

# 鱼住 NFC 二代身份证读卡 云解码 SDK 开发文档

(Ver 3.05)

成都鱼住未来科技有限公司

二次开发: <http://www.aidoing.com.cn>

技术支持: [faq@yzfuture.cn](mailto:faq@yzfuture.cn)

售后: [sales@yzfuture.cn](mailto:sales@yzfuture.cn)

电话: 028-82880293    4001616832

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	ZH
2021/10/13	V2.0.1	添加了初始化和反初始化操作	ZH
2023/05/17	V2.0.2	添加了私有化服务支持、对接口名进行了统一规范	ZH

## 1. 概述

本 SDK 支持二代身份证、港澳居住证、外国永久居住证、国际电子护照及 AID 的读取。可有居于证件安全特性鉴别证件真伪，以及支持证件芯片内的信息读取（包括证件物理信息、证件文字信息、证件照片等数字信息）。

1) 支持的读卡设备硬件类型主要包括：

- 手机 NFC 射频读卡
- 手机 OTG 外接读卡器
- 鱼住 USB 接口的二代证读卡器设备（包括 YNR10X 系列在线设备、YNR20X 离线设备）
- 其他集成鱼住二代证读卡功能的设备、模块等。

2) 支持的读卡形式主要包括：

**在线读卡模式**，即使用中心读卡认证解码模式。该模式需要设备能联网通讯。

**离线读卡模式**，需要配置离线读卡器设备，该模式不需要设备联网通讯。

3) 解码认证系统部署方式：

**鱼住云服务**，使用鱼住分布式认证服务系统 SDK 接入鱼住云，开发者申请鱼住云开发者账号即可。

**私有部署服务**，由用户私有化部署解码认证服务系统，支持私网、专网部署服务系统，SDK 直连本地私有部署。

护照及 AID 功能开通需要单独申请。

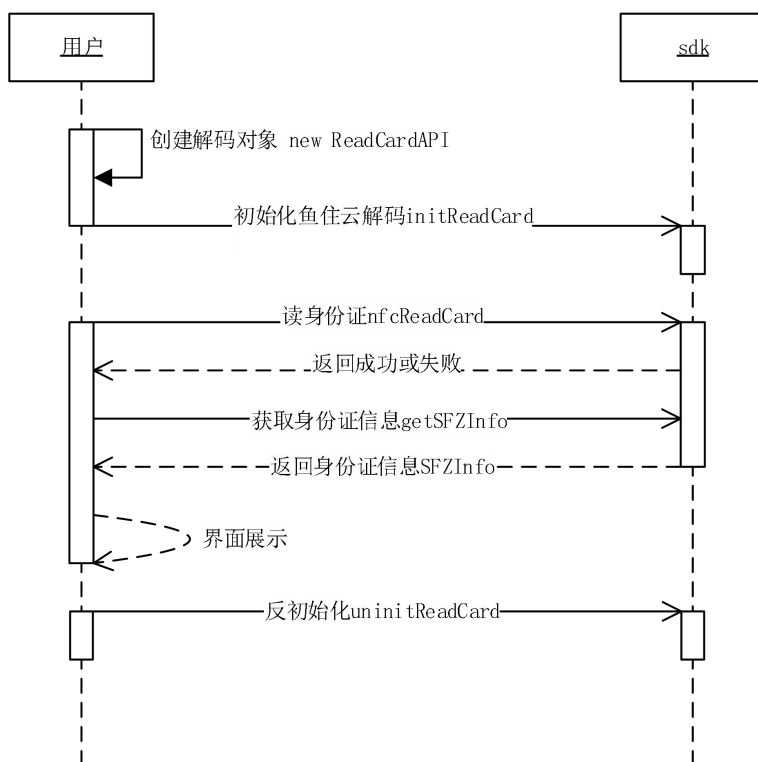
NFC 读卡用于支持 NFC 的安卓设备上。

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

SDK 接口文件目录路径 Com\idCardReader\ReadCardAPI.java



### 3. 接口

- 读卡接口类

#### ReadCardAPI

**paramContext**: android 的上下文

**cb**: 实现 ActiveCallBack 回调的类

- 初始化鱼住云解码

#### initReadCard

初始化鱼住云解码模式，云解码前需要调用

**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。

**szAppSecret**: 与 appkey 唯一对应的一个串号，与 appkey 配合使用。

**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。

**返回值**: boolean 型成功或失败

- 通过 NFC 进行鱼住云解码读卡

## ReadBCard

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

- 反初始化

## uninitReadCard

**返回值**:

无

- 获取身份证详细信息

## GetBCardInfo

**返回值**:

```
public class CardInfo {
    public int    type; // 0-身份证 1-港澳居住证 2-台湾居住证 3-外国人永久居留身份证 4-外国人永久居留身份证（旧版）
    public Sfz    sfz;  // 身份证/港澳居住证/台湾居住证
    public Foreigner  foreigner; // 外国人永久居留身份证
    public ForeignerOld  foreignerOld; // 外国人永久居留身份证（旧版）
    public String  sn; // 卡片 SN
    public String  serialNo; // 解码流水号
}

public class Sfz {
    public String  name;      // 姓名
    public String  sex;       // 性别
    public int     sexCode;   // 性别编码
    public String  nation;   // 民族
    public int     nationCode; // 民族编码
    public String  birthday;  // 出生日期
    public String  address;   // 地址
    public String  no;        // 证件号码
    public String  department; // 签发机关
    public String  beginTime; // 开始有效期
    public String  endTime;   // 结束有效期
    public String  otherNo;   // 其它号码(居住证用)
    public String  signNum;   // 签发次数(居住证用)
    public String  remark;    // 备用
    public String  type;      // 证件类型
    public String  remark1;   // 备用
}
```

```
        public byte[] photo;    // 头像 bmp 格式
        public byte[] finger;   // 指纹
    }

    public class Foreigner {
        public String name;    // 姓名
        public String sex;     // 性别
        public int sexCode;    // 性别编码
        public String nation;  // 民族
        public int nationCode; // 民族编码
        public String birthday; // 出生日期
        public String enName;   // 英文名
        public String no;       // 证件号码
        public String department; // 签发机关
        public String beginTime; // 开始有效期
        public String endTime;   // 结束有效期
        public String otherNo;   // 其它号码
        public String signNum;   // 签发次数
        public String country;   // 国家
        public String countryCode; // 国家编码
        public String type;      // 证件类型
        public String remark;     // 备用
        public byte[] photo;     // 头像 bmp
        public byte[] finger;    // 指纹
    }

    public class ForeignerOld {
        public String enName;    // 英文名
        public String sex;       // 性别
        public int sexCode;      // 性别编码
        public String no;        // 证件号码
        public String country;   // 国家
        public String countryCode; // 国家编码
        public String name;      // 姓名
        public String beginTime; // 开始有效期
        public String endTime;   // 结束有效期
        public String birthday;  // 出生日期
        public String version;   // 版本号
        public String department; // 签发机关
        public String type;      // 证件类型
        public String remark;     // 备用
        public byte[] photo;     // 头像 bmp
        public byte[] finger;    // 指纹
    }
}
```

- 获取执行过程中的出错信息

`getErrorInfo`

- 设置读卡器类型（非必须）

`setDeviceType`

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）

- 设置解码服务类型（非必须）

`setServerType`

默认为连接鱼住云服务器进行解码，当需要进行私有部署服务器解码时调此接口切换。

`ntype`: 服务器类型（0-鱼住云解码 1-私有本地部署 其它-鱼住云解码）

- 初始化私有解码（非必须）

`initReadCard_localServer`

初始化本地解码模式，本地解码前需要调用

`szIP`: 服务器 IP

`nPort`: 服务器端口

`szKey`: 与本地服务器通讯用的密钥，需与本地服务后台配置相同，避免其它人盗用服务。

返回值: boolean 型成功或失败

- 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);
}
```

返回身份证读卡进度

`void readProgress(int npaogress, String szinfo);`

返回身份证读卡进度，一共 20 步。

返回调试信息

`void setUserInfo(String sztxt);`

函数空实现即可，有时会返回调试信息。

## 4. 错误信息

0	成功
-1	未知错误
-2	参数无效
-3	格式出错
-4	获取卡号失败
-5	密钥失败
-100~-999	读芯片数据出错
-1000~-1999	网络发送出错
-2000~-19999	与卡片交互出错
-20000~-25000	设备鉴权出错
-25001~-29999	数据包解析出错
-30000~-39999	读卡设备异常或卡片通讯异常
-40000~-49999	身份证信息异常
-89999 以下	解码初始化没有完成

## 5. 调用样例

参看解码 demo 里面的 IDCardScannerActivity 类

请修改 IDCardScannerActivity 中的以下几行：

```
private static final String SPKEY_APPKEY = ""; // remark:请修改为你们在
鱼住云平台上申请到的 appkey";
private static final String SPKEY_SECRET = ""; // remark:请修改为你们在
鱼住云平台上申请到的 appsecret";
private static final String SPKEY_USERDATA = ""; // remark:请修改为你们
在鱼住云平台上申请到的 appuserData";
```

然后直接编译完成就可以调用鱼住云平台了。

如果需要调用本地私有部署的服务器请修改以下内容

```
private static final String m_szLocalIP = "192.168.15.181"; // remark:
本地服务器地址，请按实际修改
private static final int m_nLocalPort = 32458; // remark:本地服务器端
口，请按实际修改
private String m_szLocalKey = ""; // remark:请修改为你们实
际的 key，这个 key 是在服务器管理界面上配置的 key，为了避免其它客户不经论证偷用解
码功能
```

编译完成就可以直接调用本地解码平台了（前提是有本地解码服务器）

```
ReadCardAPI readCard;
String szAppKey= "xxxxxxxxxxxxxxxx"; // 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= "xxxxxxxxxxxxxxxx"; // 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= "xxxxxxxxxxxxxxxx"; // 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

readCard = new ReadCardAPI(getApplicationContext(), this);
```



```

if (readCard != null)
{
    if (readCard.initReadCard(m_szServerIP, m_nServerPort, szAppKey,
szAppSecret, szAppUserData))
    {
        if (ReadCardAPI.ReadBCard(intent)) == 90 {} // 解码成功
        else {} // 解码失败
    }
}

```



## 6. 使用出错及解决方法

- Gradle 版本不匹配

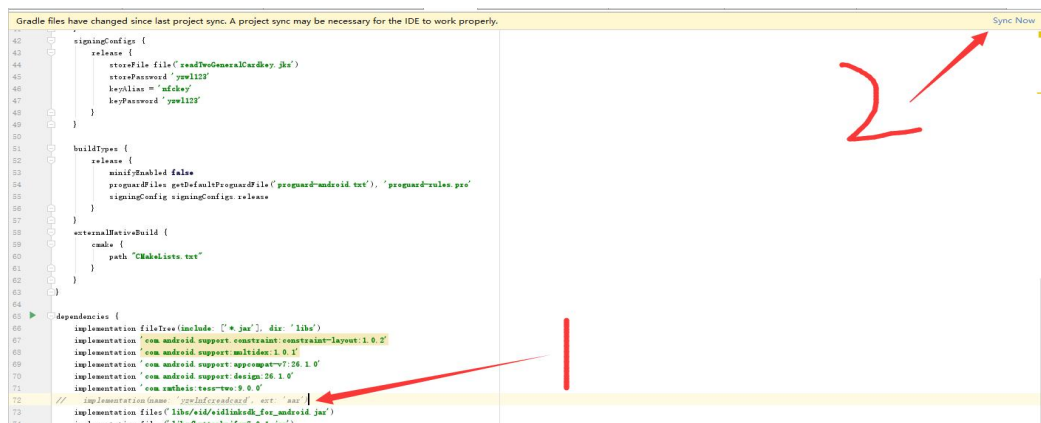
Demo 中用的 gradle 版本是 gradle-5.4.1-all.zip

- 方法调用不起

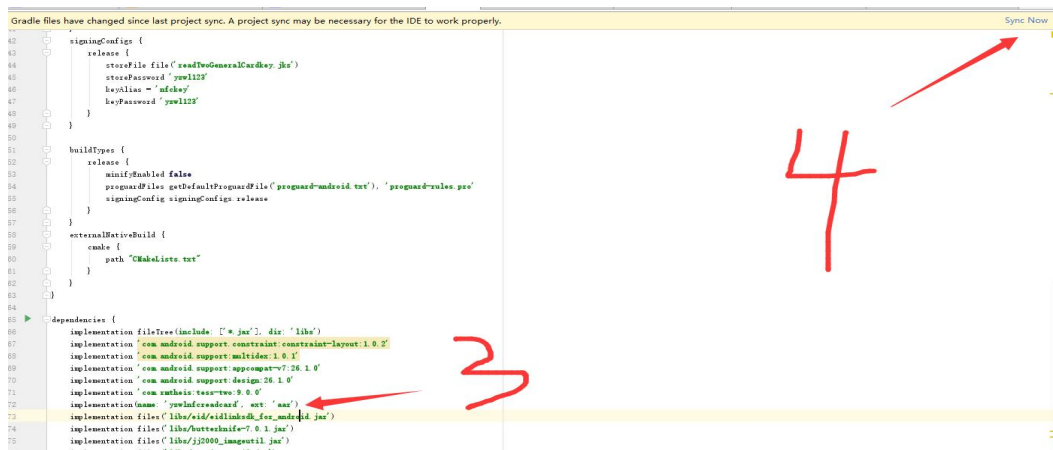
可能是 aar 没有加载起，或者替换了 aar 后没有同步工程，可按照下列图示进行：

先注释掉 build.gradle 文件中的 implementation(name:

'yzwlncfreadcard', ext: 'aar')这一行, 然后点" Sync Now"



去掉注释 build.gradle 文件中的 implementation(name: 'yzwlncfreadcard', 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。

- **运行报错：加载 libwltdecode.so 失败，找不到 libwltdecode.so 库**

删除项目 build 目录，重新编译安装即可

- **安卓 12 调用 NFC 报未知错误**

安卓 12 的新特性，在 PendingIntent 创建的时候，如果 SDK 版本大于等于 31 最后一位参数不能为 0，需要是 PendingIntent.FLAG\_MUTABLE

```
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);
```

### 指定 PendingIntent 的可变性

在 Android 12 中创建 PendingIntent 的时候，需要显示的声明是否可变，请分别使用 PendingIntent.FLAG\_MUTABLE 或 PendingIntent.FLAG\_IMMUTABLE 标志，如果您的应用试图在不设置任何可变标志的情况下创建 PendingIntent 对象，系统会抛出 IllegalArgumentException 异常，错误日志如下所示：

```
PACKAGE_NAME: Targeting S+ (version 10000 and above) requires that one of \
FLAG_IMMUTABLE or FLAG_MUTABLE be specified when creating a PendingIntent.
```

```
Strongly consider using FLAG_IMMUTABLE, only use FLAG_MUTABLE if \
some functionality depends on the PendingIntent being mutable, e.g. if \
it needs to be used with inline replies or bubbles.
```

## 7. 微信小程序插件

插件地址：

[https://mp.weixin.qq.com/wxopen/plugindevdoc?appid=wx0d82ce42bf0f4960&token=&lang=zh\\_CN](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

## 8. 微信小程序 demo 试用

微信扫码进入 demo 即可：



## 9. 微信小程序样例源码

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

## 10.