * Called when the image capture procedure has completed.
             */
            void onCaptureCompleted();

            void onCaptureFailed(int reason);
        }
```

返回值：无

2.5 startIdentify

函数原型：`public void startIdentify(IdentifyCallback identifyCallback, int[] ids) ;`
功能说明：进行指纹认证。
参数说明：

- identifyCallback：指纹认证的回调函数，应用需要自己实现，其接口定义如下

```
        public static interface IdentifyCallback {
            /**
             * Called when the identification/verification procedure has succeeded to find a match.
             * @param fingerId the identifier of the matched fignerprint.
             * @param updated set if the fingerprint data acquired during identification was used to improve
             * the biometric data record for this identity.
             */
            void onIdentified(int fingerId, boolean updated);
            /**
             * Called when the identification/verification procedure has failed to find a match.
             */
            void onNoMatch();
        }
```

- ids：已经注册的指纹ID数组，可通过getIds()获得。

返回值：无

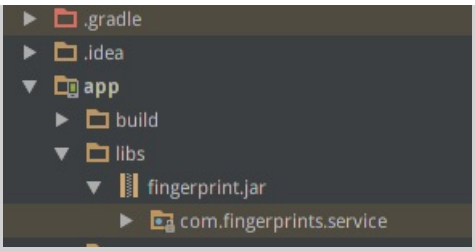
2.6 getIds

函数原型：`public int[] getIds() ;`
功能说明：获取已经注册的指纹ID列表。
参数说明：无
返回值：

- 数组：返回已经注册的指纹ID数组，最多支持5个指纹。
- null：获取指纹ID失败。

3 demo

以上所示接口仅使用与魅族MX5, POR5, MA01, MA01C, 魅蓝3S等android L机型, 对于魅族android M机型来说, 指纹接口与android原生接口保持一致,具体使用方式请参照android官方文档.
一般来讲, 第三方应用主要使用的是指纹识别的功能, 所以这里针对指纹识别给出demo.首先加入指纹的SDK fingerprint.jar 如下图所示:



然后在AndroidManifest里面配置使用指纹的权限如下：

```
<uses-permission android:name="com.fingerprints.service.ACCESS_FINGERPRINT_MANAGER" />
```

调用接口如下所示：

```
 private void initFingPrintManager() {
        if (mFM == null) {
            mFM = FingerprintManager.open();    //调用open方法得到FingerprintManager
        }
 }

public void startVerify() {
    Log.d(TAG, "startVerify");
     initFingPrintManager();    //得到FingerprintManager实例
    if (mFM.getIds() == null) {    //得到系统中已经录入的指纹个数
        Log.d(TAG, "no fingerprints enrolled");
        return;
    }
    if (mIdentifyCallback == null) {
        mIdentifyCallback = createIdentifyCallback();    //创建指纹认证回调函数
    }

    mFM.startIdentify(mIdentifyCallback, mFM.getIds());    //调用指纹认证接口
}

private IdentifyCallback createIdentifyCallback() {
    return new FingerprintManager.IdentifyCallback() {

        @Override
        public void onIdentified(int id, boolean updated) {    //认证成功
            Log.d(TAG, "onIdentified!, fingerId:" + id);
            mFM.release();    //认证成功后release, 需要注意的是在不使用指纹功能后必须要调用release, 也就是说open和release严格配对
                                    //否则会造成mBack不能使用, 因为只有调用release之后才能从指纹模式切换到back模式
        }

        @Override
        public void onNoMatch() {    //认证失败
            Log.d(TAG, "onNoMatch! ");
            startVerify();    //一次认证失败后重新再次发起认证
        }
    };
}
```