NFC 读卡接口

成都鱼住未来科技有限公司

二次开发：http://www.yzfuture.cn
技术支持：faq@yzfuture.cn
售后：sales@yzfuture.cn
电话：028-82880293

日期	版本	说明	作者
2021/07/01	V2.0.0	NFC&OTG 证件读取，支持身份证/港澳居民居住证/护照/AID	TygerZH
2021/10/13	V2.0.1	添加了初始化和反初始化操作	TygerZH

1. 概述

本 SDK 支持二代身份证、护照及 AID 的读取。

二代身份证接口添加了身份鉴权操作，只有当鉴权通过的用户才可以使用公司提供的解码服务器进行身份证解码。

护照及 AID 功能开通需要单独申请。

NFC 读卡用于支持 NFC 的安卓设备上。

OTG 读卡用于支持 USB 的安卓设备上(USB 口需要配套我公司专门证件读卡器)。

带解码授权的 USB 读卡器，直接插入就可使用，不用填 appkey/appsecret/appuserdata 三个参数

2. 接口概要

接口文件在 com\readTwoGeneralCard\OTGReadCardAPI.java 中。

3. 调用样例

```
OTGReadCardAPI      ReadCardAPI;
String              szAppKey= "xxxxxxxxxxxxxxxxx";// 按自己实际内容填写，参照《NFC 服务注册流程 V2》创建应用步骤获取，使用带解码授权的 USB 读卡器读卡此处可以为空
String              szAppSecret="xxxxxxxxxxxxxxxxx";// 按自己实际内容填写，参照《NFC 服务注册流程 V2》创建应用步骤获取，使用带解码授权的 USB 读卡器读卡此处可以为空
String              szAppUserData="xxxxxxxxxxxxxxxxx";// 按自己实际内容填写，参照《NFC 服务注册流程 V2》新增终端步骤获取，使用带解码授权的 USB 读卡器读卡此处可以为空
ReadCardAPI = new OTGReadCardAPI(getApplicationContext(), this);
if (ReadCardAPI != null)
{
    ReadCardAPI.initReadCard(m_szServerIP, m_nServerPort, szAppKey, szAppSecret,
szAppUserData);
}
if (离线读卡器)
{
    ReadCardAPI.setDeviceType(1);
}
else
{
    ReadCardAPI.setDeviceType(0);
}
if (ReadCardAPI.NfcReadCard(intent)) == 90)
{
        // 解码成功
}
```

```
else
{
        // 解码失败
}
```

4. 回调

在使用本 SDK 前必须实现 ActiveCallBack 接口中的相关函数，原型如下：

```
public interface ActiveCallBack{
    void readProgress(int npaogress);
    void setUserInfo(String sztxt);
    void upgradeInfo(String szupdate);
    }
```

- **void readProgress(int npaogress);**
  返回身份证读卡进度，一共 20 步。

- **void setUserInfo(String sztxt);**
  函数空实现即可，有时会返回调试信息。

- **void upgradeInfo(String szupdate);**
  SDK 升级提示，当系统检测到 SDK 需要升级时会通知 SDK，然后 SDK 会回调给上层应用，上层应用需要酌情处理。

5. 接口

- **OTGReadCardAPI**
  接口初始化操作。
  paramContext:android 的上下文
  cb:实现 ActiveCallBack 回调的类

- **setDeviceType**
  设置读卡器类型，默认为标准读卡器，当读卡器环境有变化时需要调用（比如原来是标准读卡器需要切换成离线读卡器时需要调用一次）。
  ndeviceType:读卡器类型（0-标准版　1-离线版）

- **GetVersion**
  获取当前版本号

- **initReadCard**
  初始化，最先调用，必须
  szIP:服务器 IP
  nPort:服务器端口
  szAppkey：NFC 应用的标识符，由系统自动生成，且唯一标识某一个 NFC 应用。

(我公司带永久授权的标准读卡器 OTG 读卡，此处可以为空，不用注册也可使用)(请参照文档《NFC 服务注册流程 V2》创建应用步骤进行申请)

szAppSecret:与 appkey 唯一对应的一个串号，与 appkey 配合使用，当用户觉察 appkey 信息泄露的时候，可在后台更新 secret,并在程序中使用新的 secret，此时旧的 secret 失效,即使其它人用旧 secret 也无法使用。(请参照文档《NFC 服务注册流程 V2》创建应用步骤进行申请)

szAppUserData：终端标识。用户自定义的终端唯一标识符，同一个 NFC 应用下不能重复。只有与终端标识绑定后的设备才有解码授权，否则系统会认为是非法设备,拒绝服务。**(离线读卡器,此处填写离线读卡器 SamId)**(请参照文档《NFC 服务注册流程 V2》新增终端步骤进行申请)

返回值：boolean 型成功或失败

- uninitReadCard

  反初始化操作。

  参数：

  无

  返回值：

  无

- NfcReadCard

  通过 NFC 读卡，同步操作，执行结束返回状态。

  intent:NFC 句柄,OTG 时填 null

  返回值：

  41 - 失败

  90 - 成功

- GetTwoCardInfo

  当读卡为身份证类型的时候，获取身份证详细信息。

  返回值：

```
public class TwoCardInfo {
    public String szTwoIdName; // 姓名
    public String szTwoIdSex; // 性别
    public String szTwoIdNation; // 民族
    public String szTwoIdBirthday; // 出生日期
    public String szTwoIdAddress; // 住址
    public String szTwoIdNo; // 身份证号码
    public String szTwoIdSignedDepartment; // 签发机关
    public String szTwoIdValidityPeriodBegin; // 有效期起始日期 YYYYMMDD
    public String szTwoIdValidityPeriodEnd; // 有效期截止日期 YYYYMMDD 有效期为长期时存储“长期”
    public String szTwoIdNewAddress; // 最新住址
    public byte[] arrTwoIdPhoto; // 照片信息
    public byte[] arrTwoIdFingerprint; // 指纹信息

    public String   szSNID;
```

```
    public String   szDNID;

    public String  szTwoOtherNO; // 通行证类号码
    public String  szTwoSignNum;      // 签发次数
    public String  szTwoRemark1;   // 预留区
    public String  szTwoType;     // 证件类型标识
    public String  szTwoRemark2;   // 预留区

}
```

- GetErrorInfo

  获取执行过程中的出错信息。

- GetAppKeyUseNum

  获取当前设备授权信息

```
public class clientAuthInfo {
    public int         bNumAuth; //  1 时 nNum 有效  0 时 szDate 有效
    public int nNum; // 设备自己可用次数，不包括应用自带次数
    public byte[] szDate = new byte[64]; // 设备包时到期时间
}
```

6. 错误信息

通过 GetErrorInfo 获取详细信息

7. 使用出错及解决方法

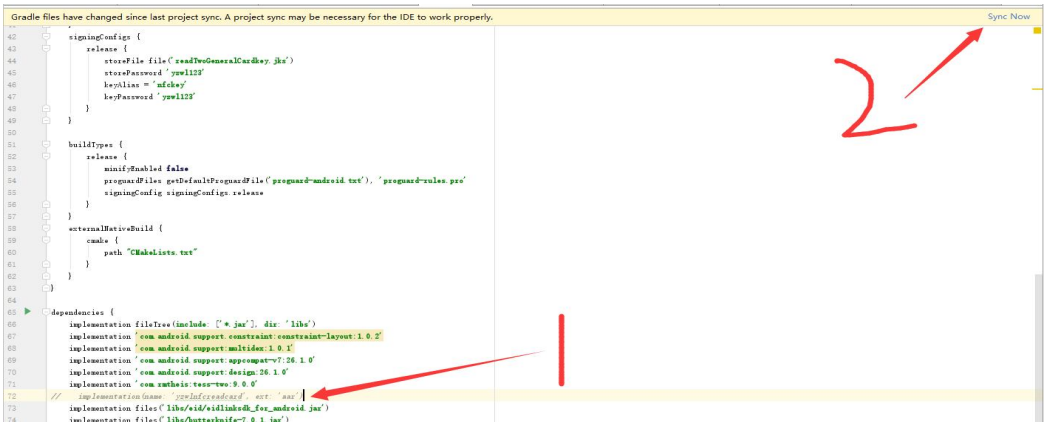
- **Gradle 版本不匹配**

  Demo 中用的 gradle 版本是 gradle-5.4.1-all.zip

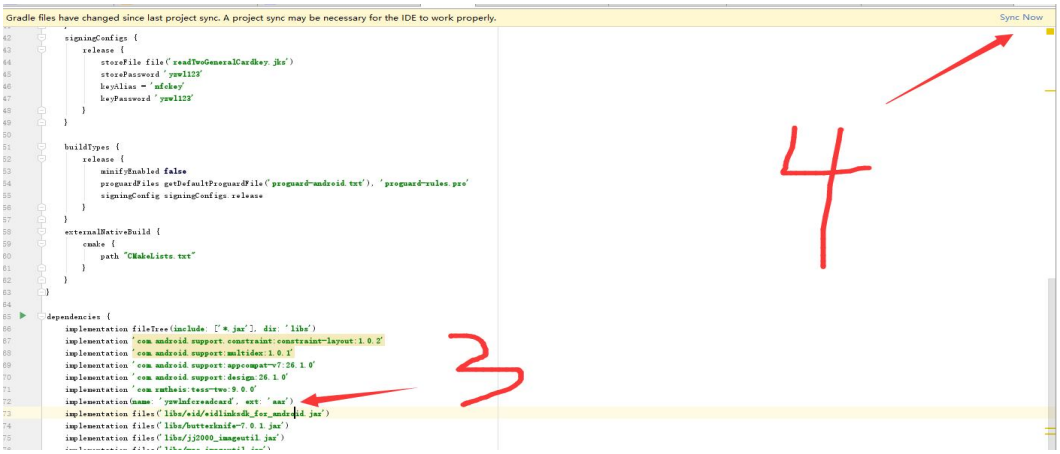
- **方法调用不起**

  可能是 aar 没有加载起，或者替换了 aar 后没有同步工程，可按照下列图示进行：

  先注释掉 build.gradle 文件中的 implementation(name: 'yzwlnfcreadcard', ext: 'aar')这一行，然后点”Sync Now”



去掉注释 build.gradle 文件中的 implementation(name: 'yzwlnfcreadcard', ext: 'aar')注释，然后点”Sync Now”



编译生成新的 apk 即可

- **FileProvider 冲突**

  由于 aar 包中需要操作本地文件，所以使用了 FileProvider，如果外面也用的话，有可能会报冲突，造成 App 读取文件自己的 FileProvider 失败，解决方法如下：

  添加如下代码：tools:replace="android:authorities"

  完整 provider 节内容如下：

```
<provider
android:name="android.support.v4.content.FileProvider"
android:authorities="${applicationId}.fileprovider"
android:exported="false"
android:grantUriPermissions="true"
tools:replace="android:authorities">
<meta-data
android:name="android.support.FILE_PROVIDER_PATHS"
android:resource="@xml/file_paths"
tools:replace="android:resource" />
```

```
</provider>
```

- **Apk 更新失败**

  请查看是否是写错了下载地址，导致你们的 apk 更新失败，我们 aar 对更新没有影响

- **Demo 源码编译出来不可用**

  Demo 只是提供调用样例，里面的 appkey 是用的我们统一的 appkey，这个 key 是没有试用权限的。如果需要调用的话，请用我们官网上的 demo 程序注册一个新的 appkey，然后把新 appkey 替换到代码中重新生成即可使用。

- **初始化失败**

  可能我们专用读卡器没有插好，建议重新插拔后再试。

- **OTG 读卡没有反应**

  可能是读卡器没有插好，建议重新插拔后再试。如果读卡器插入正常的话，在第一次 OTG 读卡的时候应该会弹出一个“是否允许应用访问 USB 设备”的请求框，需要点确定给读卡权限才可以用。

8. 微信小程序插件

插件地址：

https://mp.weixin.qq.com/wxopen/plugindevdoc?appid=wx0d82ce42bf0f4960&token=&lang=zh_CN

参数介绍：

appId:小程序 appId

appKey:《NFC 服务注册流程 V2》创建应用步骤中自动生成的字符串

appSecret:《NFC 服务注册流程 V2》创建应用步骤中自动生成的字符串，为防止信

息被盗用用户可手动更新

userData:《NFC 服务注册流程 V2》新增终端步骤中用户自定义的字符串

9. 微信小程序 demo 试用

微信扫码进入 demo 即可：

10. 微信小程序样例源码

源码地址：https://gitee.com/yz-future/yz-reader-demo.git

11. 安卓调用步骤

环境：android studio Chipmunk | 2021.2.1

- 工程配置

  a) 在项目目录（app\src\main\AndroidManifest.xml）添加以下权限

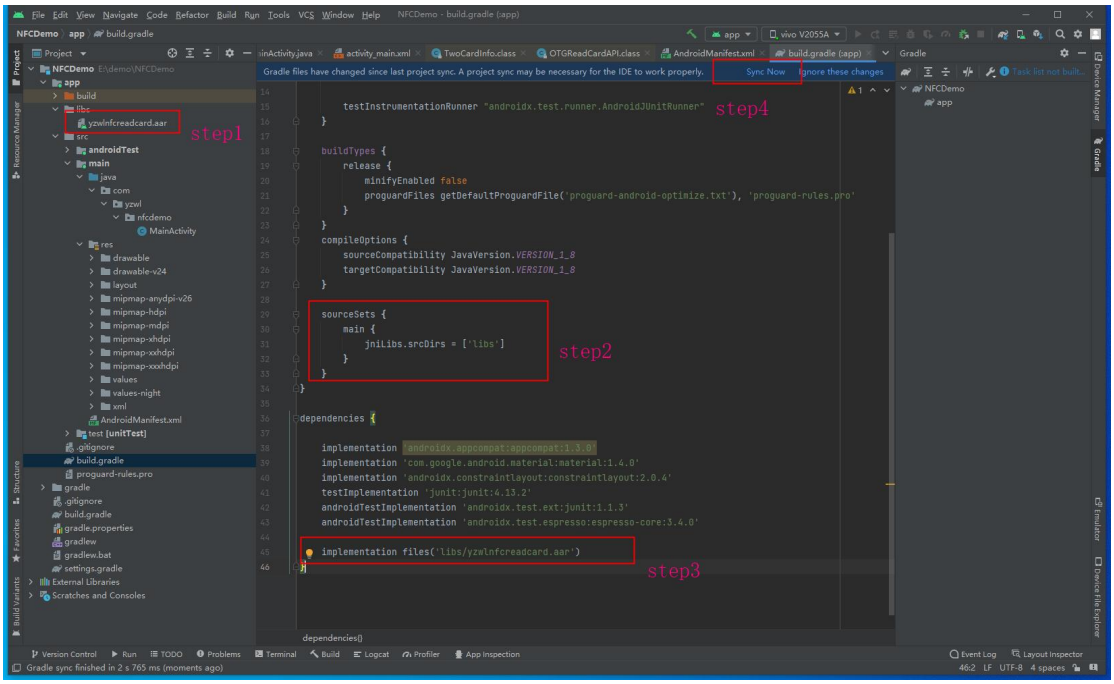
```
<!-- USB -->
<uses-feature
    android:name="android.hardware.usb.host"
    android:required="true" />
<uses-feature android:name="android.hardware.usb.accessory" />


<!-- NFC-->
<uses-feature
    android:name="android.hardware.nfc"
    android:required="true" />


<uses-permission android:name="android.permission.NFC" />
<uses-permission android:name="android.permission.INTERNET" />
<uses-permission android:name="android.permission.READ_PHONE_STATE" />
```

b) 在项目目录（app\build.gradle）添加以下片段，导入 SDK(minSdkVersion 大于等于 16)：

```
android {
    sourceSets {
        main {
            jniLibs.srcDirs = ['libs']
        }
    }
}
dependencies {
    implementation files('libs/yzwlnfcreadcard.aar')
}
```



- 工程项目代码集成步骤
  - a) SDK 初始化
    - i. 在 Activity 中定义读卡参数及相关对象

```
//注册获取
public final static String APP_KEY = "";
//注册获取
public final static String APP_SECRETE = "";
//绑定设备获取
public final static String USER_DATA = "";
public final static String NFC_SERVICE = "id.yzfuture.cn";
public final static int NFC_PORT = 443;
//读卡是耗时操作 放到子线程完成
private  ExecutorService nfcExecutorService;
//读卡 api
private OTGReadCardAPI mReadCardAPI;
//NFC 对象定义{***}
//OTG 对象定义{***}
```

    - ii. 定义读卡回调，获取读卡进度

```
protected ActiveCallBack readCallBack = new ActiveCallBack() {
    /**
     * 读卡进度回调
     * @param i 0 ~ 19
     * @param s
     */
    @Override
```

```
    public void readProgress(int i, String s) {
        Log.d(NFC_TAG, "" + i * 100 / 19 );
    }

    @Override
    public void setUserInfo(String s) {
    }

    @Override
    void upgradeInfo(String szupdate){
      // 升级提示
    }
};
```

iii. 在 Activity 的 onCreate 方法中初始化相关对象

```
@Override
protected void onCreate(Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);
    //初始化 NFC 对象
    //  {***}
    //初始化 USB 对象
    //  {***}

    //初始化相关对象
    try {
        mReadCardAPI = new OTGReadCardAPI(MainActivity.this, readCallBack);
        //初始化 API 参数
        mReadCardAPI.initReadCard(NFC_SERVICE, NFC_PORT, APP_KEY, APP_SECRETE, USER_DATA);
    }catch (Exception e){
        e.printStackTrace();
    }
}
```

iv. 定义读卡函数

```
private void startReadCard(final Intent intent) {
    if (mReadCardAPI == null)return;
    nfcExecutorService.execute(new Runnable() {
        @Override
        public void run() {
            try {
                int num =  mReadCardAPI.NfcReadCard(intent);
                if (num == 90) {
                    //读卡成功 获取读卡信息然后根据需求刷新 UI
                    TwoCardInfo cardInfo = mReadCardAPI.GetTwoCardInfo();
                    runOnUiThread(new Runnable() {
```

```
                    @Override
                    public void run() {
                        Log.e(NFC_TAG, "姓名：" + cardInfo.szTwoIdName);
                        Log.e(NFC_TAG, "性别：" + cardInfo.szTwoIdSex);
                        Log.e(NFC_TAG, "民族：" + cardInfo.szTwoIdNation);
                        Log.e(NFC_TAG, "出生日期：" + cardInfo.szTwoIdBirthday);
                        Log.e(NFC_TAG, "住址：" + cardInfo.szTwoIdAddress);
                        Log.e(NFC_TAG, "身份证号：" + cardInfo.szTwoIdNo);
                        Log.e(NFC_TAG, "签发机关：" + cardInfo.szTwoIdSignedDepartment);
                        Log.e(NFC_TAG, "有限期限：" + cardInfo.szTwoIdValidityPeriodBegin +
                                "-" + cardInfo.szTwoIdValidityPeriodEnd);
                    }
                });
            }else {
                //读卡失败 获取错误码和错误信息
                Log.e(NFC_TAG, "错误信息:" + mReadCardAPI.GetErrorCode() + mReadCardAPI.GetErrorInfo());
            }
        }catch (Exception e){
            Log.e(NFC_TAG, "身份证读卡失败");
            e.printStackTrace();
        }
    }
});
}
```

b) NFC 读卡

i. 在 Activity 中定义 NFC 相关对象

```
//NFC 对象定义
//nfc 控制器
private NfcAdapter mNfcAdapter;
public static final String NFC_TAG = "NFC_TAG";
```

ii. 在 Activity 的 onCreate 方法中初始化 NFC 对象

```
@Override
protected void onCreate(Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);
    // 初始化 NFC 对象
    nfcExecutorService = Executors.newSingleThreadExecutor();
    mNfcAdapter = NfcAdapter.getDefaultAdapter(MainActivity.this);
    if (mNfcAdapter == null){
        Log.e(NFC_TAG, "当前设备不支持 NFC");
        if (!mNfcAdapter.isEnabled()){
            Log.e(NFC_TAG, "NFC 未打开，提示用户去《设置》打开");
        }
    }
```

```
    //初始化 USB 对象
    //  {***}
    //初始化相关对象
    // {***}
}
```

iii. 在 Activity 的 onResume 方法以及 onPause 方法中，向系统申请（取消）处理 NFC 句柄

```
@Override
protected void onResume() {
    super.onResume();
    //向系统申请处理 NFC 句柄
    if (mNfcAdapter != null){
        try {
            Intent intent = new Intent(MainActivity.this, MainActivity.class);
            intent.addFlags(Intent.FLAG_ACTIVITY_SINGLE_TOP);
            PendingIntent pendingIntent = PendingIntent.getActivity(MainActivity.this,
                    0,
                    intent,
                    android.os.Build.VERSION.SDK_INT >= 31 ? PendingIntent.FLAG_MUTABLE : 0);
            IntentFilter tagDetected = new IntentFilter(NfcAdapter.ACTION_TECH_DISCOVERED);
            tagDetected.addCategory(Intent.CATEGORY_DEFAULT);
            String[][] techLists = new String[][]{new String[]{NfcB.class.getName()}, new String[]{NfcA.class.getName()}};
            mNfcAdapter.enableForegroundDispatch(MainActivity.this, pendingIntent, new IntentFilter[]{tagDetected}, techLists);
        }catch (Exception e){
            Log.e(NFC_TAG, "向系统申请处理 NFC 句柄失败");
            e.printStackTrace();
        }
    }
}


@Override
protected void onPause() {
    super.onPause();
    if (mNfcAdapter != null){
        mNfcAdapter.disableForegroundDispatch(MainActivity.this);
    }
}
```

iv. 在 Activity 的 onNewIntent 方法中，读取 Intent 句柄中的身份证信息

```
@Override
protected void onNewIntent(Intent intent) {
    super.onNewIntent(intent);
    //用户身份证靠近手机会来到这个回调 在这个地方读卡
```

```
    startReadCard(intent);
}
```

c) OTG 读卡

i. 在 Activity 中定义 NFC 相关对象

```
//OTG 对象定义
//用于获取 OTG 使用权限
private UsbManager mUsbManager;
```

ii. 在 Activity 的 onCreate 方法中初始化 USB 管理对象

```
@Override
protected void onCreate(Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);
    // 初始化 NFC 对象
    // {***}
    //初始化 USB 对象
    mUsbManager = (UsbManager) getSystemService(Context.USB_SERVICE);
    //初始化相关对象
    // {***}
}
```

iii. 定义一个 BroadcastReceiver 用于监听 USB 接入变化以及用户授权情况

```
private static final String ACTION_USB_PERMISSION = "com.android.usb.USB_PERMISSION";


private final BroadcastReceiver mUsbPreReceiver = new BroadcastReceiver() {
    public void onReceive(Context context, Intent intent) {
        String action = intent.getAction();
        UsbDevice device = (UsbDevice) intent.getParcelableExtra(UsbManager.EXTRA_DEVICE);
        if (TextUtils.isEmpty(action) || device == null || mUsbManager == null)return;
        switch (action){
            case ACTION_USB_DEVICE_ATTACHED:
                //用户插入读卡器以后会来到这个回调，以下代码用于向用户请求 USB 权限
                try {
                    PendingIntent pendingIntent = PendingIntent.getBroadcast(MainActivity.this,
                            0,
                            new Intent(ACTION_USB_PERMISSION),
                            android.os.Build.VERSION.SDK_INT >= 31 ? PendingIntent.FLAG_MUTABLE : 0);
                    mUsbManager.requestPermission(device, pendingIntent);
                }catch (Exception e){
                    e.printStackTrace();
                    Log.e(NFC_TAG, "请求 USB 授权失败");
                }
                break;
            case ACTION_USB_DEVICE_DETACHED:
                // 拔出 USB 根据自身需求处理
```

```
                break;
            case ACTION_USB_PERMISSION:
                //用户点击同意或拒绝以后来到这个回调
                if (intent.getBooleanExtra(UsbManager.EXTRA_PERMISSION_GRANTED, false)) {
                    //用户点击了同意
                    Log.e(NFC_TAG, "用户同意 USB 授权");
                }else{
                    //用户点击拒绝了
                    Log.e(NFC_TAG, "用户拒绝 USB 授权");
                }
                break;
            default:
                break;
        }
    }
};
```

iv. 在 Activity 的 onResume 方法以及 onPause 方法中，注册及取消 BroadcastReceiver 监听

```
@Override
protected void onResume() {
    super.onResume();

    //监听 USB OTG 接入的情况
    IntentFilter dFilter = new IntentFilter();
    //插入设备监听
    dFilter.addAction(ACTION_USB_DEVICE_ATTACHED);
    //拔出设备监听
    dFilter.addAction(ACTION_USB_DEVICE_DETACHED);
    //打开设备监听
    dFilter.addAction(ACTION_USB_PERMISSION);
    registerReceiver(mUsbPreReceiver, dFilter);
}

@Override
protected void onPause() {
    super.onPause();
    unregisterReceiver(mUsbPreReceiver);
}
```

v. 在需要使用读卡器读卡时主动调用 API 读取身份证信息

```
private void readOTG(){
    startReadCard(null);
}
```

12.