# SUNMI PAY SDK V2 开发文档 v3.2.2

上海商米科技有限公司

上海市杨浦区淞沪路 388 号创智天地 7 号楼 3 楼

目录

# 1. 版本控制

| 文档版本 | 日期 | 修改记录 | 适配的 SDK 版本 | |
|---|---|---|---|---|
| 3.0.0 | 2017/11/30 | 初始版本 | SunmiPaySDKService1.0.1 | |
| 3.0.1 | 2018/1/22 | 文档优化 | SunmiPaySDKService1.0.3/SunmiPayHardwareService3.0.44 | |
| 3.0.2 | 2018/2/23 | 错误码整理 | SunmiPaySDKService1.0.4/SunmiPayHardwareService3.0.44 | |
| 3.0.3 | 2018/3/21 | 增加 setSysParam()设置系统参数 ,增加 TransPreProcess() | SunmiPaySDKService1.0.6/SunmiPayHardwareService3.1.2 | |
| 3.0.4 | 2018/4/12 | 增加 smartCardExchange 参数说明 | SunmiPaySDKService1.0.6/SunmiPayHardwareService3.1.2 | |
| 3.0.5 | 2018/5/3 | 增加对国密 SM4 算法的支持 | SunmiPayHardwareService_v3.2.07 | |
| 3.0.6 | 2018/5/18 | 增加 SysParam 常量定义的关键字 "RESERVED" 的开启和关闭蜂鸣器使用说明 | SunmiPayHardwareService_v3.2.07 | |
| 3.0.7 | 2018/6/5 | 增加 mifareReadBlock()的 blockData 格式说明 | SunmiPayHardwareService_v3.2.07 | |
| 3.0.8 | 2018/8/16 | 增加 DUKPT 接口 | SunmiPayHardwareService_v3.2.18 | |
| 3.0.9 | 2018/8/31 | 增加纯检卡接口和读卡片数据接口 | SunmiPayHardwareService_v3.2.19 | |
| 3.1.0 | 2018/9/21 | 修改 apduCommand()描述 | SunmiPayHardwareService_v3.2.19 | |
| 3.2.0 | 2018/10/12 | 增加 V2 版 api | SunmiPayHardwareService_v3.2.20 | |
| 3.2.1 | 2018/10/22 | 增加 sp 相应错误码 | SunmiPayHardwareService_v3.2.20 | |
| 3.2.2 | 2018/10/24 | 增加 API: smartCardExchange() 和 smartCardExchangeNISO() | SunmiPayHardwareService_v3.3.0 | |
| 3.2.3 | 2018/11/01 | 1.修改了 dataEncrypt 方法参数顺序错误问题（文档和代码中参数顺序不匹配）<br>2.修改 SunmiPayKernel 中各个方法的 comment | SunmiPayHardwareService_v3.3.0 | |
| 3.2.4 | 2018/11/02 | 修改 EMVTransDataV2 类 | SunmiPayHardwareService_v3.3.0 | |
| 3.2.5 | 2018/11/05 | 1.添加 initEmvProcess()接口<br>2.修改获取/设置 TLV 数据的接口<br>3.修改 EMV 交易流程图 | SunmiPayHardwareService_v3.3.0 | |
| 3.2.6 | 2018/11/07 | 1.删除 EMVTransData 部分字段<br>2.更新 EMV 流程图 | SunmiPayHardwareService_v3.3.0 | |
| 3.2.7 | 2018/11/15 | 1.更新 PinPadConfigV2 类说明<br>2.修改 getKeyCheckValue 出参说明<br>3.Emv 流程增加以下接口：<br>(1)onAppFinalSelect()<br>(2)importAppFinalSelectStatus()<br>4.移除 apduCommandNISO()接口 | SunmiPayHardwareService_v3.3.0 | |
| 3.2.8 | 201/11/16 | 1.smartCardExChangeNISO()接口改名为 transmitApdu() | SunmiPayHardwareService_v3.3.0 | |

| | | | | |
|---|---|---|---|---|
| | | 2.transmitApdu()直接返回卡片数据 | | |
| 3.2.9 | 2018/11/19 | 修改 transmitApdu()接口描述，参数 sendBuff、recvBuff 最大长度限制为 255B | SunmiPayHardwareService_v3.3.0 | |
| 3.2.10 | 2018/11/22 | 修改 EMVListenerV2. onCertVerify ()接口，参数 certType 改为 int 类型 | SunmiPayHardwareService_v3.3.0 | |
| 3.2.11 | 2018/11/24 | 1.增加接口 EMVListenerV2. onRequestSignature()、importSignatureStatus()<br>2.增加错误码-50027<br>3.修改蜂鸣器接口 buzzerOnDevice() | SunmiPayHardwareService_v3.3.0 | |
| 3.2.12 | 2018/11/29 | 1.增加接口 EMVListenerV2. onCardDataExchangeComplete() | SunmiPayHardwareService_v3.3.0 | |
| 3.2.13 | 2018/12/04 | 1.增加接口 EMVListenerV2.onConfirmationCodeVerified()<br>2.添加错误码-50028<br>3.更新 EMV 流程图 | SunmiPayHardwareService_v3.3.0 | |
| 3.2.14 | 2018/12/08 | 1.修改接口 EMVListenerV2.onWaitAppSelect()，参数类型改为 EMVCandidateV2<br>2.修改接口 EMVOptV2.importOnlineProcStatus() | SunmiPayHardwareService_v3.3.0 | |
| 3.2.15 | 2018/12/12 | 1.回调接口 EMVListenerV2.onTransResult()添加 code 码描述 | SunmiPayHardwareService_v3.3.0 | |
| 3.2.16 | 2018/12/13 | 增加-20001———-20006 错误码 | SunmiPayHardwareService_v3.3.0 | |
| 3.2.17 | 2019/01/07 | 添加 RSA 相关的接口 | SunmiPayHardwareService_v3.3.16 | |
| 3.2.18 | 2019/01/17 | 1.添加接口 removeRSAKey()<br>2.transmitApdu()中移除 T=0、T=1 的描述 | SunmiPayHardwareService_v3.3.17 | |
| 3.2.19 | 2019/01/21 | 更新 RSA、Certificate 相关接口的参数密钥说明 | SunmiPayHardwareService_v3.3.17 | |
| 3.2.20 | 2019/03/18 | 1.修改 saveKeyDukpt()接口，增加生成初始化 KSN 代码<br>2.增加获取初始化 KSN 接口：dukptGetInitKsn()<br>3.更新 transmitApdu()接口，支持接触卡透传 apdu<br>4.更新 apduCommand()接口，修改参数 ApduSendV2、ApduRecvV2 中字段 Lc、Le、outLen 等的值域 | SunmiPayHardwareService_v3.3.28 | |
| 3.2.21 | 2019/03/19 | 1.修改 saveKeyDukpt()接口，移除生成初始化 KSN 代码<br>2.获取获取初始化 KSN 接 dukptGetInitKsn()，获取 ksn 时不依赖 saveKeyDukpt()接口 | SunmiPayHardwareService_v3.3.29 | |

| | | | | |
|---|---|---|---|---|
| | | 3.添加接口 EMVOptV2. abortTransactProcess() | | |
| 3.2.22 | 2019/04/10 | 1.添加 MifarePlus 卡数据交互接口<br>2.添加 SLE4442/4428 卡数据交互接口<br>3.添加 AT24C01/02/04/08/16/32/64/128/256/512 卡数据交互接口 | SunmiPayHardwareService_v3.3.32 | |
| 3.2.23 | 2019/05/29 | 1.添加接口 EMVOptV2.transactProcessEx();EMVOptV2.importDataExchangeStatus();<br>2.添加接口 EMVListenerV2.onRequestDataExchange() | SunmiPayHardwareService_v3.3.40 | |
| 3.2.24 | 2019/07/06 | 1.添加 AT88SCxx 系列卡操作接口<br>2.添加 transmitApduEx()接口，改进 Mifare 卡 apdu 透传<br>3.移除修改记录中[适配的SDK 版本]栏的 Firmware 版本<br>4.添加 sysGetRandom()接口<br>5.添加 RSA 签名、验签接口 | SunmiPayHardwareService_v3.3.46 | |
| 3.2.25 | 2019/07/30 | 1.添加密钥注入接口 injectPlaintextKey()、injectCiphertextKey()<br>2. CheckCardCallbackV2()中增加回调接口 findICCardEx()、findRFCardEx()、onErrorEx()<br>3.CheckCardCallbackV2. findMagCard()返回数据中增加 cardType、trackErrorCode 等字段<br>4.添加查询电子现金余额接口 queryECBalance() | SunmiPayHardwareService_v3.3.48 | |
| 3.2.26 | 2019/09/16 | 1.修改 checkCard 接口说明<br>2.修改 AidV2，增加 clsStatusCheck、zeroCheck 字段，支持 DRL<br>3.EMVOptV2 增加接口 addDrlLimitSet()、deleteDrlLimitSet()、setTermParamEx()，支持 DRL | SunmiPayHardwareService_v3.3.52 | |
| 3.2.27 | 2019/10/15 | 1.增加获取系统参数关键字：EMVVersion,PaypassVersion,PaywaveVersion,QPBOCVersion,EntryVersion,MirVersion,JCBVersion,PAGOVersion,EmvKernelCheckSum<br>2.修改输 PIN 提示文字<br>3.增减错误码：-70001~ -70006 | SunmiPayHardwareService_v3.3.53 | |
| 3.2.28 | 2019/10/28 | 1.EmvOptV2 增加接口 queryAidCapkList()、transactPreProcess()<br>2.PinPadOptV2 增加接口 cancelInput | SunmiPayHardwareService_v3.3.55 | |

| | | | | |
|---|---|---|---|---|
| | | Pin()，支持代码取消输 PIN<br>3.NFC 检卡返回数据中增加卡类别、ATQA | | |
| 3.2.29 | 2019/11/29 | 1.安全模块增加接口 dataEncryptDukptEx()、dataDecryptDukptEx()、calcMacDukptEx()、verifyMacDukptEx()、saveTR31Key()、saveCiphertextKeyRSA()、saveRSAKey()<br>2.增加获取系统参数关键字：AEVersion<br>3.EMV flowType 增加为 0x04 的类型<br>4.EMV 模块增加接口函数 addRevocList()、deleteRevocList()、sysSetTime()、sysGetTime()、clearData()<br>5.修改 signingRSA()、verifySignatureRSA()接口的参数名称 | SunmiPayHardwareService_v3.3.58 | |
| 3.2.30 | 2020/02/19 | 1.AidV2 中增加 kernelType、paramType 字段<br>2.增加获取系统参数关键字 FLASHVersion<br>3.修改 EMV FlowType 中各常量的定义 | SunmiPayHardwareService_v3.3.62 | |
| 3.2.31 | 2020/02/24 | 1.卡片模块增加接口：ctx512ReadBlock()、ctx512WriteBlock()、ctx512UpdateBlock()、ctx512GetSignature()、ctx512MultiReadBlock()、mifareIncValueDx()、mifareDecValueDx()、mifareTransfer()、mifareRestore()<br>2.密码键盘模块增加设置显示文字接口：setPinPadText()<br>3.安全模块增加删除密钥接口 deleteKey()、保存 dukpt-AES 密钥接口 saveKeyDukptAES()<br>4.系统参数增加设置/获取非接参数关键字：PCD_PARAM_A、PCD_PARAM_B、PCD_PARAM_C | SunmiPayHardwareService_v3.3.64 | |
| 3.2.32 | 2020/04/22 | 1.卡片模块增加接口 checkCardEx()、transmitApduExx()、transmitMultiApdus()<br>2.系统模块增加接口 setStatusBarDropDownMode()、setNavigationBarVisibility()、setHideNavigationBarItems()、sysPowerManage()<br>3.系统参数增加关键字：DPASVersion、APEMVVersion<br>4.增加 EMV 交易结果常量 AidlConstants.EMV.TransResult<br>5.修改 importOnlineProcStatus()接 | SunmiPayHardwareService_v3.3.66 | |

| | | | | |
|---|---|---|---|---|
| | | 口参数 status 描述 | | |
| 3.2.33 | 2020/05/22 | 1.修改 setTermParamEx()接口参数描述 | SunmiPayHardwareService_v3.3.74 | |
| 3.2.34 | 2020/06/02 | 1.PinPad 模块增加 setPinPadMode()、getPinPadMode()接口 | SunmiPayHardwareService_v3.3.76 | |
| 3.2.35 | 2020/06/04 | 1.修改 setTermParamEx()接口参数描述 | SunmiPayHardwareService_v3.3.77 | |
| 3.2.36 | 2020/06/16 | 1.检卡模块增加接口 checkCardEnc()<br>2.CheckCardCallbackV2.findMagCard()中增加关键字“pan”，“name”，“expire” | SunmiPayHardwareService_v3.3.80 | |
| 3.2.37 | 2020/06/17 | 1.增加 ETCOptV2 模块 | SunmiPayHardwareService_v3.3.81 | |
| 3.2.38 | 2020/07/03 | 1.系统参数增减关键字：SupportETC | SunmiPayHardwareService_v3.3.84 | |
| 3.2.39 | 2020/07/06 | 1.修改 EMV transactProcessEx()接口描述<br>2.修改 checkCardEx()接口参数描述 | SunmiPayHardwareService_v3.3.86 | |
| 3.2.40 | 2020/09/11 | 1.SecuityOptV2 增加接口：calcMacEx()、generateSM2Keypair()、injectSM2Key()、sm2Sign()、sm2VerifySign()、sm2EncryptData()、sm2DecryptData()、calcSecHash()<br>2.EMVOptV2 增加接口 setAccountDataSecParam()、getAccountSecData()<br>3.PinPadOptV2 中增加接口：initPinPadEx()<br>4.系统参数增加关键字：PUREVersionFul、EFTPOSVersionFull、APEMVVersionFull<br>5.修改 Dukpt-3DES 密钥范围为 0-9 或 1100-1199<br>打印模块添加 setPrintSpeed()接口 | SunmiPayHardwareService_v3.3.95 | |
| 3.2.41 | 2020/11/06 | 1.附录中添加密钥索引说明 | SunmiPayHardwareService_v3.3.95 | |
| 3.2.42 | 2020/11/11 | 1.添加 RSA transformation: RSA/ECB/OAEPWithSHA-1AndMGF1Padding、RSA/ECB/OAEPWithSHA-256AndMGF1Padding、RSA/ECB/OAEPWithSHA-512AndMGF1Padding<br>2.打印模块添加 setPrintHeatPoint()接口 | SunmiPayHardwareService_v3.3.96 | |
| 3.3.43 | 2020/12/16 | 1.EMVOptV2.setTermParamEx()参数增加 key：contactlessManualSelApp、importScriptData | SunmiPayHardwareService_v3.3.98 | |

| | | | | |
|---|---|---|---|---|
| | | 2.AidlConstants.EMV.TLVOpCode 中增加 OP_ADD_SELF_DEFINE_TAG 、OP_DEL_SELF_DEFINE_TAG<br>2. 系统参数增加关键字 SecMode、PCD_IFMVersion | | |
| 3.2.44 | 2020/12/29 | 1.修改 saveKeyDukptAES()接口，参数 KSN 长度改为 12 字节 | SunmiPayHardwareService_v3.3.99 | |
| 3.2.45 | 2020/12/30 | 1.ETC 模块添加扣费相关接口<br>2.修改 ETC 模块扣费相关接口，搜索 OBU 改为回调方式<br>3.ETC 模块增加 OBU 防休眠（保活）接口 | SunmiPayHardwareService_v3.3.100 | |
| 3.2.46 | 2021/02/04 | 1. 增加接口 EMVOptV2.importTermRiskManagementStatus()<br>2. 增加接口：EMVListenerV2.onTermRiskManagement() | SunmiPayHardwareService_v3.3.102 | |
| 3.2.47 | 2021/04/28 | 1. 卡片模块增加读卡模块增加 smartCardIoControl()接口<br>2.安全模块增加校验 Mac、RSA、RKI 相关的扩展接口<br>3.安全模块更新 MKSK、dukpt 密钥索引范围<br>4.移除关键字 SysParam.SDK_VERSION | SunmiPayHardwareService_v3.3.112 | |
| 3.2.48 | 2021/07/27 | 1.PinPadOptV2 增加接口 setAntiExhaustiveProtectionMode()、getAntiExhaustiveProtectionMode() | SunmiPayHardwareService_v3.3.126 | |
| 3.2.49 | 2021/08/30 | 1.更新错误码表，添加 libbase 新增的错误码 | SunmiPayHardwareService_v3.3.136 | |
| 3.2.50 | 2021/09/16 | 1. 增加常量 AidlConstants.EMV.KernelType.RUPAY<br>2.更新 dukptIncreaseKSN()接口的描述 | SunmiPayHardwareService_v3.3.137 | |

# ● 2. 概述

## 2.1 引言

SunmiPaySDK 是 Sunmi 基于固件封装的贴近 java 开发者的一套调用硬件的接口，开发中通过该 SDK 可以快速调用 Sunmi 金融机具相应的固件接口，实现自己相关业务逻辑的开发。SDK 主要包含：设备信息基础模块，卡操作模块，密码键盘模块，EMV 模块，安全模块。

本文档为 SunmiPaySDK V2 版接口文档，相比于 V1 版接口，V2 接口更加便于开发这理解和调用。

## 2.2 快速集成商米支付 SDK

### 2.2.1 快速集成

本文是商米金融机具标准的集成开发指南文档。用以指导 SDK 的使用方法，默认读者已经熟悉 IDE 的基本使用方法，具有一定的 Android 编程基础，熟悉金融相关规范、流程以及相关概念（密钥，pinblock，pan,emv,mac 等相关概念），目前 SDK 只支持在商米设备 P1N, P1-4G, P2 Pro, P2 Lite 上运行，使用 SDK 前，请务必仔细阅读该文档，使用前请检查以下前提条件

1. 设备机型为 P1N, P1-4G, P2 Pro, P2 Lite。（设置-关于设备-型号）
2. 点开设置—>应用—>右上角—>点击选择显示系统应用
   P1N, P1-4G, P2 Pro, P2 Lite 机具上已安装 SunmiPayHardwareService 版本 3.1.2 或者更高版本。
3. android studio 快速集成，将 PayLib_vx.x.x.aar 包放在 libs 文件夹下

```
repositories {
       flatDir {
           dirs 'libs'
       }
}
dependencies {
    ......
   compile(name: 'PayLib', ext: 'aar')
}
```

导入 aar 报完成后，重新 build 项目。

### 2.2.2 SDK 所支持的系统版本以及 IDE

目前 SDK 只支持 API-19(Android 4.4)
目前 SDK 只支持 Android studio、Intellij 的集成

# 3. API

## 3.1 SunmiPayKernel SDK 操作对象

### 3.1.1 获取 SunmiPayKernel 实例

| | | |
|---|---|---|
| 原型 | SunmiPayKernel getInstance() | |
| 功能 | 获取 SDK 单例对象 | |
| 参数 | [in] | None |
| 返回值 | SunmiPayKernel | |
| 备注 | None | |

### 3.1.2 连接 PaySDK

| | | |
|---|---|---|
| 原型 | initPaySDK (Context context,ConnectCallback connectCallback) | |
| 功能 | 连接支付服务 | |
| 参数 | context[in] | 上下文对象 |
| | connectCallback[in] | 连接状态回调 详见：ConnectCallback |
| 返回值 | None | |
| 备注 | 建议在 app 的 Application 类中连接支付服务，未调用该方法就去调用任意 SDK API 方法将会抛出 NullPointerException | |

### 3.1.3 断开与 PaySDK 的连接

| | | |
|---|---|---|
| 原型 | destroyPaySDK () | |
| 功能 | 断开与支付服务的连接 | |
| 参数 | [in] | None |
| 返回值 | None | |
| 备注 | 断开与支付服务的连接。调用该函数后，SunmiPayKernel 中所有操作模块对象全部置为 null，调用任意 SDK API 会抛出 NullPointerException。为防止内存泄露，在 Activity 的 onDestroy()方法中调用此方法是最佳实践。 | |

### 3.1.4 ConnectCallback-连接回调

#### 3.1.4.1 成功连接 PaySDK 后回调

| | |
|---|---|
| 原型 | onConnectPaySDK () |
| 功能 | 支付服务的连接成功时回调本方法 |

| | | |
|---|---|---|
| 参数 | [in] | None |
| 返回值 | None | |
| 备注 | SDK 所有 api 的调用必须在收到该回调后进行，初始化 SDK 后，开发者需要关注该函数的回调，收到该函数表示 SDK 初始化成功，可以调用 SDK 相关 API | |

#### 3.1.4.2 成功断开与 PaySDK 的连接后回调

| | | |
|---|---|---|
| 原型 | onDisconnectPaySDK () | |
| 功能 | 断开与支付服务的连接时回调本方法 | |
| 参数 | [in] | None |
| 返回值 | None | |
| 备注 | 1.收到此回调表明当前 app 已断开与 Pay SDK 的连接，此后若调用任意 SDK API 将抛出 NullPointerException。若要重新连接 Pay SDK，请调用 initPaySDK()方法<br>2.当 Pay SDK 服务死亡时本方法也会被回调 | |

## 3.2 公共成员变量

绑定 PaySDK 成功后，SunmiPayKernel 中提供了一些功能模块，用于与 PaySDK 交互，见下表：

| 变量名 | 说明 | 备注 |
|---|---|---|
| mPinPadOptV2 | 密码键盘模块 | 包含安全的密码键盘相关 API |
| mBasicOptV2 | 基础信息模块 | 获取基础信息，控制 LED 灯，蜂鸣器等 API |
| mReadCardOptV2 | 卡操作模块 | 读银行卡，会员卡，M1 卡等相关操作 API |
| mEMVOptV2 | EMV 模块 | Emv 流程的相关 API |
| mSecurityOptV2 | 安全模块 | 保存密钥，mac 计算，加密等 API |

## 3.3 基础操作模块

### 3.3.1 获取系统参数

| | | |
|---|---|---|
| 原型 | String getSysParam(String key) | |
| 功能 | 通过用户参数关键字，读取系统资源关键字的属性 | |
| 参数 | key[in] | 用户参数关键字见 AidlConstantsV2. SysParam |
| 返回值 | 所查询的属性值 | |
| 备注 | 如果所要求的属性值不存在，则返回“NULL” | |

### 3.3.2 设置系统参数

| | |
|---|---|
| 原型 | int setSysParam(String key, String value) |

| | | |
|---|---|---|
| 功能 | 写入资源窗口关键字的属性 | |
| 参数 | key[in] | 仅能设置 RESERVED |
| | value[in] | 关键字属性值 |
| 返回值 | 0-成功<br>非 0-错误码 | |
| 备注 | 仅能设置 RESERVED | |

### 3.3.3 蜂鸣器

| | | |
|---|---|---|
| 原型 | int buzzerOnDevice(int count, int freq, int duration, int interval) | |
| 功能 | 控制设备上的蜂鸣器响 | |
| 参数 | count[in] | 连续鸣响次数 0~100 |
| | freq[in] | 鸣叫的频率或声调（单位：Hz） |
| | duration[in] | 鸣叫时长（单位：ms） |
| | interval[in] | 次鸣响的时间间隔（单位：ms） 0~10000 |
| 返回值 | 0-成功<br>非 0-错误码 | |
| 备注 | P2lite/P2Pro/P2 设备上无实体蜂鸣器，蜂鸣声通过扬声器发出，本接口的参数 **freq** 和 **duration** 无效 | |

### 3.3.4 LED 灯控制

| | | |
|---|---|---|
| 原型 | int ledStatusOnDevice (int ledIndex, int ledStatus) | |
| 功能 | 控制设备上的 LED 灯状态 | |
| 参数 | ledIndex[in] | 参考附录：LedLight 常量定义 AidlConstantsV2.LedLight |
| | ledStatus[in] | LED 状态，1 表示 LED 灭，0 表示 LED 亮 |
| 返回值 | 0-成功<br>非 0-错误码 | |
| 备注 | None | |

### 3.3.5 设置屏幕独占

| | | |
|---|---|---|
| 原型 | int setScreenMode(int mode) | |
| 功能 | 设置屏幕独占 禁用底部导航栏和 SystemUI 下拉框、禁用音量键 | |
| 参数 | mode[in] | 设置屏幕独占的模式，1：设置屏幕独占，-1 设置取消屏幕独占 |
| 返回值 | 0-成功<br>非 0-错误码 | |
| 备注 | 设置屏幕独占一般需要适当的时机取消，建议用户使用该接口配合电源管理锁使用，保持屏幕常亮，不锁屏，否则可能出现息屏后还保持屏幕独占，必须拔电池重启设备才能解除屏幕独占 | |

### 3.3.6 获取随机数

| 原型 | int sysGetRandom(byte[] randData,int len) | |
|---|---|---|
| 功能 | 获取指定长度的随机数 | |
| 参数 | randData[out] | Buffer，存放获取到的随机数 |
| | len[in] | 随机数的长度，范围 0~256 |
| 返回值 | 0-成功<br>非 0-错误码 | |
| 备注 | | |

### 3.3.7 LED 灯控制（扩展）

| 原型 | int ledStatusOnDeviceEx(int redStatus, int greenStatus, int yellowStatus, int blueStatus) | |
|---|---|---|
| 功能 | 单次控制所有 LED 灯亮灭 | |
| 参数 | redStatus[in] | 红灯状态，0-亮，1-灭 |
| | greenStatus[in] | 绿灯状态，0-亮，1-灭 |
| | yellowStatus[in] | 黄灯状态，0-亮，1-灭 |
| | blueStatus[in] | 蓝灯状态，0-亮，1-灭 |
| 返回值 | 0-成功<br>非 0-错误码 | |
| 备注 | | |

### 3.3.8 设置状态栏下拉模式

| 原型 | int setStatusBarDropDownMode(int mode) | |
|---|---|---|
| 功能 | 设置状态栏下拉模式 | |
| 参数 | mode[in] | 下拉模式，0-启用下拉，1-禁用下拉 |
| 返回值 | 0-成功<br>非 0-错误码 | |
| 备注 | | |

### 3.3.9 设置导航栏可见性

| 原型 | int setNavigationBarVisibility(int visibility) | |
|---|---|---|
| 功能 | 设置导航栏可见性 | |
| 参数 | visibility[in] | 导航栏可见性，0-隐藏，1-显示 |
| 返回值 | 0-成功<br>非 0-错误码 | |

| | |
|---|---|
| 备注 | |

### 3.3.10 设置隐藏导航栏图标

| | | |
|---|---|---|
| 原型 | int setHideNavigationBarItems(int flag) | |
| 功能 | 隐藏导航栏图标 | |
| 参数 | flag[in] | 要隐藏的图标类型组合，如：<br>STATUS_BAR_DISABLE_HOME=0x00200000;//隐藏 home 键<br>STATUS_BAR_DISABLE_BACK = 0x00400000;//隐藏 back 键<br>STATUS_BAR_DISABLE_RECENT = 0x01000000;//隐藏 recent 键<br>flag=STATUS_BAR_DISABLE_HOME \| STATUS_BAR_DISABLE_BACK 可同时隐藏 home 键和 back 键 |
| 返回值 | 0-成功<br>非 0-错误码 | |
| 备注 | | |

### 3.3.11 电源管理

| | | |
|---|---|---|
| 原型 | int sysPowerManage(int mode) | |
| 功能 | 设备电源管理 | |
| 参数 | mode[in] | 模式，1-休眠(不支持)，2-关机，3-重启 |
| 返回值 | 0-成功<br>非 0-错误码 | |
| 备注 | | |

## 3.4 卡操作模块

### 3.4.1 ReadCardOpt 卡操作模块对象

#### 3.4.1.1 检卡

| | | |
|---|---|---|
| 原型 | void checkCard(int cardType, CheckCardCallbackV2 callback, int timeout) | |
| 功能 | 针对各类卡的检卡，支持磁条卡，IC 卡和非接卡，检卡完成后，会将卡片类型放到 CheckCardCallbackV2 中。 | |
| 参数 | cardType[in] | 卡类型组合，支持同时检多种卡，值为 CardType.value 的组合。例如同时检磁条、NFC 和 IC 卡可传入值：<br>CardType. MAGNETIC.getValue() \| CardType. NFC.getValue() \| CardType. IC.getValue() |

| | | |
|---|---|---|
| | callback[in] | 检卡回调，详见 CheckCardCallbackV2 |
| | timeout[in] | 超时时间，（单位为秒） 参数取值范围：1-120（秒） |
| 返回值 | None | |
| 备注 | 不区分银行卡和非银行卡 | |

### 3.4.1.2 取消检卡

接口使用说明：人为返回必须调用取消检卡，终止底层阻塞线程，否则下次执行函数会失败（例如点击物理返回键，点击界面导航条返回键需要调用该函数）

| | | |
|---|---|---|
| 原型 | void cancelCheckCard() | |
| 功能 | 取消检卡 | |
| 参数 | [in] | None |
| 返回值 | None | |
| 备注 | 检卡未返回（CheckCardCallbackV2 成功或者失败接口未回调）离开界面前需要调用该函数 | |

### 3.4.1.3 APDU 指令交互(Recommended)

| | | |
|---|---|---|
| 原型 | int apduCommand (int cardType, ApduSendV2 send, ApduRecvV2 recv) | |
| 功能 | 与接触式 IC 卡之间使用接口协议（T=0 及 T=1）进行数据交互；<br>与非接触式 IC 卡之间使用 T=CL 协议进行数据交互。 | |
| 参数 | cardTypy[in] | 当前操作的卡片类型 |
| | send[in] | class ApduSendV2 {<br>byte[] command; //命令<br>short lc; //dataIn 中有效数据的长度(0~256)<br>byte[] dataIn; //数据体（最长 256 字节）<br>short le; //le (0~256)<br>} |
| | recv[out] | class ApduRecvV2 {<br>short outLen; // outData 中的有效数据的长度(0~256)<br>byte[] outData; //返回数据（最长 256 字节）<br>byte swa; //swa<br>byte swb; //swb<br>} |
| 返回值 | 0-成功<br>非 0-错误码 | |
| 备注 | ApduSendV2 中各字段的含义及对最终发送给卡片的 apdu 的影响 ,请参见文档: **apdu format and implement in sunmi way**<br>apdu format and implement in sunmi | |

### 3.4.1.4 APDU 指令交互(不推荐)

| | | |
|---|---|---|
| 原型 | int smartCardExchange(int cardType, byte[] apduSend, byte[] apduRecv) | |
| 功能 | 与接触式 IC 卡之间使用接口协议（T=0 及 T=1）进行数据交互；<br>与非接触式 IC 卡之间使用 T=CL 协议进行数据交互。 | |
| 参数 | cardTypy[in] | 当前操作的卡片类型 |
| | apduSend[in] | 要发送的 APDU 指令，格式为：<br>Command(4B)+LC(1B，值为 len)+inData(len B)+LE(1B)<br>LC 的值是无符号数，范围 0~255 |
| | apduRecv[out] | 应答数据单元，apduRecv.length>=260，格式为：<br>outLen(2B,大端模式，值为 len)+outData(len B)+SWA(1B)+SWB(1B) |
| 返回值 | 0-成功<br>非 0-错误码 | |
| 备注 | None | |

### 3.4.1.5 透传 APDU 指令

| | | |
|---|---|---|
| 原型 | int transmitApdu(int cardType, byte[] sendBuff, byte[] recvBuff) | |
| 功能 | **以透传数据的方式**与接触式 IC 卡之间使用接口协议（T=0 及 T=1）进行数据交互；<br>与非接触式 IC 卡之间使用 T=CL 协议进行数据交互。 | |
| 参数 | cardTypy[in] | 当前操作的卡片类型 |
| | sendBuff [in] | 要透传给卡片的数据，最大为 255B |
| | recvBuff [out] | 卡片的应答数据，应答数据的最大长度为 255B，recvBuff.length>=255 |
| 返回值 | >=0- recvBuff 中有效数据的长度<br><0-错误码 | |
| 备注 | None | |

### 3.4.1.6 卡片下电

| | | |
|---|---|---|
| 原型 | int cardOff(int cardType) | |
| 功能 | 下电(接触式 IC 卡)或移走卡片（非接触式 IC 卡）。 | |
| 参数 | cardType[in] | 卡类型，同时支持多种卡检卡，传入参数参考：<br>AidlConstantsV2.CardType.getValue()根据检卡成功的卡片类型填写，一次一种 |
| 返回值 | 0-卡片已经下电(接触式 IC)或移走(非接触式 IC 卡)<br>非 0-错误码 | |
| 备注 | 非接触式 IC 卡调用此函数关闭载波。 | |

#### 3.4.1.7 判断卡片是否在位

| 原型 | int getCardExistStatus(int cardType) | |
|---|---|---|
| 功能 | 判断卡是否在位， | |
| 参数 | cardType[in] | 卡类型（非复合类型），仅支持:<br>NFC、IC、PSAM0、PSAM1<br>每次只能为 4 种类型中的一种 |
| 返回值 | AidlConstantsV2.CardExistStatus<br>CARD_ABSENT = 0x01;// 卡片不在位<br>CARD_PRESENT = 0x02;// 卡片在位 | |
| 备注 | None | |

#### 3.4.1.8 Mifare Classsic

##### 3.4.1.8.1 M1 卡片认证

| 原型 | int mifareAuth(int keyType, int block, byte[] key) | |
|---|---|---|
| 功能 | Mifare 卡片认证 | |
| 参数 | keyType[in] | 密钥类型，0 表示 KEY A、1 表示 KEY B； |
| | block[in] | 认证块号 |
| | key[in] | 密钥数据，共 6 字节 |
| 返回值 | 0-认证成功<br>非 0-错误码 | |
| 备注 | None | |

##### 3.4.1.8.2 M1 读块数据

| 原型 | int mifareReadBlock( int block, byte[] outData) | |
|---|---|---|
| 功能 | Mifare 卡片读块数据 | |
| 参数 | block[in] | 待读取的块号； |
| | outData [out] | 缓存区，存储读取到块数据 |
| 返回值 | >=0-outData 中有效数据的长度<br><0-错误码 | |
| 备注 | None | |

##### 3.4.1.8.3 M1 写块数据

| 原型 | int mifareWriteBlock( int block, byte[] data) | |
|---|---|---|
| 功能 | Mifare 卡片写块数据 | |
| 参数 | block[in] | 待写入的块号 |
| | data[in] | 待写入的块数据 |

| | |
|---|---|
| 返回值 | 0-写入数据块成功<br>非 0-错误码 |
| 备注 | None |

#### 3.4.1.8.4 M1 加值

| | | |
|---|---|---|
| 原型 | int mifareIncValue( int block, byte[] value) | |
| 功能 | Mifare 卡片加值 | |
| 参数 | block[in] | 待加值的块号 |
| | value[in] | 加值金额，共 4 字节（小端模式，低字节在前，高字节在后） |
| 返回值 | 0-加值成功<br>非 0-错误码 | |
| 备注 | None | |
| 示例 | None | |

#### 3.4.1.8.5 M1 减值

| | | |
|---|---|---|
| 原型 | int mifareDecValue( int block, byte[] value) | |
| 功能 | Mifare 减值 | |
| 参数 | block[in] | 待减值的块号 |
| | value[in] | 减值金额，共 4 字节，（小端模式，低字节在前，高字节在后） |
| 返回值 | 0-减值成功<br>非 0-错误码 | |
| 备注 | None | |
| 示例 | None | |

#### 3.4.1.8.6 M1 加值（敛缩）

| | | |
|---|---|---|
| 原型 | int mifareIncValueDx(int block, byte[] value) | |
| 功能 | Mifare 卡片加值（不包含 transfer 功能） | |
| 参数 | block[in] | 待加值的块号 |
| | value[in] | 加值金额，共 4 字节（小端模式，低字节在前，高字节在后） |
| 返回值 | 0-加值成功<br>非 0-错误码 | |
| 备注 | M1 加值包含加值（加值后的值缓存在寄存器）、transfer（将寄存器中的值写到指定的块）两个步骤。**mifareIncValue()**接口包含此两步，本接口不包含 transfer 步骤 | |
| 示例 | | |

#### 3.4.1.8.7 M1 减值（敛缩）

| | | |
|---|---|---|
| 原型 | int mifareDecValueDx(int block, byte[] value) | |
| 功能 | Mifare 卡片减值（不包含 transfer 功能） | |
| 参数 | block[in] | 待减值的块号 |

| | | |
|---|---|---|
| | value[in] | 减值金额，共 4 字节，（小端模式，低字节在前，高字节在后） |
| 返回值 | 0-减值成功<br>非 0-错误码 | |
| 备注 | M1 减值包含减值（减值后的值缓存在寄存器）、transfer（将寄存器中的值写到指定的块）两个步骤。**mifareDecValue()**接口包含此两步，本接口不包含 transfer 步骤 | |
| 示例 | None | |

#### 3.4.1.8.8 M1 传输

| | | |
|---|---|---|
| 原型 | int mifareTransfer(int destBlock) | |
| 功能 | Mifare 传输（将数据寄存器中的值写到指定的块） | |
| 参数 | destBlock[in] | 存储数据的块号 |
| 返回值 | 0-成功<br>非 0-错误码 | |
| 备注 | None | |
| 示例 | None | |

#### 3.4.1.8.9 M1 存储

| | | |
|---|---|---|
| 原型 | int mifareRestore(int srcBlock) | |
| 功能 | Mifare 存储（将数据块中的内容存到数据寄存器中） | |
| 参数 | srcBlock [in] | 数据块号 |
| 返回值 | 0-成功<br>非 0-错误码 | |
| 备注 | None | |
| 示例 | None | |

### 3.4.1.9 Mifare Ultralight C

#### 3.4.1.9.1 Mifare Ultralight C 卡片认证

| | | |
|---|---|---|
| 原型 | int mifareUltralightCAuth(byte[] authKey) | |
| 功能 | Mifare ultralight c 卡片认证 | |
| 参数 | authKey[in] | 认证密钥 |
| 返回值 | 0-认证成功<br>-1-认证失败 | |
| 备注 | None | |

#### 3.4.1.9.2 Mifare Ultralight C 读取数据

| | |
|---|---|
| 原型 | int mifareUltralightCReadData(int block,byte[] outData) |
| 功能 | Mifare ultralight c 读取数据 |

| | | |
|---|---|---|
| 参数 | block[in] | 块号 |
| | outData[out] | 缓存区，存储读取到块数据 |
| 返回值 | >=0-outData 中有效数据的长度<br><0-错误码 | |
| 备注 | None | |

#### 3.4.1.9.3 Mifare Ultralight C 写入数据

| | | |
|---|---|---|
| 原型 | int mifareUltralightCWriteData(int block,byte[] data) | |
| 功能 | Mifare ultralight c 读取数据 | |
| 参数 | block[in] | 块号 |
| | data[in] | 块数据 |
| 返回值 | 0-写数据成功<br>非 0-错误码 | |
| 备注 | None | |

### 3.4.1.10 Mifare Plus SL3

#### 3.4.1.10.1 Mifare Plus 读数据

| | | |
|---|---|---|
| 原型 | int mifarePlusReadBlock(int block, byte[] key, byte[] outData) | |
| 功能 | Mifare Plus 读块数据 | |
| 参数 | block[in] | 块号，范围取决于卡片，如 MF1PLUS80x 包含 32 个 4 block 的 sector 和 8 个 16 block 的 sector，每个 block 长 16 字节，存储空间总大小为 32*4*16+8*16*16=4096，总 block 个数为 32*4+8*16 = 256,<br>块号的取值范围为 00~FF |
| | key[in] | 块密码，16 字节 |
| | outData[out] | 缓存区，存储读取到块数据，outData.length>=16 |
| 返回值 | >=0-outData 中有效数据的长度<br><0-错误码 | |
| 备注 | None | |

#### 3.4.1.10.2 Mifare Plus 写数据

| | | |
|---|---|---|
| 原型 | int mifarePlusWriteBlock(int block, in byte[] key, byte[] data) | |
| 功能 | Mifare Plus 读块数据 | |
| 参数 | block[in] | 块号，范围取决于卡片，如 MF1PLUS80x 包含 32 个 4 block 的 sector 和 8 个 16 block 的 sector，每个 block 长 16 字节，总内存大小为 32*4*16+8*16*16=4096，总 block 个数为 32*4+8*16 = 256,<br>块号的取值范围为 00~FF |
| | key[in] | 块密码，16 字节 |
| | data[out] | 要写入的数据 |
| 返回值 | 0-成功 | |

| | | |
|---|---|---|
| | 非 0-错误码 | |
| 备注 | None | |

### 3.4.1.10.3 Mifare Plus 修改 block 密钥

| | | |
|---|---|---|
| 原型 | int mifarePlusChangeBlockKey(int block, in byte[] oldKey, in byte[] newKey) | |
| 功能 | Mifare Plus 读块数据 | |
| 参数 | block[in] | 块号，范围取决于卡片，如 MF1PLUS80x 包含 32 个 4 block 的 sector 和 8 个 16 block 的 sector，每个 block 长 16 字节，总内存大小为 32*4*16+8*16*16=4096，总 block 个数为 32*4+8*16 = 256,<br>块号的取值范围为 00~FF |
| | oldKey [in] | 旧密码，16 字节 |
| | newKey [out] | 新密码，16 字节 |
| 返回值 | 0-成功<br>非 0-错误码 | |
| 备注 | None | |

### 3.4.1.11 SLE4442/4428

### 3.4.1.11.1 SLE 卡片认证

| | | |
|---|---|---|
| 原型 | int sleAuthKey(in byte[] key) | |
| 功能 | SLE 卡片认证 | |
| 参数 | key[in] | 密码，SLE4442 密码长度为 3 字节，SLE4428 密码长度为 2 字节，默认值全 F |
| 返回值 | 0-成功<br>非 0-错误码 | |
| 备注 | None | |

### 3.4.1.11.2 SLE 修改密码

| | | |
|---|---|---|
| 原型 | int sleChangeKey(byte[] newKey) | |
| 功能 | SLE 修改密码 | |
| 参数 | newKey[in] | 新密码，SLE4442 密码长度为 3 字节，SLE4428 密码长度为 2 字节，默认值全 F |
| 返回值 | 0-成功<br>非 0-错误码 | |
| 备注 | 修改密码前必须先验证密码 | |

### 3.4.1.11.3 SLE 读数据

| | | |
|---|---|---|
| 原型 | int sleReadData(int startAddress, int length, byte[] outData) | |
| 功能 | SLE 读一段连续内存数据 | |
| 参数 | startAddress [in] | 起始地址 |
| | length[in] | 数据长度 |

| | | |
|---|---|---|
| | outData[out] | 缓存区，存储读到的数据 |
| 返回值 | >=0-outData 中有效数据的长度<br><0-错误码 | |
| 备注 | | |

### 3.4.1.11.4 SLE 写数据

| | | |
|---|---|---|
| 原型 | int sleWriteData(int startAddress, byte[] data) | |
| 功能 | SLE 写数据到一段连续的内存 | |
| 参数 | startAddress [in] | 起始地址 |
| | data[in] | 要写的数据 |
| 返回值 | 0-成功<br>非 0-错误码 | |
| 备注 | 如果卡片有密码，则写数据前需要先验证密码 | |

### 3.4.1.11.5 SLE 获取剩余可认证次数

| | | |
|---|---|---|
| 原型 | int sleGetRemainAuthCount() | |
| 功能 | SLE 获取卡片剩余可验证密码的次数 | |
| 参数 | [in] | None |
| 返回值 | >=0-剩余认证次数<br><0-错误码 | |
| 备注 | 当剩余认证次数为 0 时，卡片被锁，无法进行密码验证或写数据操作 | |

### 3.4.1.11.6 SLE 设置存储单元保护位

| | | |
|---|---|---|
| 原型 | int sleWriteProtectionMemory(int startAddress, int length) | |
| 功能 | SLE 锁定一段内存单元，使其不能再写入数据，无论密码验证成功与否 | |
| 参数 | startAddress [in] | 起始地址 |
| | length[in] | 长度 |
| 返回值 | 0-成功<br>非 0-错误码 | |
| 备注 | | |

### 3.4.1.11.7 SLE 读取一段存储单元的保护位

| | | |
|---|---|---|
| 原型 | int sleReadMemoryProtectionStatus(int startAddress, int length, byte[] dataOut) | |
| 功能 | SLE 读取一段存储单元的保护位 | |
| 参数 | startAddress [in] | 起始地址 |
| | length[in] | 长度 |
| | dataOut[out] | 缓存区，存储读取到的存储单元保护位，每个存储单元保护位值可以为：0-锁定，1-未锁定 |
| 返回值 | >=0- dataOut 中有效数据的长度<br><0-错误码 | |

| | |
|---|---|
| 备注 | |

### 3.4.1.12 AT24C01/02/04/08/16/32/64/128/256/512

#### 3.4.1.12.1 AT24C 读数据

| | | |
|---|---|---|
| 原型 | int at24cReadData(int startAddress, int length, byte[] outData) | |
| 功能 | AT24C 读取一段连续的存储单元的数据 | |
| 参数 | startAddress [in] | 起始地址 |
| | length[in] | 长度 |
| | dataOut[out] | 缓存区，存储读取到的数据 |
| 返回值 | >=0-dataOut 中有效数据的长度<br><0-错误码 | |
| 备注 | | |

#### 3.4.1.12.2 AT24C 写数据

| | | |
|---|---|---|
| 原型 | int at24cWriteData(int startAddress, byte[] data) | |
| 功能 | AT24C 向一段连续的存储单元中写入数据 | |
| 参数 | startAddress [in] | 起始地址 |
| | data [in] | 要写入的数据 |
| 返回值 | 0-成功<br>非 0-错误码 | |
| 备注 | | |

### 3.4.1.13 AT88SC

#### 3.4.1.13.1 AT88SC 卡片认证

| | | |
|---|---|---|
| 原型 | int at88scAuthKey(byte[] key, int rwFlag, int zoneNo) | |
| 功能 | AT88SC 卡片认证 | |
| 参数 | key[in] | 密码，AT88SC1608 密码长度为 3 字节，默认值全 F |
| | rwFlag[in] | 读写标志，0-写密码，1-读密码 |
| | zoneNo[in] | 用户区域编号，范围 0~7 |
| 返回值 | 0-成功<br>非 0-错误码 | |
| 备注 | AT88SC1608 共 8 个用户区（0~7），每个区 256 字节，共 2048 字节，地址范围 0~2047（0~7FF） | |

#### 3.4.1.13.2 AT88SC 修改密码

| | |
|---|---|
| 原型 | int at88scChangeKey(byte[] newKey, int rwFlag, int zoneNo) |
| 功能 | AT88SC 修改密码 |

| | | |
|---|---|---|
| 参数 | newKey[in] | 新密码，AT88SC1608 密码长度为 3 字节 |
| | rwFlag[in] | 读写标志，0-写密码，1-读密码 |
| | zoneNo[in] | 用户区域编号，范围 0~7 |
| 返回值 | 0-成功<br>非 0-错误码 | |
| 备注 | 修改密码前必须先验证密码 | |

#### 3.4.1.13.3 AT88SC 读数据

| | | |
|---|---|---|
| 原型 | int at88scReadData(int startAddress, int length, int zoneFlag, byte[] outData) | |
| 功能 | AT88SC 读一段连续内存数据 | |
| 参数 | startAddress [in] | 起始地址 |
| | length[in] | 数据长度 |
| | zoneFlag[in] | 区域标志，0-配置区，1-用户区 |
| | outData[out] | 缓存区，存储读到的数据 |
| 返回值 | >=0-outData 中有效数据的长度<br><0-错误码 | |
| 备注 | None | |

#### 3.4.1.13.4 AT88SC 写数据

| | | |
|---|---|---|
| 原型 | int at88scWriteData(int startAddress, int zoneFlag, byte[] dataIn) | |
| 功能 | AT88SC 写数据到一段连续的内存 | |
| 参数 | startAddress [in] | 起始地址 |
| | zoneFlag[in] | 区域标志，0-配置区，1-用户区 |
| | data[in] | 要写的数据，0~253 字节 |
| 返回值 | 0-成功<br>非 0-错误码 | |
| 备注 | 1.验密通过后才可写相应的区<br>2.区号为 7 的区验密通过后可以写配置区 | |

#### 3.4.1.13.5 AT88SC 获取剩余可认证次数

| | | |
|---|---|---|
| 原型 | int at88scGetRemainAuthCount(int rwFlag, int zoneNo) | |
| 功能 | AT88SC 获取区域剩余可验证密码的次数 | |
| 参数 | rwFlag[in] | 读写标志，0-写密码，1-读密码 |
| | zoneNo[in] | 用户区域编号，范围 0~7 |
| 返回值 | >=0-剩余认证次数<br><0-错误码 | |
| 备注 | 当剩余认证次数为 0 时，卡片被锁，无法进行密码验证或写数据操作 | |

### 3.4.1.14 透传 APDU 指令（扩展）

| | | |
|---|---|---|
| 原型 | int transmitApduEx(int cardType, byte[] sendBuff, byte[] recvBuff) | |
| 功能 | **以透传数据的方式**与接触式 IC 卡之间使用接口协议（T=0 及 T=1）进行数据交互；<br>与非接触式 IC 卡之间使用 T=CL 协议进行数据交互。 | |
| 参数 | cardTypy[in] | 当前操作的卡片类型 |
| | sendBuff [in] | 要透传给卡片的数据，最大为 255B |
| | recvBuff [out] | 卡片的应答数据，应答数据的最大长度为 255B，recvBuff.length>=255 |
| 返回值 | >=0- recvBuff 中有效数据的长度<br><0-错误码 | |
| 备注 | **本接口与 transmitApdu()仅在卡类型为 Mifare 时有区别**，详述如下：<br>1.当卡类型为 **Mifare** 时，发送的第一字节(B1)表示通信参数的设置：<br>Bit0: 1-enable rx crc, 0-disable rx crc<br>Bit1: 1-enable tx crc, 0-disable tx crc<br>Bit2: 0-enable rx parity, 1-disable rx parity<br>Bit3: 0-enable tx parity, 1-disable tx parity<br>Bit4-bit7: TxLastBits, TxLastBits=0-发送最后一字节的全部数据，TxLastBits=n（n≠0）-发送最后一字节的 n bit.<br>2.**transmitApdu 接口中不需要传入 B1，SDK 默认发送 0x03+sendBuff 给卡片**<br>3.**transmitApduEx 接口中需要传入 B1，sendBuff[0]为 B1，B1 的规则参照 1 中的描述** | |

### 3.4.1.15 CTX512B

#### 3.4.1.15.1 CTX512B 读块数据

| | | |
|---|---|---|
| 原型 | int ctx512ReadBlock(int block, byte[] outData) | |
| 功能 | CTX512B 卡读块数据 | |
| 参数 | block[in] | 待读取的块号 |
| | outData[out] | 缓存区，存储读取到块数据（2B） |
| 返回值 | >=0-outData 中有效数据的长度<br><0-错误码 | |
| 备注 | | |

#### 3.4.1.15.2 CTX512B 写块数据

| | |
|---|---|
| 原型 | int ctx512WriteBlock(int block, byte[] data) |

| | | |
|---|---|---|
| 功能 | CTX512B 卡写块数据 | |
| 参数 | block[in] | 待写入的块号 |
| | data[in] | 待写入的数据（2B） |
| 返回值 | 0-写入数据成功<br>非 0-错误码 | |
| 备注 | | |

### 3.4.1.15.3 CTX512 更新块数据

| | | |
|---|---|---|
| 原型 | int ctx512UpdateBlock(int block, byte[] data) | |
| 功能 | CTX512B 卡更新块数据 | |
| 参数 | block[in] | 待更新的块号 |
| | data[in] | 待更新的数据（2B） |
| 返回值 | 0-更新数据成功<br>非 0-错误码 | |
| 备注 | | |

### 3.4.1.15.4 CTX512B 获取签名数据

| | | |
|---|---|---|
| 原型 | int ctx512GetSignature(int block, byte[] random, byte[] outData) | |
| 功能 | CTX512B 卡获取签名数据 | |
| 参数 | block[in] | 块号 |
| | random [in] | 随机数据(6B) |
| | outData[out] | 缓存区，存储获取到的签名数据（2B） |
| 返回值 | >=0-outData 中有效数据的长度<br><0-错误码 | |
| 备注 | | |

### 3.4.1.15.5 CTX512B 读连续的 4 个块数据

| | | |
|---|---|---|
| 原型 | int ctx512MultiReadBlock(int startBlock, byte[] outData) | |
| 功能 | CTX512B 卡读连续的 4 个块数据 | |
| 参数 | startBlock [in] | 起始块号 |
| | outData[out] | 缓存区，存储读取到块数据（8B） |
| 返回值 | >=0-outData 中有效数据的长度<br><0-错误码 | |
| 备注 | | |

### 3.4.1.16 检卡（扩展）

| | |
|---|---|
| 原型 | void checkCardEx(int cardType, int ctrCode, int stopOnError, CheckCardCallbackV2 checkCardCallback, int timeout) |
| 功能 | 针对各类卡的检卡，支持磁条卡，IC 卡和非接卡，检卡完成后，会将卡片类型放到 |

<table>
<tr><td></td><td colspan="2">CheckCardCallbackV2 中。</td></tr>
<tr><td rowspan="5">参数</td><td>cardType[in]</td><td>卡类型组合，支持同时检多种卡，值为 CardType.value 的组合。例如同时检磁条、NFC 和 IC 卡可传入值：<br>CardType. MAGNETIC.getValue() | CardType. NFC.getValue() | CardType. IC.getValue()</td></tr>
<tr><td>ctrCode[in]</td><td>卡片激活控制码,默认值 0，格式如下：<br>Bit0-bit1：接触卡工作电压<br>0：VCC_3000mV<br>1：VCC_1800mV<br>2：VCC_5000mV<br>3：预留<br>Bit2：接触 CPU 卡及 SAM 卡上电复位速率<br>0：SPD_1X<br>1：SPD_4X<br>Bit3： pps 是否支持<br>0：NOT SUPPORT<br>1：SUPPORT<br>Bit4：接触 CPU 卡及 SAM 卡协议流程<br>0：ICC_SPEC<br>1：ICC_EMV</td></tr>
<tr><td>stopOnError[in]</td><td>是否出错即停止（cardType 为复合类型时有效），0-不停止，1-停止</td></tr>
<tr><td>callback[in]</td><td>检卡回调，详见 CheckCardCallbackV2</td></tr>
<tr><td>timeout[in]</td><td>超时时间，（单位为秒） 参数取值范围：1-120（秒）</td></tr>
<tr><td>返回值</td><td colspan="2">None</td></tr>
<tr><td>备注</td><td colspan="2">不区分银行卡和非银行卡</td></tr>
</table>

### 3.4.1.17 透传 APDU 指令（二次扩展）

<table>
<tr><td>原型</td><td colspan="2">int transmitApduExx(int cardType, int ctrCode, byte[] sendBuff, byte[] recvBuff);</td></tr>
<tr><td>功能</td><td colspan="2">以透传数据的方式与接触式 IC 卡之间使用接口协议（T=0 及 T=1）进行数据交互；<br>与非接触式 IC 卡之间使用 T=CL 协议进行数据交互。</td></tr>
<tr><td rowspan="2">参数</td><td>cardTypy[in]</td><td>当前操作的卡片类型</td></tr>
<tr><td>ctrCode[in]</td><td>卡片数据交互控制码，说明如下：<br>Bit0-bit3: 设置非接 CPU 卡 apdu 帧等待时间（fwi）
<table>
<tr><th>fwi(bit0–bit3)</th><th>等待时间<br>(单位: ms)</th><th>fwi(bit0–bit3)</th><th>等待时间<br>(单位: ms)</th></tr>
<tr><td>0x0-0x3</td><td>卡指定时间</td><td>0xA</td><td>309</td></tr>
<tr><td>0x4</td><td>4.832</td><td>0xB</td><td>618.5</td></tr>
<tr><td>0x5</td><td>9.664</td><td>0xC</td><td>1237</td></tr>
<tr><td>0x6</td><td>19.3</td><td>0xD</td><td>2474</td></tr>
<tr><td>0x7</td><td>38.7</td><td>0xE</td><td>4948</td></tr>
<tr><td>0x8</td><td>77.3</td><td>Other</td><td>卡指定时间</td></tr>
</table></td></tr>
</table>

<table>
<tr><td></td><td></td><td>
<table>
<tr><td>0x9</td><td>154.3</td><td></td><td></td></tr>
</table>
<br>Bit4-bit5：非接 cpu 卡 apdu 重试次数<br>0：不重试<br>1：重试 1 次<br>2：重试 2 次<br>3：预留
</td></tr>
<tr><td></td><td>sendBuff [in]</td><td>要透传给卡片的数据，最大为 255B</td></tr>
<tr><td></td><td>recvBuff [out]</td><td>卡片的应答数据，应答数据的最大长度为 255B，recvBuff.length>=255</td></tr>
<tr><td>返回值</td><td colspan="2">>=0- recvBuff 中有效数据的长度<br><0-错误码</td></tr>
<tr><td>备注</td><td colspan="2">当卡类型为 Mifare 时，参数 sendBuff 的格式与 transmitApduEx()中 sendBuff 的格式相同</td></tr>
</table>

### 3.4.1.18 透传多条 APDU 指令

<table>
<tr><td>原型</td><td colspan="2">int transmitMultiApdus(int cardType, int ctrCode, List<String> sendList, List<String>recvList)</td></tr>
<tr><td>功能</td><td colspan="2">以透传数据的方式与接触式 IC 卡之间使用接口协议（T=0 及 T=1）进行数据交互；<br>与非接触式 IC 卡之间使用 T=CL 协议进行数据交互。单次透传的 APDU 最大条数为 8 条</td></tr>
<tr><td>参数</td><td>cardTypy[in]</td><td>当前操作的卡片类型</td></tr>
<tr><td></td><td>ctrCode[in]</td><td>卡片数据交互控制码，说明如下：<br>Bit0-bit3: 设置非接 CPU 卡 apdu 帧等待时间（fwi）
<table>
<tr><td>fwi(bit0–bit3)</td><td>等待时间<br>(单位: ms)</td><td>fwi(bit0–bit3)</td><td>等待时间<br>(单位: ms)</td></tr>
<tr><td>0x0-0x3</td><td>卡指定时间</td><td>0xA</td><td>309</td></tr>
<tr><td>0x4</td><td>4.832</td><td>0xB</td><td>618.5</td></tr>
<tr><td>0x5</td><td>9.664</td><td>0xC</td><td>1237</td></tr>
<tr><td>0x6</td><td>19.3</td><td>0xD</td><td>2474</td></tr>
<tr><td>0x7</td><td>38.7</td><td>0xE</td><td>4948</td></tr>
<tr><td>0x8</td><td>77.3</td><td>Other</td><td>卡指定时间</td></tr>
<tr><td>0x9</td><td>154.3</td><td></td><td></td></tr>
</table>
<br>Bit4-bit5：非接 cpu 卡 apdu 重试次数<br>0：不重试<br>1：重试 1 次<br>2：重试 2 次<br>3：预留</td></tr>
<tr><td></td><td>sendList[in]</td><td>要透传给卡片的 APDU 列表(Hex list)，list 中每个 item 代表一条 APDU</td></tr>
<tr><td></td><td>recvList[out]</td><td>卡片的应答数据(Hex list)，list 中每个 item 代表一条应答数据</td></tr>
<tr><td>返回值</td><td colspan="2">>=0- 成功<br><0-错误码</td></tr>
<tr><td>备注</td><td colspan="2">当卡类型为 Mifare 时，参数 sendBuff 的格式与 transmitApduEx()中 sendBuff 的格式相同</td></tr>
</table>

### 3.4.1.19 磁道加密检卡

| | | |
|---|---|---|
| 原型 | int checkCardEnc(Bundle bundle, CheckCardCallbackV2 checkCardCallback, int timeout) | |
| 功能 | 针对各类卡的检卡，支持磁条卡，IC 卡和非接卡，检卡完成后，会将卡片类型放到 CheckCardCallbackV2 中。<br>**当卡类型为 CardType.MAGNETIC.getValue()时，checkCardCallback 返回密文磁道数据** | |
| 参数 | bundle[in] | 检卡参数，包含如下 key：<br>cardType: int，卡类型,同时支持 NFC,IC,MAG 卡检卡<br>encKeySystem :int, 密钥体系，参见 密钥体系常量<br>encKeyIndex: int，磁道数据加密索引，一般传入 TDK 索引<br>encMode: int，加密模式<br>encIv: byte[]，初始向量，加密模式为 ECB 传空，为其他加密模式传入 8 字节向量<br>encPaddingMode: byte，磁道数据进行 DES 加密时，长度不是 8 的倍数，则在后面补齐 EncPaddingMode 至长度为 8 的倍数的数据<br>encMaskStart: int，表示账号前 EncMaskStart 位为明文，范围是 0~6<br>encMaskEnd: int，表示账号后 EncMaskEnd 位为明文，范围是 0~4<br>encMaskWord: char，为 0 或者是非数字字符，表示账号 EncMaskStart 至 encMaskWord 为掩码,默认为 '*'<br>ctrCode: int，卡片激活控制码,默认值 0<br>b0~b1: 接触卡工作电压：<br>00-VCC_3000m<br>01-VCC_1800mV<br>02-VCC_5000mV<br>03-预留<br>b2: 接触 CPU 卡及 SAM 卡上电复位速率<br>0-SPD_1X<br>1-SPD_4X<br>b3: 是否支持 PPS<br>0-不支持<br>1-支持<br>b4: 接触 CPU 卡及 SAM 卡协议流程<br>0-ICC_SPEC<br>1-ICC_EMV<br>stopOnError: int，是否出错即停：0-不停止，1-停止 |
| | callback[in] | 检卡回调，详见 CheckCardCallbackV2 |
| | timeout[in] | 超时时间，（单位为秒） 参数取值范围：1-120（秒） |
| 返回值 | None | |
| 备注 | 不区分银行卡和非银行卡 | |

### 3.4.1.20 设置智能卡相关参数

| | |
|---|---|
| 原型 | int smartCardIoControl(int cardType, int ctrCode, byte[] dataIn, byte[] dataOut) |

| 功能 | 设置智能卡相关参数 | |
|---|---|---|
| 参数 | cardType[in] | 卡类型 |
| | ctrCode[in] | 控制码，取值如下：<br>0-设置 Felica 卡轮询指令的系统码(默认为 0xffff)，2 字节，高位在前<br>1-设置非接指令交互的超时时间，单位：ms，4 字节，高位在前<br>2-获取非接寄存器配置(TLV 格式)<br>3-设置非接寄存器配置(TLV 格式)<br>4-获取非接参数配置(TLV 格式)<br>5-设置非接参数配置(TLV 格式 |
| | dataIn[in] | 输入数据 |
| | dataOut[out] | 输出数据 |
| 返回值 | >=0- dataOu 中有效数据的长度<br><0-错误码 | |
| 备注 | | |

## 3.4.2 CheckCardCallbackV2 检卡回调对象

### 3.4.2.1 寻到磁条卡

| 原型 | void findMagCard(Bundle info) | |
|---|---|---|
| 功能 | 检到磁条卡 | |
| 参数 | Info[in] | 包含如下数据：<br>cardType：int，卡片类型<br>TRACK1：Strin，磁道 1 数据<br>TRACK2：String，磁道 2 数据<br>TRACK3：String，磁道 3 数据<br>pan：String，PAN 数据（**checkCardEnc()**返回）<br>name：String，持卡人姓名（**checkCardEnc()**返回）<br>expire：String，卡片有效期（**checkCardEnc()**返回）<br>servicecode: String，卡片服务码（**checkCardEnc()**返回)<br>track1ErrorCode：int，磁道 1 错误码<br>track2ErrorCode：int，磁道 2 错误码<br>track3ErrorCode：int，磁道 3 错误码<br>磁道错误码取值如下：<br>-1：磁道无数据<br>-2：磁道奇偶校验错<br>-3：磁道 LRC 校验错 |
| 返回值 | None | |
| 备注 | None | |

#### 3.4.2.2 寻到 IC 卡

| 原型 | void findICCard(String atr) | |
|---|---|---|
| 功能 | 检到 IC 卡 | |
| 参数 | atr[in] | 卡片的 ATR |
| 返回值 | None | |
| 备注 | None | |

#### 3.4.2.3 寻到非接卡

| 原型 | void findRFCard(String uuid) | |
|---|---|---|
| 功能 | 检到非接卡 | |
| 参数 | uuid[in] | 卡片的 UUID |
| 返回值 | None | |
| 备注 | None | |

#### 3.4.2.4 检卡错误

| 原型 | void onError(int code, string message) | |
|---|---|---|
| 功能 | 检卡错误回调 | |
| 参数 | code[in] | 错误码 |
| | message[in] | 错误信息 |
| 返回值 | None | |
| 备注 | None | |

#### 3.4.2.5 寻到 IC 卡（扩展）

| 原型 | void findICCardEx(Bundle info) | |
|---|---|---|
| 功能 | 检到 IC 卡 | |
| 参数 | info[in] | 包含如下数据：<br>cardType：int，检到的卡类型<br>atr：String，卡片的 ATR |
| 返回值 | None | |
| 备注 | 本方法提供比 **findICCard()**更详细的信息，检卡时本方法和 **findICCard()**都会被回调，客户端可根据需求，选择实现两者中的一者即可 | |

#### 3.4.2.6 寻到非接卡（扩展）

| 原型 | void findRFCardEx(in Bundle info) | |
|---|---|---|
| 功能 | 检到非接卡 | |
| 参数 | info[in] | 包含如下数据：<br>cardType：int，检到的卡类型<br>uuid：String，卡片的 UUID<br>ats：String，卡片的 ATS<br>cardCategory：int，卡片类别（ ‘A’ 或 ‘B’ ）<br>atqa；byte[]，卡片的 ATQA |
| 返回值 | None | |
| 备注 | 本方法提供比 **findRFCard()**更详细的信息，检卡时本方法和 **findRFCard()**都会被回调，客户端可根据需求，选择实现两者中的一者即可 | |

#### 3.4.2.7 检卡错误（扩展）

| 原型 | void onErrorEx(Bundle info) | |
|---|---|---|
| 功能 | 检卡错误 | |
| 参数 | info [in] | 包含如下数据：<br>cardType：int，当前出错的卡类型<br>code：int，错误码<br>message：String，错误信息 |
| 返回值 | None | |
| 备注 | 本方法提供比 **onError()**更详细的信息，检卡时本方法和 **onError()**都会被回调，客户端可根据需求，选择实现两者中的一者即可 | |

## 3.5 密码键盘模块

### 3.5.1 PinPadOpt 密码键盘对象

#### 3.5.1.1 初始化密码键盘

| 原型 | void initPinPad(PinPadConfigV2 config, PinPadListenerV2 listener) | |
|---|---|---|
| 功能 | 初始化密码键盘，包含传入初始化配置参数，回调函数 | |
| 参数 | config[in] | 密码键盘配置，详见 PinPadConfigV2 |
| | listerner[in] | 输 PIN 回调，详见 PinPadListenerV2 |
| | [out] | None |
| 返回值 | None | |

| | |
|---|---|
| 备注 | None |

### 3.5.1.2 取消输 PIN

| | | |
|---|---|---|
| 原型 | void cancelInputPin() | |
| 功能 | 取消输 PIN | |
| 参数 | [in] | None |
| | [out] | None |
| 返回值 | None | |
| 备注 | None | |

### 3.5.1.3 设置密码键盘显示的文字

| | | |
|---|---|---|
| 原型 | void setPinPadText(PinPadTextConfigV2 config) | |
| 功能 | 设置密码键盘显示的文字 | |
| 参数 | config[in] | 文字配置，详见 PinPadTextConfigV2 |
| | [out] | None |
| 返回值 | None | |
| 备注 | None | |

### 3.5.1.4 设置密码键盘模式

| | | |
|---|---|---|
| 原型 | int setPinPadMode(Bundle bundle) | |
| 功能 | 设置密码键盘模式 | |
| 参数 | bundle [in] | 密码键盘模式参数，包含如下 key：<br>normal：int，普通模式（0-关闭，1-开启。本模式和其他模式互斥）<br>longPressToClear：int，长按 Clear 按钮清除输入的内容（0-关闭，1-开启）<br>silent：int，输 PIN 时静音（0-关闭，1-开启）<br>greenLed：int，输 PIN 时 LED 绿灯亮（0-关闭，1-开启） |
| | [out] | None |
| 返回值 | 0-成功<br>非 0-错误码 | |
| 备注 | 1.默认模式为**普通模式**（有按键音+短按 Clear 按钮清除+LED 绿灯亮）<br><br>2.**普通(normal)模式**和其他模式互斥，若 normal 值为 1，则其他参数无效<br><br>3.调用本接口仅对下次输 PIN 有效，输完 PIN 后自动恢复为**普通模式** | |

### 3.5.1.5 获取密码键盘模式

| | | |
|---|---|---|
| 原型 | int getPinPadMode(Bundle bundle) | |
| 功能 | 获取当前密码键盘模式 | |
| 参数 | [in] | None |
| | bundle [out] | 密码键盘模式参数，包含如下 key：<br>normal：int，普通模式（0-关闭，1-开启）<br>longPressToClear：int，长按 Clear 按钮清除输入的内容（0-关闭，1-开启）<br>silent：int，输 PIN 时静音（0-关闭，1-开启）<br>greenLed：int，输 PIN 时 LED 绿灯亮（0-关闭，1-开启） |
| 返回值 | 0-成功<br>非 0-错误码 | |
| 备注 | | |

### 3.5.1.6 初始化密码键盘（扩展）

| | | |
|---|---|---|
| 原型 | String initPinPadEx(Bundle config, PinPadListenerV2 listener) | |
| 功能 | 初始化密码键盘 | |
| 参数 | config[in] | 密码键盘配置，包含如下 key：<br>pinPadType: 密码键盘类型(类型 int，0-预置密码键盘(由服务实现样式统一的键盘,默认值) ，1-客户端自定义的密码键盘)<br>pinType：PIN 类型标识(类型 int，0-联机 PIN，1-脱机 PIN)<br>isOrderNumKey: 是否顺序键盘(类型 int，0-乱序键盘(默认值)，1-顺序键盘)<br>pan：PAN 数据，ASCII 格式转换成的 byte 例如“123456”.getBytes("US-ASCII")(类型 byte[])<br>pinKeyIndex：PIK(PIN key)索引(类型 int)<br>minInput：最小输入位数(类型 int，默认 0)<br>maxInput：最大输入位数(类型 int，默认 6)<br>inputStep：PIN 步长(类型 int，默认 1)<br>timeout：超时时间，单位：ms(类型 int，默认 60000)<br>isSupportbypass :是否支持 bypass PIN(类型 int ,0-不支持 ,1-支持(默认值))<br>pinblockFormat：PinBlock 格式(类型 int，默认 0)，参考 PinBlock 格式<br>algorithmType：加密 Pin 的算法类型(类型 int，0-3DES(返回 8 字节 PinBlock),1-SM4(返回 16 字节 PinBlock),2-AES(返回 16 字节 PinBlock))<br>keySystem：密钥体系(类型 int, 0-SEC_MKSK(默认值), 1-SEC_DUKPT)<br>diversify：分散因子(类型 byte[]) ，3DES 密钥计算 PinBlock 前使用分散因子与 PIK 运算产生新的 PIK，并使用新 PIK 来计算 PinBlock |
| | listerner[in] | 输 PIN 回调，详见 PinPadListenerV2 |
| | [out] | None |
| 返回值 | None | |
| 备注 | None | |

#### 3.5.1.7 设置 PIN 防穷举保护模式

| 原型 | int setAntiExhaustiveProtectionMode(int level) | |
|---|---|---|
| 功能 | 设置 PIN 防穷举保护模式 | |
| 参数 | level[in] | 等级，范围 1-5，1-2min 4 次，2-6min 12 次，3-15min 30 次，4-30min 60 次，5-60min 120 次 |
| 返回值 | >=0-距离新周期生效需等待的时间，单位：min<br><0-错误码 | |
| 备注 | | |

#### 3.5.1.8 获取 PIN 防穷举保护模式

| 原型 | int getAntiExhaustiveProtectionMode() | |
|---|---|---|
| 功能 | 获取 PIN 防穷举保护模式 | |
| 参数 | [in] | None |
| 返回值 | >=0-当前生效的模式，范围 1-5<br><0-错误码 | |
| 备注 | | |

### 3.5.2 PinPadListenerV2 密码键盘回调

**注意，该接口为 AIDL 间传递的接口，传递该回调的时候，必须按照 aidl 接口实现。**

#### 3.5.2.1 点击数字

| 原型 | void onPinLength(int len) | |
|---|---|---|
| 功能 | 传入当前输入的位数，用于输入密码时屏幕显示星号 | |
| 参数 | len[in] | 已输入位数 |
| 返回值 | None | |
| 备注 | None | |

#### 3.5.2.2 返回 PinBlock

| 原型 | void onConfirm(int type, byte[] pinBlock) |
|---|---|
| 功能 | 按下确认键，返回 PinBlock.<br>（1）type = 0(联机 PIN)：<br>（a）pinBlock=null，bypass 模式（用户未输 pin 并直接按下确认键）<br>（b）pinBlock！=null，PinBlock 数据，长度为 8B（3DES）或 16B（SM4/AES）<br>（2）type = 1（脱机 PIN）,pinblock 长度为 1 且 pinblock[0]=0，pinBlock 在该情况下无意义 |

| | | |
|---|---|---|
| 参数 | type[in] | type = 0,表示联机；type = 1 表示脱机 pin |
| | pinBlock [out] | pinBlock 数据 |
| 返回值 | None | |
| 备注 | **脱机 PIN 由卡片直接验证，SDK 无需返回 PinBlock，SDK 返回长度为 1 的字节数组仅用于标记这种情况** | |

#### 3.5.2.3 点击密码键盘取消键

| | | |
|---|---|---|
| 原型 | void onCancel () | |
| 功能 | 点击取消 | |
| 参数 | [in] | None |
| 返回值 | None | |
| 备注 | None | |

#### 3.5.2.4 输 PIN 出错

| | | |
|---|---|---|
| 原型 | void onError (int errorCode) | |
| 功能 | 输 PIN 出错回调 | |
| 参数 | errorCode [in] | ErrorCode 返回错误码 详见错误码定义 |
| 返回值 | None | |
| 备注 | None | |

## 3.6 安全模块

### 3.6.1 保存明文密钥

| | | |
|---|---|---|
| 原型 | int savePlaintextKey(int keyType, byte[] keyValue, byte[] checkValue, int keyAlgType, int keyIndex) | |
| 功能 | 保存明文密钥 | |
| 参数 | keyType[in] | 密钥类型，参考附录：Aidl 常量类.密钥类型定义 |
| | keyValue [in] | 密钥数据 |
| | checkValue[in] | 密钥校验值 如果为 null 则不校验（当 keyAlgType 为 KEY_ALG_TYPE_SM4 时，checkValue 的值为 16 字节） |
| | keyAlgType[in] | 用于指定当前需要保存的密钥算法类型，参考附录：Aidl 常量类.密钥算法类型 |
| | keyIndex[in] | 保存的索引 范围：0~199 |
| 返回值 | 0-成功<br>非 0-错误码 | |
| 备注 | 密钥索引需要用户保证唯一，比如：先在索引 4 存储 MAC 密钥后再次在索引 4 存储磁道密钥，那么 MAC 密钥将会被磁道密钥覆盖 | |

### 3.6.2 保存密文密钥

| | | |
|---|---|---|
| 原型 | int saveCiphertextKey(int keyType, byte[] keyValue, byte[] checkValue, int encryptIndex, int keyAlgType, int keyIndex) | |
| 功能 | 保存密文密钥 | |
| 参数 | keyType[in] | 密钥类型 ,参考附录 :Aidl 常量类.密钥类型定义 ,不能密文保存 KEY_TYPE_KEK 类型 |
| | keyValue[in] | 密钥数据 |
| | checkValue[in] | 密钥校验值 如果为 null 则不校验（当 keyAlgType 为 KEY_ALG_TYPE_SM4 时，checkValue 的值为 16 字节） |
| | encryptIndex[in] | 用于解密密钥密文的索引，注意，这里是 TMK 的索引 |
| | keyAlgType[in] | 用于指定当前需要保存的密钥算法类型，参考附录：Aidl 常量类.密钥算法类型 |
| | keyIndex[in] | 保存的索引 范围：0~199 |
| 返回值 | 0-成功<br>非 0-错误码 | |
| 备注 | 保存 TMK 时可以指定解密的 TMK 索引，这样可以支持多层 TMK<br>密钥索引需要用户保证唯一，比如：先在索引 4 存储 MAC 密钥后再次在索引 4 存储磁道密钥，那么 MAC 密钥将会被磁道密钥覆盖 | |

### 3.6.3 计算 mac 值

| | | |
|---|---|---|
| 原型 | int calcMac (int keyIndex, int macType , byte[] dataIn, byte[] dataOut) | |
| 功能 | 实现数据 MAC 计算 | |
| 参数 | keyIndex[in] | mac 密钥索引(范围：0~199) |
| | macType[in] | mac 加密算法，参考附录：Aidl 常量.MAC 算法类型 |
| | dataIn[in] | 用于进行 mac 计算的源数据 |
| | dataOut[out] | 计算生成的 mac 值 |
| 返回值 | >=0-计算完成的 mac 数据长度<br><0-错误码，详见 AidlErrorCode | |
| 备注 | dataIn 为待计算的报文数据，SDK 内部已经实现了多种 MAC 算法，只需要把待计算 MAC 的数据源传进来，结果将会通过 dataOut 返回，传进去的 dataOut 固定为 8 字节长的 byte 如: byte[] dataOut = new byte[8]; | |

### 3.6.4 加密数据

| | | |
|---|---|---|
| 原型 | int dataEncrypt (int keyIndex , byte[] dataIn, int encryptionMode,byte[] iv, byte[] dataOut) | |
| 功能 | 数据加密功能 | |
| 参数 | keyIndex[in] | 用于加密的密钥索引(范围：0~199) |
| | dataIn[in] | 用于进行加密计算的源数据 |
| | encryptionMode[in] | 加密模式，参考附录：Aidl 常量.加密模式常量 |
| | Iv[in] | 初始向量，加密模式为 ECB 传空，为其他加密模式传入 8 字节向量 |
| | dataOut[out] | 计算生成的密文 |

| | |
|---|---|
| 返回值 | 0-成功<br>非 0-错误码 |
| 备注 | None |
| 备注 | 1.当 **encryptionMode** 为 DATA_MODE_OFB、DATA_MODE_CFB 时，dataIn 的长度可以为非 8、16 的整数倍，否则必须为 8、16 的整数倍<br>2.dataOut 的长度应和 dataIn 的长度相等<br>例如：byte[] dataIn = new byte[16];<br>那么 dataOut 的长度也应该为 16 字节 |

### 3.6.5 解密数据

| | | |
|---|---|---|
| 原型 | int dataDecrypt (int keyIndex , byte[] dataIn, int encryptionMode, byte[] iv, byte[] dataOut) | |
| 功能 | 数据解密功能 | |
| 参数 | keyIndex[in] | 用于加密的密钥索引(范围：0~199) |
| | dataIn[in] | 用于进行解密计算的源数据 |
| | encryptionMode[in] | 加密模式，参考附录：Aidl 常量.加密模式常量 |
| | iv[in] | 初始向量 ，加密模式为 ECB 传空，为其他加密模式传入 8 字节向量 |
| | dataOut[out] | 计算生成的明文 |
| 返回值 | 0-成功<br>非 0-错误码 | |
| 备注 | None | |
| 示例 | 1.当 **encryptionMode** 为 DATA_MODE_OFB、DATA_MODE_CFB 时，dataIn 的长度可以为非 8、16 的整数倍，否则必须为 8、16 的整数倍<br>2.dataOut 的长度应和 dataIn 的长度相等<br>例如：byte[] dataIn = new byte[16];<br>那么 dataOut 的长度也应该为 16 字节 | |

### 3.6.6 注入 Dukpt 密钥

| | | |
|---|---|---|
| 原型 | Int saveKeyDukpt(int keyType,byte[] keyValue, byte[] checkValue,byte[] ksn,int encryptType,int keyIndex) | |
| 功能 | 注入 Dukpt 密钥 | |
| 参数 | keyType[in] | 密钥类型。包含：<br>BDK-基础分散密钥:对应 Aidl 常量.密钥类型常量 KEY_TYPE_DUPKT_BDK<br>IPEK-初始 PIN 加密密钥：对应 Aidl 常量.密钥类型常量 KEY_TYPE_DUPKT_IPEK |
| | keyValue[in] | 密钥数据 |
| | checkValue[in] | 密钥校验值（当注入 BDK 传空） |
| | ksn[in] | KSN |
| | encryptType[in] | 密钥算法 |
| | keyIndex[in] | 保存的索引（范围为 0-9，1100~1199) |

| | |
|---|---|
| 返回值 | 0-成功<br><0-错误码 |
| 备注 | None |
| 示例 | None |

### 3.6.7 Dukpt 密钥计算 mac

| | | |
|---|---|---|
| 原型 | int calcMacDukpt(int keyIndex, int macType , byte[] dataIn, byte[] dataOut) | |
| 功能 | dukpt 密钥计算 mac | |
| 参数 | keyIndex [in] | 密钥索引(范围为 3DES：0-9 或 1100-1199，AES：10-19 或 2100-2199) |
| | macType [in] | mac 算法，参考附录：Aidl 常量.MAC 算法类型 |
| | dataIn [in] | 待计算的 mac 数据 |
| | dataOut[out] | mac 结果 |
| 返回值 | 0-成功<br><0-错误码 | |
| 备注 | 1.dataIn 为待计算的报文数据，SDK 内部已经实现了多种 MAC 算法，只需要把待计算 MAC 的数据源传进来，结果将会通过 dataOut 返回，传进去的 dataOut 固定为 8 字节长的 byte 如: byte[] dataOut = new byte[8];<br>2. keyIndex 范围为 1100-1199(Dukpt 扩展密钥)时，dataIn 长度<=1016 | |

### 3.6.8 Dukpt 密钥加密数据

| | | |
|---|---|---|
| 原型 | int dataEncryptDukpt(int keyIndex, byte[] dataIn, int encryptionMode, byte[] iv, byte[] dataOut) | |
| 功能 | Dukpt 密钥加密数据 | |
| 参数 | keyIndex [in] | 密钥索引(范围为 3DES：0-9 或 1100-1199，AES：10-19 或 2100-2199) |
| | dataIn[in] | 待加密数据 |
| | encryptionMode[in] | 加密模式，参考附录：Aidl 常量.加密模式常量 |
| | iv[in] | 初始向量 ，加密模式为 ECB 传空，为其他加密模式传入 8 字节向量 |
| | dataOut[out] | 加密结果 |
| 返回值 | 0-成功<br><0-错误码 | |
| 备注 | None | |
| 示例 | 1.当 **encryptionMode** 为 DATA_MODE_OFB、DATA_MODE_CFB 时，dataIn 的长度可以为非 8、16 的整数倍，否则必须为 8、16 的整数倍<br>2.dataOut 的长度应和 dataIn 的长度相等<br>例如：byte[] dataIn = new byte[16];<br>那么 dataOut 的长度也应该为 16 字节<br>3. keyIndex 范围为 1100-1199(Dukpt 扩展密钥)时，dataIn 长度<=1024 | |

### 3.6.9 Dukpt 密钥解密数据

| | | |
|---|---|---|
| 原型 | int dataDecryptDukpt(int keyIndex, byte[] dataIn, int encryptionMode, byte[] iv, byte[] dataOut) | |
| 功能 | Dukpt 密钥解密数据 | |
| 参数 | keyIndex[in] | 密钥索引(范围为 3DES：0-9 或 1100-1199，AES：10-19 或 2100-2199) |
| | dataIn[in] | 待解密数据 |
| | encryptionMode[in] | 加密模式，参考附录：Aidl 常量.加密模式常量 |
| | iv[in] | 初始向量 ，加密模式为 ECB 传空，为其他加密模式传入 8 字节向量 |
| | dataOut[out] | 解密结果 |
| 返回值 | 0-成功<br><0-错误码 | |
| 备注 | None | |
| 示例 | 1.当 **encryptionMode** 为 DATA_MODE_OFB、DATA_MODE_CFB 时，dataIn 的长度可以为非 8、16 的整数倍，否则必须为 8、16 的整数倍<br>2.dataOut 的长度应和 dataIn 的长度相等<br>例如：byte[] dataIn = new byte[16];<br>那么 dataOut 的长度也应该为 16 字节<br>3. keyIndex 范围为 1100-1199(Dukpt 扩展密钥)时，dataIn 长度<=1024 | |

### 3.6.10 Dukpt 的 KSN 自增 1

| | | |
|---|---|---|
| 原型 | int dukptIncreaseKSN(int keyIndex) | |
| 功能 | Dukpt KSN 增加 1 | |
| 参数 | keyIndex[in] | dukpt 密钥索引(范围为 3DES：0-9 或 1100-1199，AES：10-19 或 2100-2199) |
| 返回值 | 0-成功<br><0-错误码 | |
| 备注 | **客户端应检查本方法的返回值，当返回值小于 0 时应再次调用本方法，直到返回值为 0，防止由于 KSN 自增失败引起下次调用 dukpt 功能失败** | |
| 示例 | None | |

### 3.6.11 Dukpt 获取当前 KSN

| | | |
|---|---|---|
| 原型 | int dukptCurrentKSN(int keyIndex, byte[] outKSN) | |
| 功能 | Dukpt 获取当前的 KSN | |
| 参数 | keyIndex[in] | dukpt 密钥索引(范围为 3DES :0-9 或 1100-1199 ,AES :10-19 或 2100-2199) |
| | outKSN[out] | buffer，存放获取到的 KSN。Dukpt-3DES KSN 长度为 10 字节，dukpt-AES KSN 长度为 12 字节 |
| 返回值 | 0-成功<br><0-错误码 | |

| | |
|---|---|
| 备注 | 用于 Dukpt 密钥体系密钥 |
| 示例 | None |

### 3.6.12 获取密钥 CheckValue

| | | |
|---|---|---|
| 原型 | int getKeyCheckValue(int keySystem, int keyIndex, byte[] dataOut) | |
| 功能 | 获取密钥的 checkValue | |
| 参数 | keySystem[in] | 密钥体系。参考附录：Aidl 常量类.密钥体系常量 |
| | keyIndex[in] | 密钥索引。keySystem 为 SEC_DUKPT 时，索引范围为 3DES：0-9 或 1100-1199，AES：10-19 或 2100-2199，keySystem 为 SEC_MKSK 时，索引范围为 0-199 |
| | dataOut[out] | 固定为 4 个字节 |
| 返回值 | 0-成功<br><0-错误码 | |
| 备注 | 通过注入 BDK 时设定的密钥索引得到的是 BDK 对应 IPEK 的 CheckValue | |
| 示例 | None | |

### 3.6.13 获取硬件序列号加密数据（仅用于国内市场）

| | | |
|---|---|---|
| 原型 | int getTUSNEncryptData (String dataIn, byte[] dataOut) | |
| 功能 | 获取密文硬件序列号 | |
| 参数 | dataIn[in] | 用于计算密文的分散值 |
| | dataOut[out] | 计算生成的密文 |
| 返回值 | 0-成功<br>非 0-错误码 | |
| 备注 | None | |
| 示例 | dataIn 加密随机因子（银行卡为卡号后六位，扫码类为码的后六位）<br>dataOut　固定传进去 4 个字节的 byte[]　如: byte[] dataOut = new byte[4]; | |

### 3.6.14 生成 RSA 公私钥对

| | | |
|---|---|---|
| 原型 | int generateRSAKeys(int pubKeyIndex, int pvtKeyIndex, int keysize, String pubExponent) | |
| 功能 | 生成 RSA 公私钥对 | |
| 参数 | pubKeyIndex [in] | 公钥保存的位置索引，范围 0-19 |
| | pvtKeyIndex [out] | 私钥保存的位置索引，范围 0-19 |
| | keysize [in] | 密钥长度（512~65536，单位：bit，必须为 64 的倍数，一般为 512、1024 等） |
| | pubExponent [in] | 公钥指数 |
| 返回值 | 0-成功<br>非 0-错误码 | |
| 备注 | None | |
| 示例 | None | |

### 3.6.15 获取 RSA 公钥

| 原型 | int getRSAPublicKey(int pubKeyIndex, byte[] dataOut) | |
|---|---|---|
| 功能 | 获取 ANS.1 X509 标准编码格式的 RSA 公钥 | |
| 参数 | pubKeyIndex [in] | 公钥的位置索引，范围 0-19 |
| | dataOut[out] | 出参 buffer，存放公钥数据 |
| 返回值 | >=0 - dataOut 中有效数据的长度<br><0-错误码 | |
| 备注 | None | |
| 示例 | None | |

### 3.6.16 RSA 算法加密数据

| 原型 | int dataEncryptRSA(String transformation, int keyIndex, byte[] dataIn, byte[] dataOut) | |
|---|---|---|
| 功能 | 用 RSA 算法加密数据 | |
| 参数 | transformation [in] | 算法/工作模式/填充模式，参见 Aidl 常量.RSA transformation |
| | keyIndex [in] | 加密密钥索引，范围 0-19 |
| | dataIn [in] | 待加密的数据 |
| | dataOut [out] | 出参 buffer，存放加密后的数据 |
| 返回值 | >=0 – dataOut 中有效数据的长度<br><0 - 错误码 | |
| 备注 | None | |
| 示例 | None | |

### 3.6.17 RSA 算法解密数据

| 原型 | int dataDecryptRSA(String transformation, int keyIndex, byte[] dataIn, byte[] dataOut) | |
|---|---|---|
| 功能 | 用 RSA 算法解密数据 | |
| 参数 | transformation [in] | 算法/工作模式/填充模式，参见 Aidl 常量.RSA transformation |
| | keyIndex [in] | 解密密钥索引，范围 0-19 |
| | dataIn [in] | 待解密的数据 |
| | dataOut [out] | 出参 buffer，存放解密后的数据 |
| 返回值 | >=0 - dataOut 中有效数据的长度<br><0 - 错误码 | |
| 备注 | None | |
| 示例 | None | |

### 3.6.18 移除 RSA 密钥

| 原型 | int removeRSAKey(int keyIndex) |
|---|---|
| 功能 | 移除 RSA 公/私钥 |

| | | |
|---|---|---|
| 参数 | keyIndex [in] | RSA 密钥存储位置索引，范围 0-19 |
| 返回值 | 0 - 成功<br>非 0 - 错误码 | |
| 备注 | None | |
| 示例 | None | |

### 3.6.19 保存数字证书

| | | |
|---|---|---|
| 原型 | int storeCertificate(int certIndex, byte[] certData) | |
| 功能 | 保存数字证书 | |
| 参数 | certIndex [in] | 证书存储位置索引，范围 0-19 |
| | certData [in] | 证书数据 |
| 返回值 | 0 - 成功<br>非 0 - 错误码 | |
| 备注 | None | |
| 示例 | None | |

### 3.6.20 获取数字证书

| | | |
|---|---|---|
| 原型 | int getCertificate(int certIndex, byte[] dataOut) | |
| 功能 | 获取保存的数字证书 | |
| 参数 | certIndex [in] | 证书索引，范围 0-19 |
| | dataOut [out] | 出参 buffer，存放证书数据 |
| 返回值 | >=0 – dataOut 中有效数据的长度<br><0 - 错误码 | |
| 备注 | None | |
| 示例 | None | |

### 3.6.21 获取初始 KSN

| | | |
|---|---|---|
| 原型 | int dukptGetInitKSN(byte[] outKSN) | |
| 功能 | 获取初始 ksn 数据 | |
| 参数 | outKSN [out] | 出参 buffer，存放初始 KSN 数据 |
| 返回值 | >=0 – outKSN 中有效数据的长度<br><0 - 错误码 | |
| 备注 | None | |
| 示例 | None | |

### 3.6.22 RSA 算法签名数据

| | |
|---|---|
| 原型 | int signingRSA(String signAlg, int pvtKeyIndex, byte[] dataIn, byte[] dataOut) |

| | | |
|---|---|---|
| 功能 | RSA 算法签名数据 | |
| 参数 | signAlg [in] | 签名算法，参见 Aidl 常量.RSA 签名算法 |
| | pvtKeyIndex [in] | RSA 私钥索引，范围 0-19 |
| | dataIn [in] | 待签名的数据 |
| | dataOut [out] | Buffer，存放签名后的数据 |
| 返回值 | >=0 – dataOut 中有效数据的长度<br><0 - 错误码 | |
| 备注 | None | |
| 示例 | None | |

### 3.6.23 RSA 算法验签

| | | |
|---|---|---|
| 原型 | int verifySignatureRSA(String signAlg, byte[] pubKey, byte[] srcData, byte[] signature) | |
| 功能 | 获取初始 ksn 数据 | |
| 参数 | signAlg [in] | 签名算法，参见 Aidl 常量.RSA 签名算法 |
| | pubKey [in] | RSA 公钥数据，ANS.1 X509 标准编码格式 |
| | srcData [in] | 签名前的数据（原始数据） |
| | signature [in] | 签名数据 |
| 返回值 | =0-成功，<0-错误码 | |
| 备注 | None | |
| 示例 | None | |

### 3.6.24 注入明文密钥

| | | |
|---|---|---|
| 原型 | int injectPlaintextKey(String targetPkgName, int keyType, byte[] keyValue, byte[] checkValue, int keyAlgType, int keyIndex) | |
| 功能 | 注入明文密钥 | |
| 参数 | targetPkgName | 目标 App（使用密钥的 App）的包名 |
| | keyType[in] | 密钥类型：参考附录：Aidl 常量类.密钥类型定义 |
| | keyValue [in] | 密钥数据 |
| | checkValue[in] | 密钥校验值 如果为 null 则不校验( 当 keyAlgType 为 KEY_ALG_TYPE_SM4 时，checkValue 的值为 16 字节） |
| | keyAlgType[in] | 密钥的算法类型，参考附录：Aidl 常量类.密钥算法类型 |
| | keyIndex[in] | 密钥索引，范围：0~199 |
| 返回值 | 0 - 成功<br>非 0 - 错误码 | |
| 备注 | 1.密钥索引要保证唯一性，不同的密钥应存在不同的索引。例如：先在索引 4 存储 MAC 密钥后再次在索引 4 存储磁道密钥，那么 MAC 密钥将会被磁道密钥覆盖<br>2.调用本接口的 App 和目标 App 必须具有相同的签名，否则目标 App 无法使用注入的密钥 | |

### 3.6.25 注入密文密钥

| | | |
|---|---|---|
| 原型 | int injectCiphertextKey(String targetPkgName, int keyType, in byte[] keyValue, in byte[] checkValue, int encryptIndex, int keyAlgType, int keyIndex) | |
| 功能 | 注入密文密钥 | |
| 参数 | targetPkgName | 目标 App（使用密钥的 App）的包名 |
| | keyType[in] | 密钥类型：参考附录：Aidl 常量类.密钥类型定义，不能密文保存 KEY_TYPE_KEK 类型 |
| | keyValue[in] | 密钥数据 |
| | checkValue[in] | 密钥校验值 如果为 null 则不校验( 当 keyAlgType 为 KEY_ALG_TYPE_SM4 时，checkValue 的值为 16 字节） |
| | encryptIndex[in] | 解密密钥的索引，注意，这里是 TMK 的索引 |
| | keyAlgType[in] | 密钥的算法类型，参考附录：Aidl 常量类.密钥算法类型 |
| | keyIndex[in] | 保存的索引，范围：0~199 |
| 返回值 | 0 - 成功<br>非 0 - 错误码 | |
| 备注 | 1.保存 TMK 时可以指定解密的 TMK 索引，这样可以支持多层 TMK<br>2.密钥索引要保证唯一性，比如：先在索引 4 存储 MAC 密钥后再次在索引 4 存储磁道密钥，那么 MAC 密钥将会被磁道密钥覆盖<br>3.调用本接口的 App 和目标 App 必须具有相同的签名，否则目标 App 无法使用注入的密钥 | |

### 3.6.26 Dukpt 密钥加密数据（扩展）

| | | |
|---|---|---|
| 原型 | int dataEncryptDukptEx(int keySelect, int keyIndex, byte[] dataIn, int encryptionMode, byte[] iv, byte[] dataOut) | |
| 功能 | Dukpt 密钥加密数据 | |
| 参数 | keySelect[in] | Dukpt 密钥选择，参见 Aidl 常量.Dukpt 密钥选择 |
| | keyIndex[in] | 密钥索引(范围为 3DES：0-9 或 1100-1199，AES：10-19 或 2100-2199) |
| | dataIn[in] | 待加密数据 |
| | encryptionMode[in] | 加密模式，参考附录：Aidl 常量.加密模式常量 |
| | iv[in] | 初始向量 ，加密模式为 ECB 传空，为其他加密模式传入 8 字节向量 |
| | dataOut[out] | 加密结果 |
| 返回值 | 0-成功<br><0-失败 | |
| 备注 | None | |
| 示例 | 1.当 **encryptionMode** 为 DATA_MODE_OFB、DATA_MODE_CFB 时，dataIn 的长度可以为非 8、16 的整数倍，否则必须为 8、16 的整数倍<br>2.dataOut 的长度应和 dataIn 的长度相等<br>例如：byte[] dataIn = new byte[16];<br>那么 dataOut 的长度也应该为 16 字节<br>3. keyIndex 范围为 1100-1199(Dukpt 扩展密钥)时，dataIn 长度<=1024 | |

### 3.6.27 Dukpt 密钥解密数据（扩展）

| | | |
|---|---|---|
| 原型 | int dataDecryptDukptEx(int keySelect, int keyIndex, byte[] dataIn, int encryptionMode, byte[] iv, byte[] dataOut) | |
| 功能 | Dukpt 密钥解密数据 | |
| 参数 | keySelect[in] | Dukpt 密钥选择，参见 Aidl 常量.Dukpt 密钥选择 |
| | keyIndex[in] | 密钥索引(范围为 3DES：0-9 或 1100-1199，AES：10-19 或 2100-2199) |
| | dataIn[in] | 待解密数据 |
| | encryptionMode[in] | 加密模式，参考附录：Aidl 常量.加密模式常量 |
| | iv[in] | 初始向量 ，加密模式为 ECB 传空，为其他加密模式传入 8 字节向量 |
| | dataOut[out] | 解密结果 |
| 返回值 | 0-成功<br><0-失败 | |
| 备注 | None | |
| 示例 | 1.当 **encryptionMode** 为 DATA_MODE_OFB、DATA_MODE_CFB 时，dataIn 的长度可以为非 8、16 的整数倍，否则必须为 8、16 的整数倍<br>2.dataOut 的长度应和 dataIn 的长度相等<br>例如：byte[] dataIn = new byte[16];<br>那么 dataOut 的长度也应该为 16 字节<br>3. keyIndex 范围为 1100-1199(Dukpt 扩展密钥)时，dataIn 长度<=1024 | |

### 3.6.28 Dukpt 密钥计算 mac（扩展）

| | | |
|---|---|---|
| 原型 | int calcMacDukptEx(int keySelect, int keyIndex, int macType, byte[] dataIn, byte[] dataOut) | |
| 功能 | dukpt 密钥计算 mac | |
| 参数 | keySelect[in] | Dukpt 密钥选择，参见 Aidl 常量.Dukpt 密钥选择 |
| | keyIndex [in] | 密钥索引(范围为 3DES：0-9 或 1100-1199，AES：10-19 或 2100-2199) |
| | macType [in] | mac 算法 |
| | dataIn [in] | 待计算的 mac 数据 |
| | dataOut[out] | mac 结果 |
| 返回值 | 0-成功<br><0-失败 | |
| 备注 | 1. keyIndex 范围为 1100-1199(Dukpt 扩展密钥)时，dataIn 长度<=1016 | |

### 3.6.29 Dukpt 密钥校验 mac（扩展）

| | | |
|---|---|---|
| 原型 | int verifyMacDukptEx(int keySelect, int keyIndex, int macType, byte[] dataIn, byte[] macData) | |
| 功能 | dukpt 密钥校验 mac | |
| 参数 | keySelect[in] | Dukpt 密钥选择，参见 Aidl 常量.Dukpt 密钥选择 |
| | keyIndex [in] | 密钥索引(范围为 3DES：0-9 或 1100-1199，AES：10-19 或 2100-2199) |
| | macType [in] | mac 算法 |

| | | |
|---|---|---|
| | dataIn [in] | 原始数据（计算 Mac 时的输入数据） |
| | macData[in] | Mac 数据 |
| 返回值 | 0-成功<br>非 0-失败 | |
| 备注 | 1. keyIndex 范围为 1100-1199(Dukpt 扩展密钥)时，dataIn 长度<=1016 | |

### 3.6.30 保存 TR31 密钥

| | | |
|---|---|---|
| 原型 | int saveTR31Key(byte[] keyValue, int kbpkIndex, int keyIndex) | |
| 功能 | 保存 TR31 密钥 | |
| 参数 | keyValue[in] | 密钥数据 |
| | kbpkIndex[in] | KBPK 索引，范围：0~199 |
| | keyIndex[in] | 密钥索引，范围：0~199 |
| 返回值 | 0-成功<br><0-错误码 | |
| 备注 | 保存 TR31 密钥前必须先保存 KBPK | |

### 3.6.31 保存密文密钥-解密密钥为 RSA 密钥

| | | |
|---|---|---|
| 原型 | int saveCiphertextKeyRSA(int keyType, byte[] keyValue, byte[] checkValue, int keyAlgType, int keyIndex, int encryptIndexRSA, String transformation) | |
| 功能 | 保存密文密钥，解密密钥为 RSA 私钥 | |
| 参数 | keyType [in] | 密钥类型，参考附录：Aidl 常量类.密钥类型定义 |
| | keyValue[in] | 密钥数据 |
| | checkValue[in] | 密钥校验值 如果为 null 则不校验( 当 keyAlgType 为 KEY_ALG_TYPE_SM4 时，checkValue 的值为 16 字节） |
| | keyAlgType[in] | 密钥的算法类型，参考附录：Aidl 常量类.密钥算法类型 |
| | keyIndex [in] | 保存的索引 范围：0-19 |
| | encryptIndexRSA [in] | 解密密钥的索引，注意，这里是 RSA 私钥的索引 |
| | transformation [in] | 算法/工作模式/填充模式, 参考附录：Aidl 常量类.RSA transformation |
| 返回值 | 0-成功<br><0-错误码 | |
| 备注 | | |

### 3.6.32 保存 RSA 密钥

| | | |
|---|---|---|
| 原型 | int saveRSAKey(int keyType, byte[] keyValue, int keyIndex) | |
| 功能 | 保存 RSA 密钥 | |
| 参数 | keyType[in] | 密钥类型，0-RSA 公钥，1-RSA 私钥 |
| | keyValue[in] | 密钥数据，keyType 为 0(公钥)，为 ANS.1 X509 标准编码格式,<br>keyType 为 1(私钥)，为 ANS.1 PKCS#8 标准编码格式 |
| | keyIndex[in] | 密钥索引，范围：0-19 |

| | |
|---|---|
| 返回值 | 0-成功<br><0-错误码 |
| 备注 | |

### 3.6.33 删除密钥

| | | |
|---|---|---|
| 原型 | int deleteKey(int keySystem, int keyIndex) | |
| 功能 | 删除密钥 | |
| 参数 | keySystem [in] | 密钥体系。参考附录：Aidl 常量类.密钥体系常量 |
| | keyIndex [in] | 密钥索引。<br>keySystem 为 SEC_DUKPT ：索引范围 3DES：0-9 或 1100-1199，AES：10-19 或 2100-2199<br>keySystem 为 SEC_MKSK：索引范围 0-199<br>keySystem 为 SEC_RSA_KEY：索引范围为 0-19 |
| 返回值 | 0-成功<br><0-错误码 | |
| 备注 | | |

### 3.6.34 保存 Dukpt-AES 密钥

| | | |
|---|---|---|
| 原型 | int saveKeyDukptAES(int dukptKeyType, int keyType, byte[] keyValue, byte[] checkValue, byte[] ksn, int encryptType, int keyIndex) | |
| 功能 | 保存 Dukpt-AES 密钥 | |
| 参数 | dukptKeyType[in] | DukptAES 密钥类型，参见 DukptKeyType 常量定义 |
| | keyType[in] | 密钥类型。包含：<br>BDK-基础分散密钥:对应 Aidl 常量.密钥类型常量 KEY_TYPE_DUPKT_BDK<br>IPEK-初始 PIN 加密密钥：对应 Aidl 常量.密钥类型常量<br>KEY_TYPE_DUPKT_IPEK |
| | keyValue[in] | 密钥数据 |
| | checkValue[in] | 密钥校验值（默认 8 字节 0） |
| | ksn[in] | KSN（长度 12，前 8 字节有效） |
| | encryptType[in] | 密钥算法 |
| | keyIndex[in] | 保存的索引，范围 10-19 或 2100-2199 |
| 返回值 | 0-成功<br><0-失败 | |
| 备注 | None | |
| 示例 | None | |

### 3.6.35 计算 mac（扩展）

| | |
|---|---|
| 原型 | int calcMacEx(int keyIndex, int keyLen, int macAlgType, byte[] diversify, byte[] dataIn, byte[] dataOut) |

| | | |
|---|---|---|
| 功能 | 实现数据 Mac 计算 | |
| 参数 | keyIndex[in] | mac 密钥索引，范围：0~199 |
| | keyLen[in] | mac 密钥长度，3DES 密钥可以指定使用密钥的前 8/16/24 字节计算 mac，<br>0-整个密钥，N-密钥的前 N 个字节计算 mac |
| | macAlgType[in] | mac 算法类型，参考附录：Aidl 常量.MAC 算法类型 |
| | diversify[in] | 分散因子，计算 mac 时 mac 密钥和分散因子运算产生新的 mac 密钥，并使用新产生的 mac 密钥计算 mac。diversify 长度 8/16 字节 |
| | dataIn[in] | 用于进行 mac 计算的源数据 |
| | dataOut[out] | 计算生成的 MAC 值 |
| 返回值 | 0-成功<br><0-错误码 | |
| 备注 | mac 长度根据 mac 密钥类型不同而不同，3DES 密钥返回 8 字节 mac，SM4 密钥返回 16 字节 mac | |

### 3.6.36 生成 SM2 公私钥对

| | | |
|---|---|---|
| 原型 | int generateSM2Keypair(int pvkIndex, Bundle pubKey) | |
| 功能 | 生成 SM2 公私钥对 | |
| 参数 | pvkIndex[in] | 私钥索引，范围 0-9 |
| | pubKey[out] | 包含以下 key：<br>data：byte[]（长度 64B），公钥数据<br>kcv：byte[]（长度 5B），公钥 kcv<br>rfu: byte[]（长度 10B），RFU 数据 |
| 返回值 | 0-成功<br><0-错误码 | |
| 备注 | | |

### 3.6.37 注入 SM2 密钥

| | | |
|---|---|---|
| 原型 | int injectSM2Key(int keyIndex, Bundle keyData) | |
| 功能 | 注入 SM2 密钥 | |
| 参数 | pvkIndex[in] | 密钥索引，范围 0-9 |
| | keyData[in] | 密钥数据，包含以下 key：<br>data：byte[]（公钥长度 64B，私钥长度 32B），公钥/私钥数据<br>kcv：byte[]（长度 5B，可选），公钥/私钥 kcv<br>rfu: byte[]（长度 10B，可选），公钥/私钥 RFU 数据 |
| 返回值 | 0-成功<br><0-错误码 | |
| 备注 | | |

### 3.6.38 SM2 签名

| | |
|---|---|
| 原型 | int sm2Sign(int pukIndex, int pvkIndex, byte[] userId, byte[] dataIn, byte[] dataOut) |

| | | |
|---|---|---|
| 功能 | SM2 算法签名数据 | |
| 参数 | pukIndex[in] | 公钥索引，范围 0-9 |
| | pvkIndex[in] | 私钥索引，范围 0-9 |
| | userId[in] | 签名者 ID，小于 512 字节，国密推荐默认值为 0x31,0x32,0x33,0x34,0x35,0x36,0x37,0x38,0x31,0x32,0x33,0x34,0x35,0x36,0x37,0x38 |
| | dataIn[in] | 待签名的数据，长度小于 2048 字节 |
| | dataOut[out] | Buffer，存放签名数据（定长 64B） |
| 返回值 | >=0-dataOut 中有效数据的长度<br><0-错误码 | |
| 备注 | | |

### 3.6.39 SM2 验签

| | | |
|---|---|---|
| 原型 | int sm2VerifySign(int pukIndex, byte[] userId, byte[] dataIn, byte[] signData) | |
| 功能 | SM2 算法验签 | |
| 参数 | pukIndex[in] | 公钥索引，范围 0-9 |
| | userId[in] | 签名者 ID，小于 512 字节 |
| | dataIn[in] | 待验签的数据，长度小于 2048 字节 |
| | signData[in] | 签名数据，定长 64B |
| 返回值 | 0-成功<br><0-错误码 | |
| 备注 | | |

### 3.6.40 SM2 加密

| | | |
|---|---|---|
| 原型 | int sm2EncryptData(int pukIndex, byte[] dataIn, byte[] dataOut) | |
| 功能 | SM2 算法加密数据 | |
| 参数 | pukIndex[in] | 公钥索引，范围 0-9 |
| | dataIn[in] | 待加密的数据，长度小于 896 字节 |
| | dataOut[out] | Buffer，存放加密后的数据 |
| 返回值 | >=0-dataOut 中有效数据的长度<br><0-错误码 | |
| 备注 | | |

### 3.6.41 SM2 解密

| | | |
|---|---|---|
| 原型 | int sm2DecryptData(int pvkIndex, byte[] dataIn, byte[] dataOut) | |
| 功能 | SM2 算法解密数据 | |
| | pvkIndex[in] | 私钥索引，范围 0-9 |
| 参数 | dataIn[in] | 待解密的数据，长度小于 896 字节 |
| | dataOut[out] | Buffer，存放解密后的数据 |

| | |
|---|---|
| 返回值 | >=0-dataOut 中有效数据的长度<br><0-错误码 |
| 备注 | |

### 3.6.42 计算 Hash

| | | |
|---|---|---|
| 原型 | int calcSecHash(int mode, byte[] dataIn, byte[] dataOut); | |
| 功能 | 计算数据摘要(Hash) | |
| 参数 | mode [in] | 摘要模式，参见 Hash 算法类型 |
| | dataIn[in] | 待计算的数据，长度小于 1920 字节 |
| | dataOut[out] | Buffer，存放计算后的 hash 数据 |
| 返回值 | >=0-dataOut 中有效数据的长度<br><0-错误码 | |
| 备注 | | |

### 3.6.43 校验 Mac

| | | |
|---|---|---|
| 原型 | int verifyMac(int keyIndex, int macAlgType, byte[] dataIn, byte[] mac) | |
| 功能 | 校验 Mac | |
| 参数 | keyIndex [in] | Mac 密钥索引，范围 0-199 |
| | macAlgType [in] | Mac 算法类型，参考附录：Aidl 常量.MAC 算法类型 |
| | dataIn [in] | 待验证数据 |
| | mac[in] | mac 数据 |
| 返回值 | 0-成功<br><0-错误码 | |
| 备注 | | |

### 3.6.44 生成 RSA 公私钥对（仅支持 1024/2048 为密钥）

| | | |
|---|---|---|
| 原型 | int generateRSAKeypair(int pvkIndex, int keySize, String pubExponent, byte[] dataOut) | |
| 功能 | 生成 RSA 公私钥对（仅支持 1024/2048 位密钥） | |
| 参数 | pvkIndex [in] | 私钥索引，范围 0-19 |
| | keySize[in] | 密钥长度，支持 1024/2048 位密钥 |
| | pubExponent[in] | 公钥指数，Hex 格式，支持 03/010001 |
| | dataOut[out] | Buffer，存放公钥模 |
| 返回值 | >=0-dataOut 中有效数据的长度<br><0-错误码 | |
| 备注 | | |

### 3.6.45 注入 RSA 密钥（仅支持 1024/2048 位密钥）

| 原型 | int injectRSAKey(int keyIndex, int keySize, String module, String exponent) | |
|---|---|---|
| 功能 | 注入 RSA 密钥（仅支持 1024/2048 位密钥） | |
| 参数 | keyIndex [in] | 私钥索引，范围 0-19 |
| | keySize[in] | 密钥长度，支持 1024/2048 位密钥 |
| | module [in] | 密钥模，Hex 格式 |
| | exponent[in] | 指数，Hex 格式，支持 03/010001 |
| 返回值 | 0-成功<br><0-错误码 | |
| 备注 | | |

### 3.6.46 RSA 公钥加密或私钥解密

| 原型 | int rsaEncryptOrDecryptData(int keyIndex, int padding, byte[] dataIn, byte[] dataOut) | |
|---|---|---|
| 功能 | RSA 公钥加密或私钥解密 | |
| 参数 | keyIndex [in] | 密钥索引，范围 0-19 |
| | padding[in] | 填充模式，0-NoPadding，1-PKCS1Padding，2-PKCS7Padding |
| | dataIn [in] | 待加密/解密数据，长度小于 896 字节 |
| | dataOut[in] | Buffer，存放加密/解密的结果数据 |
| 返回值 | >=0-dataOut 中有效数据的长度<br><0-错误码 | |
| 备注 | | |

### 3.6.47 获取密钥 checkValue（扩展）

| 原型 | int getKeyCheckValueEx( Bundle bundle, byte[] dataOut) | |
|---|---|---|
| 功能 | RSA 公钥加密或私钥解密 | |
| 参数 | bundle[in] | 入参，包含如下 key：<br>keySystem：int，密钥体系<br>keyIndex：int，密钥索引<br>kcvMode：int，kcv 模式<br>targetAppPkgName：String，目标应用包名 |
| | dataOut[out] | Buffer，存放获取到的 kcv（4B） |
| 返回值 | 0-成功<br><0-错误码 | |
| 备注 | | |

### 3.6.48 删除密钥（扩展）

| 原型 | int deleteKeyEx(Bundle bundle) |
|---|---|

| 功能 | 删除密钥 | |
|---|---|---|
| 参数 | bundle[in] | 入参，包含如下 key：<br>keySystem：int，密钥体系<br>keyIndex：int，密钥索引<br>targetAppPkgName：String，目标应用包名 |
| 返回值 | 0-成功<br><0-错误码 | |
| 备注 | | |

### 3.6.49 注入密文密钥（扩展）

| 原型 | int injectCiphertextKeyEx(Bundle bundle) | |
|---|---|---|
| 功能 | 注入密文密钥 | |
| 参数 | bundle[in] | 入参，包含如下 key：<br>targetAppPkgName：String，目标应用包名<br>keyType：int，密钥类型：KEK/TMK/PIK/TDK/MAK/REC<br>keyValue：byte[]，密钥数据<br>checkValue：byte[]，密钥校验值<br>encryptIndex：int，解密密钥的索引<br>keyAlgType：int，密钥算法类型，1-3Des，2-AES，3-SM4<br>keyIndex：int，密钥保存的位置索引，0-199<br>keyLength：int，密钥长度（明文） |
| 返回值 | 0-成功<br><0-错误码 | |
| 备注 | | |

### 3.6.50 注入 dukpt 密钥（扩展）

| 原型 | int injectKeyDukptEx(Bundle bundle) | |
|---|---|---|
| 功能 | 注入 dukpt 密钥 | |
| 参数 | bundle[in] | 入参，包含如下 key：<br>targetAppPkgName：String，目标应用包名<br>keyValue：byte[]，密钥数据<br>checkValue：byte[]，密钥校验值<br>ksn：byte[]，密钥序列号<br>encryptIndex：int，解密密钥的索引<br>keyAlgType：int，密钥算法类型，1-3Des，2-AES<br>keyIndex：int，密钥保存的位置索引，3DES：0-9 或 1100-1199，AES：10-19 或 2100-2199<br>isEncrypt：bool，是否密文<br>keyLength：int，密钥长度（明文） |
| 返回值 | 0-成功<br><0-错误码 | |

| | |
|---|---|
| 备注 | |

# 3.7 EMV 操作模块

## 3.7.1 EMV 接口

### 3.7.1.1 添加 AID 参数

| | | |
|---|---|---|
| 原型 | int addAid(AidV2 aid) | |
| 功能 | 添加或更新一条 aid | |
| 参数 | aid[in] | 单条 AID，参照 AidV2 |
| 返回值 | 0-添加或修改成功<br>非 0-错误码 | |
| 备注 | None | |

### 3.7.1.2 删除 AID 参数

| | | |
|---|---|---|
| 原型 | int deleteAid(String tag9F06Value) | |
| 功能 | 根据 9f06 tag 的 value 值删除一条 aid ,如果需要删除所有 aid ,则参数 tag9F06Value 传 null | |
| 参数 | tag9F06Value[in] | 9f06 tag 的 value 值（16 进制字符串） |
| 返回值 | 0-删除成功<br>非 0-错误码 | |
| 备注 | None | |

### 3.7.1.3 添加 CAPK 参数

| | | |
|---|---|---|
| 原型 | int addCapk(CapkV2 capk) | |
| 功能 | 添加或更新一条 capk | |
| 参数 | capk[in] | 单条 CAPK，参见 CapkV2 |
| 返回值 | 0-添加或修改成功<br>非 0-错误码 | |
| 备注 | None | |

### 3.7.1.4 删除 CAPK 参数

| | | |
|---|---|---|
| 原型 | int deleteCapk(String tag9F06Value,String tag9F22Value) | |
| 功能 | 根据 9f06 tag 的 value 值和 9f22 tag 的 value 值删除一条 capk，若需要删除所有 capk，则参数 tag9F06Value 传 null | |
| 参数 | tag1Value[in] | 9f06 tag 的 value 值（16 进制字符串） |

| | | |
|---|---|---|
| | tag2Value[in] | 9f22 tag 的 value 值（16 进制字符串） |
| 返回值 | 0-删除成功<br>非 0-错误码 | |
| 备注 | None | |

### 3.7.1.5 修改终端参数

| | | |
|---|---|---|
| 原型 | int setTerminalParam(EmvTermParamV2 termParam) | |
| 功能 | 修改终端参数 | |
| 参数 | termParam [in] | 详见 EmvTermParamV2 |
| 返回值 | 0-修改成功<br>非 0-错误码 | |
| 备注 | None | |

### 3.7.1.6 检查 CAPK 和 AID 是否存在

| | | |
|---|---|---|
| 原型 | int checkAidAndCapk() | |
| 功能 | 检查内核中 AID 和 CAPK 是否存在 | |
| 参数 | [in] | None |
| 返回值 | -1：aid 和 capk 都不存在<br>0：aid 和 capk 都存在<br>1：aid 存在，capk 不存在<br>2：capk 存在，aid 不存在 | |
| 备注 | None | |

### 3.7.1.7 初始化 EMV 流程

| | | |
|---|---|---|
| 原型 | int initEmvProcess () | |
| 功能 | 初始化 EMV 流程 | |
| 参数 | [in] | None |
| 返回值 | 0-成功<br>非 0-错误码 | |
| 备注 | None | |

### 3.7.1.8 交易处理

| | | |
|---|---|---|
| 原型 | void transactProcess(EMVTransDataV2 transData, EMVListenerV2 listener) | |
| 功能 | 开始 EMV 流程处理处理 | |
| 参数 | transData [in] | 详见 EMVTransDataV2 |
| | listener [in] | Listener EMV 回调接口，详见 EMVListnerV2 |

| | |
|---|---|
| 返回值 | None |
| 备注 | None |

### 3.7.1.9 读取单条内核数据

| | | |
|---|---|---|
| 原型 | int getTlv(int opCode,String tag, byte[] outData) | |
| 功能 | 读取内核数据 | |
| 参数 | opcode[in] | 操作类型，0-普通 TLV，1-PayPass TLV，2-PayWave TLV，3-MIR TLV，4-PAGO TLV，5-JCB TLV，6-PURE TLV，7-AE TLV |
| | tag[in] | 需要取出的 TAG,例:“95” |
| | outData[out] | 取出的数据（TLV 格式） |
| 返回值 | >0-outData 中有效数据的长度<br>=0-没有读取到对应的 TAG<br><0-错误码 | |
| 备注 | None | |

### 3.7.1.10 读取内核数据列表

| | | |
|---|---|---|
| 原型 | int getTlvList(int opCode,String[] tags, byte[] outData) | |
| 功能 | 读取内核数据 | |
| 参数 | opCode[in] | 操作类型，0-普通 TLV，1-PayPass TLV，2-PayWave TLV，3-MIR TLV，4-PAGO TLV，5-JCB TLV，6-PURE TLV，7-AE TLV |
| | tags[in] | 需要取出的 TAG 列表,例 :{“95”，“9F2A”}如果有多个 TAG 需要读取，则按数组的方式传入多个 TAG |
| | outData[out] | 输出取出的数据（TLV 格式） |
| 返回值 | >0-outData 中有效数据的长度<br>=0-没有读取到对应的 TAG 列表值<br><0-错误码 | |
| 备注 | 如果内核 tag 没有对应值，那么 outData 不包含该条数据，比如传进去{“95”，“9F2A”},95 内核没有读出值，那么返回 TLV 数据为 9F2A03A00001 | |

### 3.7.1.11 设置 TLV 数据

| | | |
|---|---|---|
| 原型 | void setTlv(int opCode,String tag, String hexValue) | |
| 功能 | 设置 tlv 数据，包含授权数据，发卡行脚本处理等 | |
| 参数 | opCode [in] | 操作类型，参见 TLV 操作类型定义 |
| | tag [in] | tag，16 进制字符串 |
| | hexValue [in] | 值，16 进制字符串 |
| 返回值 | None | |
| 备注 | None | |

### 3.7.1.12 设置 TLV 数据列表

| 原型 | void setTlvList(int opCode,String[] tags, String[] hexValues) | |
|---|---|---|
| 功能 | 设置 tlv 数据，包含授权数据，发卡行脚本处理等 | |
| 参数 | opCode [in] | 操作类型，参见 TLV 操作类型定义 |
| | tags [in] | 需要取出的 TAG 列表,例：{“9F1A”，“9F33”} |
| | hexValues [in] | 需要设置的值列表，和 tags 要一一对应，如{“0156”，“E0F8C8”}则表示“9F1A”、“9F33”的值分别为“0156”、“E0F8C8” |
| 返回值 | None | |
| 备注 | None | |

### 3.7.1.13 应用选择结果

| 原型 | void importAppSelect (int selectIndex) | |
|---|---|---|
| 功能 | 向 EMV 模块导入应用选择结果，以便继续 EMV 流程 | |
| 参数 | selectIndex[in] | selectIndex 应用选择下标，从 0 开始 |
| 返回值 | None | |
| 备注 | None | |

### 3.7.1.14 应用最终选择结果

| 原型 | void importAppFinalSelectStatus(int status) | |
|---|---|---|
| 功能 | 向 EMV 流程导入应用最终选择结果，以便继续 EMV 流程 | |
| 参数 | status[in] | 卡号结果：0-成功，1-失败 |
| 返回值 | None | |
| 备注 | None | |

### 3.7.1.15 卡号确认结果

| 原型 | void importCardNoStatus(int status) | |
|---|---|---|
| 功能 | 向 EMV 流程导入卡号确认结果，以便继续 EMV 流程 | |
| 参数 | status[in] | 卡号结果：0-成功，1-失败 |
| 返回值 | None | |
| 备注 | None | |

### 3.7.1.16 身份认证结果

| 原型 | void importCertStatus(int status) | |
|---|---|---|
| 功能 | 向 EMV 流程导入身份认证结果，以便继续 EMV 流程 | |
| 参数 | status[in] | status 身份认证结果：0-成功，1-失败 |

| | |
|---|---|
| 返回值 | None |
| 备注 | None |

### 3.7.1.17 密码输入结果

| | | |
|---|---|---|
| 原型 | void importPinInputStatus(int pinType, int inputResult) | |
| 功能 | 向 EMV 模块导入 PIN 输入结果,以便继续 EMV 流程 | |
| 参数 | pinTyp[in] | PIN 类型，0-联机 PIN，1-脱机 PIN |
| | inputResult[in] | PIN 输入结果，0-处理成功，1-PIN 取消，2-PIN 跳过，3-PINPAD 故障 |
| 返回值 | None | |
| 备注 | None | |

### 3.7.1.18 联机和二次授权结果

| | | |
|---|---|---|
| 原型 | Int importOnlineProcStatus(int status, String[] tags, String[] hexValues, byte[] outData) | |
| 功能 | 向 EMV 流程导入联机和二次授权结果，以便继续 EMV 流程 | |
| 参数 | status[in] | 联机和二次授权结果: 0-联机批准，1-联机拒绝，2-联机失败 |
| | tags[in] | 联机数据 tag 列表，如{ "71", "72", "91", "8A", "89" } |
| | hexValues[in] | 各个 tag 对应的数据 |
| | outData[out] | 脚本执行结果 |
| 返回值 | >=0-outData 中有效数据的长度<br><0-错误码 | |
| 备注 | None | |

### 3.7.1.19 签名结果

| | | |
|---|---|---|
| 原型 | void importSignatureStatus(int status) | |
| 功能 | 向 EMV 流程导入签名结果，以便继续 EMV 流程 | |
| 参数 | status[in] | 签名结果: 0-成功，1-失败 |
| 返回值 | None | |
| 备注 | None | |

### 3.7.1.20 读取交易日志

| | | |
|---|---|---|
| 原型 | int readTransLog(int logType, List<String> infoOut) | |
| 功能 | 读取卡片交易日志 | |
| 参数 | logType [in] | 日志类型 0:交易日志 1:圈存日志 |
| | infoOut[out] | 读取到的日志列表 |
| 返回值 | 0-成功<br>非 0-错误码 | |
| 备注 | None | |

### 3.7.1.21 中止交易处理流程

| 原型 | void abortTransactProcess() | |
|---|---|---|
| 功能 | 中止交易处理流程 | |
| 参数 | [in] | None |
| 返回值 | None | |
| 备注 | **注意：若 EMV 流程未启动，调用本方法无任何效果；若 EMV 流程已启动，调用本方法会中断当前 EMV 流程，并回调 EMVListenerV2. onTransResult()** | |

### 3.7.1.22 导入数据交换结果

| 原型 | void importDataExchangeStatus(int status) | |
|---|---|---|
| 功能 | 向 EMV 流程导入数据交换结果，以便继续 EMV 流程 | |
| 参数 | status[in] | 数据交换结果：0-成功，1-失败 |
| 返回值 | None | |
| 备注 | None | |

### 3.7.1.23 交易处理（扩展）

| 原型 | void transactProcessEx(Bundle transData, EMVListenerV2 listener) | |
|---|---|---|
| 功能 | 开始 EMV 流程，参见 transactProcess() | |
| 参数 | transData[in] | 交易参数，参见 EMVTransDataV2，可按如下格式传值:<br>Bundle bundle = new Bundle();<br>bundle.putString("amount", amount); //交易金额<br>bundle.putString("transType", transType);//交易类型<br>bundle.putInt("flowType", flowType);//流程类型，参见 Aidl 常量. EMV FlowType 定义<br>bundle.putInt("cardType", cardType);//卡类型，2-IC，4-NFC<br>bundle.putString("cashbackAmount", cashbackAmount); //返现值<br>bundle.putInt("emvAuthLevel", level);//emv 认证级别，0-普通，1-电子现金 |
| | listener[in] | EMV 回调接口，详见 EMVListnerV2 |
| 返回值 | None | |
| 备注 | None | |

### 3.7.1.24 查询电子现金余额

| 原型 | int queryECBalance(Bundle bundle) | |
|---|---|---|
| 功能 | 查询电子现金余额 | |
| 参数 | bundle[out] | 包含如下数据：<br>9F51：String，应用货币代码（Hex 格式）<br>9F79：long，电子现金余额 |

| | |
|---|---|
| 返回值 | None |
| 备注 | None |

### 3.7.1.25 添加或修改一条 DRL LimitSet

| | | |
|---|---|---|
| 原型 | int addDrlLimitSet(DrlV2 drl) | |
| 功能 | 添加或修改一条 DRL LimitSet | |
| 参数 | drl[in] | DRL LimitSet 实体类，参见 DrlV2 |
| 返回值 | 0-成功<br>非 0-错误码 | |
| 备注 | None | |

### 3.7.1.26 删除 DRL LimitSet

| | | |
|---|---|---|
| 原型 | int deleteDrlLimitSet(String programId) | |
| 功能 | 根据 programId 的值删除一条 DRL LimitSet，若要删除所有 DRL LimitSet，则 programId 传 null | |
| 参数 | programId[in] | 应用程序 ID |
| 返回值 | 0-成功<br>非 0-错误码 | |
| 备注 | None | |

### 3.7.1.27 设置终端参数（扩展）

| | | |
|---|---|---|
| 原型 | void setTermParamEx(Bundle bundle) | |
| 功能 | 设置终端参数 | |
| 参数 | bundle[in] | 包含如下数据：<br>supportDRL：boolean，是否支持 DRL 功能<br>downloadAidParam：boolean，是否下发 AID 参数<br>downloadAidParamAll：boolean，是否下发 AID 参数 ALL<br>downloadPreParamEP：boolean，是否下发 EntryPoint 预处理参数<br>optOnlineRes：boolean，是否优化联机结果<br>ledLightingDuration：int，led 点亮时长<br>contactlessManualSelApp：boolean，非接手动选择应用<br>contactlessManualSelAppGeneral：boolean，非接手动选择应用（通用版本）<br>supportEP：boolean，是否支持 EP<br>importScriptData：boolean，联机拒绝导入脚本数据 |
| 返回值 | None | |
| 备注 | None | |

### 3.7.1.28 查询 Aid 或 Capk 列表

| 原型 | int queryAidCapkList(int type, List<String> list) | |
|---|---|---|
| 功能 | 查询 Aid 或 Capk 列表 | |
| 参数 | type[in] | 查询类型：0-查询 Aid 列表，1-查询 Capk 列表 |
| | list[out] | 查询到的 Aid/Capk 列表(Hex 格式) |
| 返回值 | 0-成功<br>非 0-错误码 | |
| 备注 | None | |

### 3.7.1.29 交易预处理

| 原型 | int transactPreProcess() | |
|---|---|---|
| 功能 | EMV 交易预处理 | |
| 参数 | [in] | None |
| | [out] | None |
| 返回值 | 0-成功<br>非 0-错误码 | |
| 备注 | None | |

### 3.7.1.30 添加 RevocationList

| 原型 | int addRevocList(RevocListV2 revocList) | |
|---|---|---|
| 功能 | 添加或更新一条 RevocationList | |
| 参数 | revocList[in] | RevocationList，参见 Aidl 常量. RevocListV2 |
| | [out] | None |
| 返回值 | 0-成功<br>非 0-错误码 | |
| 备注 | None | |

### 3.7.1.31 删除 RevocationList

| 原型 | int deleteRevocList(RevocListV2 revocList) | |
|---|---|---|
| 功能 | 删除一条 RevocationList 或清空 RevocationList | |
| 参数 | revocList[in] | RevocationList，参见 Aidl 常量. RevocListV2。若传入的值为 null，则清空所有 RevocationList |
| | [out] | None |
| 返回值 | 0-成功<br>非 0-错误码 | |
| 备注 | None | |

### 3.7.1.32 设置系统时间

| 原型 | int sysSetTime(long timeStamp) | |
|---|---|---|
| 功能 | 设置系统时间 | |
| 参数 | timeStamp[in] | 毫秒级时间戳，单位：ms |
| | [out] | None |
| 返回值 | 0-成功<br>非 0-错误码 | |
| 备注 | None | |

### 3.7.1.33 获取系统时间

| 原型 | int sysGetTime(byte[] outData) | |
|---|---|---|
| 功能 | 获取当前的系统时间，单位：s | |
| 参数 | [in] | None |
| | outData[out] | 当前的系统时间，yyyyMMddHHmmss 格式字符串转换成 HEX 字节数组，如：hexStringToBytes(“20191130142020”) |
| 返回值 | >=0- **outData** 中有效数据的长度<br><0-错误码 | |
| 备注 | None | |

### 3.7.1.34 清除数据

| 原型 | int clearData(int opCode) | |
|---|---|---|
| 功能 | 清除 EMV 数据 | |
| 参数 | opCode[in] | 操作类型，0-清除所有数据，1-清除终端数据，2-清除卡片数据。参见 Aidl 常量. 清除数据操作码定义 |
| | [out] | None |
| 返回值 | 0-成功<br>非 0-错误码 | |
| 备注 | None | |

### 3.7.1.35 设置账户数据安全参数

| 原型 | int setAccountDataSecParam(Bundle bundle) | |
|---|---|---|
| 功能 | 设置账户数据安全参数 | |
| 参数 | bundle[in] | 账户数据安全参数，包含如下 key：<br>encKeySystem: int，密钥体系，参见 密钥体系常量<br>encKeyIndex: int，磁道数据加密索引，一般传入 TDK 索引<br>encMode: int，加密模式 |

| | | |
|---|---|---|
| | | encIv: byte[]，初始向量，加密模式为 ECB 传空，为其他加密模式传入 8 字节向量<br>encPaddingMode: byte，磁道数据进行 DES 加密时，长度不是 8 的倍数，则在后面补齐 EncPaddingMode 至长度为 8 的倍数的数据<br>encMaskStart: int，表示账号前 EncMaskStart 位为明文，范围是 0~6<br>encMaskEnd: int，表示账号后 EncMaskEnd 位为明文，范围是 0~4<br>encMaskWord: char，为 0 或者是非数字字符，表示账号 EncMaskStart 至 encMaskWord 为掩码,默认为 '*' |
| | [out] | None |
| 返回值 | 0-成功<br>非 0-错误码 | |
| 备注 | None | |

### 3.7.1.36 获取账户安全数据(加密/掩码)

| | | |
|---|---|---|
| 原型 | int getAccountSecData(int opCode, String[] tags, Bundle bundle) | |
| 功能 | 获取账户安全数据 | |
| 参数 | opCode[in] | 操作码，参见 TLV 操作类型定义 |
| | tags[in] | 需要取出的 tag 列表(hex 格式),如{"5A"，"57"} |
| | bundle[out] | 输出的数据 包含 key:<br>tag+Enc，如{"5AEnc"，"57Enc"}: String, tag 加密数据<br>panMask: String, PAN 的掩码数据，只有 tag 5A 和 57 存在 |
| 返回值 | 0-成功<br>非 0-错误码 | |
| 备注 | None | |

### 3.7.1.37 导入终端风险管理结果

| | | |
|---|---|---|
| 原型 | void importTermRiskManagementStatus(int status) | |
| 功能 | 导入终端风险管理结果 | |
| 参数 | status[in] | 终端风险管理结果，0-成功，非 0-失败 |
| 返回值 | None | |
| 备注 | None | |

## 3.7.2 EMVListenerV2 EMV 回调接口

**注意，该接口为 AIDL 间传递的接口，传递该回调的时候，必须按照 aid 接口的实现**

### 3.7.2.1 请求应用选择

| | |
|---|---|
| 原型 | void onWaitAppSelect(List<EMVCandidateV2> candList, boolean isFirstSelect) |

| | | |
|---|---|---|
| 功能 | 如果卡片有多应用，则请求多应用选择。详见 EMVCandidateV2 | |
| 参数 | candList[in] | 应用列表 |
| 参数 | isFirstSelect[in] | 是否是第一次选择 |
| 返回值 | None | |
| 备注 | None | |

### 3.7.2.2 应用最终选择

| | | |
|---|---|---|
| 原型 | void onAppFinalSelect(String tag9F06Value) | |
| 功能 | 回调应用最终选择结果 | |
| 参数 | tag9F06Value[in] | Tag 9F06 的值 |
| 返回值 | None | |
| 备注 | None | |

### 3.7.2.3 请求确认卡号

| | | |
|---|---|---|
| 原型 | void onConfirmCardNo(String cardNO) | |
| 功能 | 请求确认卡号 | |
| 参数 | cardNO[in] | 卡号 |
| 返回值 | None | |
| 备注 | None | |

### 3.7.2.4 请求确认证件信息

| | | |
|---|---|---|
| 原型 | void onCertVerfiy(int certType, String certInfo) | |
| 功能 | 如果 CVM 要求确认证件信息，则请求确认证件信息 | |
| 参数 | certType[in] | 证件类型 |
| 参数 | certInfo[in] | 证件信息 |
| 返回值 | None | |
| 备注 | None | |

### 3.7.2.5 请求输密

| | | |
|---|---|---|
| 原型 | void onRequestShowPinPad(int pinType, int remainTimes) | |
| 功能 | 请求输 PIN | |
| 参数 | pinType[in] | 0-联机 PIN，1-脱机 PIN |
| | remainTimes[in] | 脱机 PIN 剩余尝试次数。当当前为联机 PIN 时，则值永远为-1；当首次启动输 PIN 时，值也为-1 |
| 返回值 | None | |
| 备注 | None | |

### 3.7.2.6 请求签名

| 原型 | void onRequestSignature() | |
|---|---|---|
| 功能 | 请求签名（如果需要） | |
| 参数 | [in] | None |
| 返回值 | None | |
| 备注 | None | |

### 3.7.2.7 请求联机和二次授权

| 原型 | void onOnlineProc() | |
|---|---|---|
| 功能 | 请求联机和二次授权（如果需要） | |
| 参数 | [in] | None |
| 返回值 | None | |
| 备注 | None | |

### 3.7.2.8 卡片数据交互完成

| 原型 | void onCardDataExchangeComplete() | |
|---|---|---|
| 功能 | 卡片数据交互完成 | |
| 参数 | [in] | None |
| 返回值 | None | |
| 备注 | None | |

### 3.7.2.9 交易结果

| 原型 | void onTransResult(int code, String desc) | |
|---|---|---|
| 功能 | 交易处理结果。本方法和 onConfirmationCodeVerified()方法互斥，即在一次 EMV 流程中两个方法中只有一个会被调用。 | |
| 参数 | code[in] | EMV 交易处理结果返回码：<br>0-正常结束，1-脱机批准，2-脱机拒绝<br><0-错误码 |
| 参数 | desc[in] | 返回码对应的错误信息 |
| 返回值 | None | |
| 备注 | None | |

### 3.7.2.10 确认码已验证

| 原型 | void onConfirmationCodeVerified() |
|---|---|
| 功能 | 通知客户端确认码已验证。本方法和 onTransResult()方法互斥，即在一次 EMV 流程中两个方法中 |

| | | |
|---|---|---|
| | 只有一个会被调用。**当前只有 PayPass NFC 可能会回调本方法。** | |
| 参数 | [in] | None |
| 返回值 | None | |
| 备注 | None | |

3.7.2.11 请求数据交互

| | | |
|---|---|---|
| 原型 | void onRequestDataExchange(String cardNo) | |
| 功能 | 请求数据交互(MIR emv 使用) | |
| 参数 | cardNo[in] | 卡号 |
| 返回值 | None | |
| 备注 | None | |

3.7.2.12 请求终端风险管理

| | | |
|---|---|---|
| 原型 | void onTermRiskManagement() | |
| 功能 | 请求终端风险管理 | |
| 参数 | [in] | |
| 返回值 | None | |
| 备注 | None | |

# 3.8 税控模块

3.8.1 税控数据读写

| | | |
|---|---|---|
| 原型 | int taxDataExchange(byte[] taxSend, byte[] taxRecv) | |
| 功能 | 读/写税控数据 | |
| 参数 | taxSend [in] | 读/写操作命令数据 |
| 参数 | taxRecv [out] | 读/写操作收到的应答数据 |
| 返回值 | >=0-taxRecv 中有效数据的长度<br><0-错误码 | |
| 备注 | 读操作命令包格式：包头（2B）+命令码（1B）+ 读取长度（2B，小端模式）<br>写操作命令包格式：包头（2B）+命令码（1B）+数据区长度 len（2B，小端模式）+数据区（len B）<br>读/写操作应答包格式：包头（2B）+应答码（1B）+数据区长度 len（2B，小端模式）+数据区（len B）<br>读/写操作返回数据的最大长度为 1030 字节，读/写操作命令示例（Hex 格式）：<br>读-发送>>：1B 1D 09 00 06<br>读-接收<<：1B 1D 00 00 06 04 01 00 02 EE DB<br>写-发送>>：1B 1D 08 00 0B 04 06 00 11 11 03 02 0A 15 39 18<br>写-接收<<：1B 1D 00 00 06 04 01 00 02 EE DB | |

# 3.9 打印模块（仅内部使用）

## 3.9.1 打印接口

### 3.9.1.1 打开打印机

| | |
|---|---|
| 原型 | int printOpen() |
| 功能 | 打印打开 |
| 参数 | None |
| 返回值 | 0-成功<br><0-错误码 |
| 备注 | 打印之前需打开打印机 |

### 3.9.1.2 关闭打印机

| | |
|---|---|
| 原型 | int printClose() |
| 功能 | 关闭打印机 |
| 参数 | None |
| 返回值 | 0-成功，<br><0-错误码 |
| 备注 | 打印完后关闭，释放打印资源 |

### 3.9.1.3 打印点行

| | | |
|---|---|---|
| 原型 | int printPointLine(byte[] pointRowData) | |
| 功能 | 打印点行 | |
| 参数 | pointRowData[in] | 点阵数据，长度为 48 的倍数，不足补 0 |
| 返回值 | >=0-打印缓冲剩余字节数<br><0-错误码 | |
| 备注 | 需要预先打开打印机 | |

### 3.9.1.4 打印走纸

| | | |
|---|---|---|
| 原型 | int printFeedPaper(int nPixels) | |
| 功能 | 打印走纸 | |
| 参数 | nPixels[in] | 走纸点行数 |
| 返回值 | =0-成功<br><0-错误码 | |
| 备注 | 需要预先打开打印机 | |

### 3.9.1.5 获取打印机状态

| 原型 | int getPrinterStatus() |
| --- | --- |
| 功能 | 获取打印机状态 |
| 参数 | None |
| 返回值 | >0-成功，参考附录：打印机状态常量<br><0-错误码 |
| 备注 | 需要预先打开打印机 |

### 3.9.1.6 获取打印机驱动版本号

| 原型 | String getPrinterDriverVersion() |
| --- | --- |
| 功能 | 获取打印机驱动版本号 |
| 参数 | None |
| 返回值 | 打印驱动版本号字符串，当为空时获取失败 |
| 备注 | |

### 3.9.1.7 设置打印灰度

| 原型 | int setGrayLevel(int level) | |
| --- | --- | --- |
| 功能 | 设置灰度 | |
| 参数 | level [in] | 灰度值，范围 70~130 |
| 返回值 | 0-成功<br><0-错误码 | |
| 备注 | | |

### 3.9.1.8 获取打印剩余缓冲

| 原型 | int getBufferRemainingRows() |
| --- | --- |
| 功能 | 获取打印剩余缓冲 |
| 参数 | None |
| 返回值 | >=0-值为缓冲剩余字节数<br><0-错误码 |
| 备注 | |

### 3.9.1.9 读取打印配置文件

| 原型 | String getPrinterConfig() |
| --- | --- |
| 功能 | 读取打印配置文件（打印由 SP 还是 MCU 控制） |
| 参数 | None |

| | |
|---|---|
| 返回值 | Y -SP 控制，N- MCU 控制，W-配置文件未下载，值为空-获取失败 |
| 备注 | |

#### 3.9.1.10 获取打印机灰度

| | |
|---|---|
| 原型 | int getPrintGrayLevel() |
| 功能 | 获取打印机灰度，灰度值范围 70~130 |
| 参数 | None |
| 返回值 | >=0-灰度值<br><0-错误码 |
| 备注 | |

#### 3.9.1.11 获取累计打印距离

| | |
|---|---|
| 原型 | int getTotalPrintDistance() |
| 功能 | 获取累计打印距离，单位：mm |
| 参数 | |
| 返回值 | >=0-累计打印的距离<br><0-错误码 |
| 备注 | |

#### 3.9.1.12 获取打印机序列号

| | |
|---|---|
| 原型 | String getPrinterSN() |
| 功能 | 获取打印机序列号 |
| 参数 | None |
| 返回值 | 打印机序列号 |
| 备注 | |

#### 3.9.1.13 注册打印状态监听对象

| | | |
|---|---|---|
| 原型 | void registerPrintCallback(PrinterCallbackV2 callback) | |
| 功能 | 注册打印状态监听对象 | |
| 参数 | callback [in] | 打印状态回调对象 |
| 返回值 | | |
| 备注 | | |

#### 3.9.1.14 取消注册打印状态回调对象

| | |
|---|---|
| 原型 | void unregisterPrintCallback() |

| | |
|---|---|
| 功能 | 取消注册打印状态回调对象 |
| 参数 | None |
| 返回值 | |
| 备注 | |

#### 3.9.1.15 设置打印速度

| | | |
|---|---|---|
| 原型 | int setPrintSpeed(int speed) | |
| 功能 | 设置打印速度 | |
| 参数 | speed [in] | 打印速度，范围 313~4291，默认 800 |
| 返回值 | 0-成功<br><0-错误码 | |
| 备注 | | |

#### 3.9.1.16 设置打印加热点

| | | |
|---|---|---|
| 原型 | int setPrintHeatPoint(int pointNum) | |
| 功能 | 设置打印加热点 | |
| 参数 | pointNum[in] | 打印加热点，值为 64/96/128/192，默认 128 |
| 返回值 | 0-成功<br><0-错误码 | |
| 备注 | | |

### 3.9.2 PrinterCallbackV2 打印状态回调

#### 3.9.2.1 打印机状态更新

| | | |
|---|---|---|
| 原型 | void onPrinterStatusUpdate(int status) | |
| 功能 | 打印机状态更新 | |
| 参数 | status [in] | 打印机状态，1-待命，2-打印中，3-缺纸，4-过热，5-电池电压低 |
| 返回值 | | |
| 备注 | | |

# 3.10 ETC 模块

## 3.10.1 ETC 接口

### 3.10.1.1 I2C 数据交互

| | | |
|---|---|---|
| 原型 | int i2cDataExchange (int addr, byte[] send, int expOutLen, int timeout, byte[] recv) | |
| 功能 | I2C 数据交互 | |
| 参数 | addr[in] | I2C 地址 |
| | send[in] | 发送的帧数据 |
| | expOutLen[in] | 期望的输出数据的长度 |
| | timeout[in] | 超时时间 |
| | recv[out] | 出参 buffer，存放接收的数据 |
| 返回值 | >=0 - recv 中有效数据的长度<br><0 - 错误码 | |
| 备注 | I2C 数据交互的各个命令格式参照文档：《ETC 模块通讯接口协议框架(讨论稿 2).docx》<br>  <br>ETC模块通讯接口协议框架(讨论稿2). | |

### 3.10.1.2 搜索 ETC 设备

| | | |
|---|---|---|
| 原型 | void search (int maxNum, ETCSearchListenerV2 listener, int timeout) | |
| 功能 | 搜索 ETC 设备 | |
| 参数 | maxNum[in] | 最大 ETC 设备数量 |
| | listener[in] | 搜索回调，参见 ETCSearchListenerV2 |
| | timeout[in] | 超时时间 |
| 返回值 | | |
| 备注 | | |

### 3.10.1.3 取消搜索 ETC 设备

| | |
|---|---|
| 原型 | void cancelSearch () |
| 功能 | 取消搜索 ETC 设备 |
| 参数 | None |
| 返回值 | None |
| 备注 | |

### 3.10.1.4 设置搜索 ETC 设备参数

| | | |
|---|---|---|
| 原型 | int setSearchParam (Bundle bundle) | |
| 功能 | 设置搜索 ETC 设备参数 | |
| 参数 | bundle[in] | 参数，包含以下 key：<br>channel：int，通讯信道，0-5.79G 信道，1-5.80G 信道，默认为 0<br>transPower：int，发射功率，可选值 0~3，值越大发射的功率越大（默认值为 3，**推荐设置为 3**）<br>fragTimeout：int，帧接收超时时间（无命令交互后，ETC 模块进入休眠时间），单位 s<br>buzzer：int，搜到设备后蜂鸣器是否鸣叫，0-不鸣叫，1-鸣叫（默认） |
| 返回值 | 0-成功<br><0-错误码 | |
| 备注 | | |

### 3.10.1.5 ETC 扣费-搜索 OBU

| | | |
|---|---|---|
| 原型 | void searchTradeOBU(int unixTime, String obuId, int timeout, ETCSearchTradeOBUListenerV2 listener) | |
| 功能 | ETC 扣费-搜索 OBU | |
| 参数 | unixTime[in] | Unix time，1970/01/01 00:00:00 到现在的秒数 |
| | obuId[in] | OBUId(Hex，4B)，可传 null |
| | timeout[in] | 超时时间，单位:ms |
| | listener[out] | 搜索 OBU 回调，参见 ETCSearchTradeOBUListenerV2 |
| 返回值 | 0-成功<br><0-错误码 | |
| 备注 | | |

### 3.10.1.6 ETC 扣费-获取安全信息密文

| | | |
|---|---|---|
| 原型 | int getTradeVehicleCipherInfo(int expectLen, String random, int macKeyVersion, int encryptVersion, Bundle bundle) | |
| 功能 | ETC 扣费-获取安全信息密文 | |
| 参数 | expectLen[in] | 期望获取到密文数据长度（1B） |
| | random[in] | 云端产生的随机数（Hex，8B，若无，则传 8 字节 0） |
| | macKeyVersion [in] | Mac 密钥版本（1B，默认传 0） |
| | encryptVersion[in] | 加密版本（1B，默认传 0） |
| | bundle [out] | 出参，应答数据，包含 key：<br>allRet：String，车辆信息密文(Hex) |
| 返回值 | 0-成功<br><0-错误码 | |

| | |
|---|---|
| 备注 | |

### 3.10.1.7 ETC 扣费-获取卡片消费记录

| | | |
|---|---|---|
| 原型 | int getTradeRecord(Bundle bundle) | |
| 功能 | ETC 扣费-获取卡片消费记录 | |
| 参数 | bundle[out] | 出参，卡片消费记录，包含 key：<br>cardType：int，卡片类型（1B），00-储值卡，01-记账卡，02-非法卡片<br>balance：int，卡片余额（4B），单位：分（0002 文件）<br>0019File：byte[]，0019 文件（Hex，43B，交易记录文件）<br>allRet：所有返回数据(Hex，48B) |
| 返回值 | 0-成功<br><0-错误码 | |
| 备注 | | |

### 3.10.1.8 ETC 扣费-消费初始化

| | | |
|---|---|---|
| 原型 | int initTrade(int keyIndex, int amount, String terminalNo, Bundle bundle) | |
| 功能 | ETC 扣费-消费初始化 | |
| 参数 | keyIndex[in] | 密钥索引（1B，默认传 01） |
| | amount[in] | 消费金额（4B，单位分） |
| | terminalNo[in] | 终端机编号（Hex，6B，PSAM 卡序列号） |
| | bundle[out] | 出参，消费初始化应答数据，包含 key：<br>balance：String，电子存折或电子钱包旧余额（Hex，4B）<br>offlineTradeNo：String，电子存折或电子钱包脱机交易序号（Hex，2B）<br>overdrawLimit：String，透支限额（Hex，3B）<br>keyVersion：String，密钥版本号（Hex，1B）<br>algorithmId：String，算法标志（Hex，1B）<br>pseudorandomNum：String，伪随机数（Hex，4B）<br>allRet：String，所有返回数据(Hex，15B) |
| 返回值 | 0-成功<br><0-错误码 | |
| 备注 | | |

### 3.10.1.9 ETC 扣费-复合消费

| | | |
|---|---|---|
| 原型 | int complexTrade(byte[] cacheData, String tradeNo, String tradeDate, String tradeTime, String mac, Bundle bundle) | |
| 功能 | ETC 扣费-复合消费 | |
| 参数 | cacheData[in] | 缓存数据（43B，0019 文件内容） |
| | tradeNo[in] | 终端脱机交易序列号（Hex，4B） |

| | | |
|---|---|---|
| | tradeDate[in] | 交易日期（Hex，4B，格式 yyyyMMdd） |
| | tradeTime [in | 交易时间（Hex，3B，格式 HHmmss） |
| | mac[in] | MAC1（Hex，4B） |
| | bundle[out] | 出参，复合消费应答数据，包含 key：<br>tac：String，TAC(Hex，4B)<br>mac2：String，Mac2(Hex，4B)<br>allRet：String，所有返回数据(Hex，8B) |
| 返回值 | 0-成功<br><0-错误码 | |
| 备注 | | |

#### 3.10.1.10 ETC 扣费-结束消费

| | | |
|---|---|---|
| 原型 | int finishTrade(int tradeResult) | |
| 功能 | ETC 扣费-结束消费，释放资源 | |
| 参数 | tradeResult[in] | 消费结果，0-交易正常，1-操作失败，2-联系运营商，3-无卡 |
| 返回值 | 0-成功<br><0-错误码 | |
| 备注 | | |

#### 3.10.1.11 ETC 扣费-OBU 防休眠

| | | |
|---|---|---|
| 原型 | int tradeHeartbeat() | |
| 功能 | ETC 扣费-防止 OBU 休眠（保活） | |
| 参数 | [in] | |
| 返回值 | 0-成功<br><0-错误码 | |
| 备注 | | |

### 3.10.2 ETCSearchListenerV2 回调接口

#### 3.10.2.1 搜索 ETC 设备成功

| | | |
|---|---|---|
| 原型 | void onSuccess (List<ETCInfoV2> list) | |
| 功能 | 搜索 ETC 设备成功 | |
| 参数 | list [in] | 搜索到的 ETC 设备列表，参见 ETCInfoV2 |
| 返回值 | None | |
| 备注 | None | |

### 3.10.2.2 搜索 ETC 设备出错

| | | |
|---|---|---|
| 原型 | void onError(int code) | |
| 功能 | 搜索 ETC 设备出错 | |
| 参数 | code [in] | 错误码 |
| 返回值 | None | |
| 备注 | None | |

## 3.10.3 ETCSearchTradeOBUListenerV2 回调接口

### 3.10.3.1 ETC 扣费-搜索 OBU 成功

| | | |
|---|---|---|
| 原型 | void onSuccess (Bundle bundle) | |
| 功能 | ETC 扣费-搜索 OBU 成功 | |
| 参数 | bundle[in] | 搜索到的 obu 信息，包含 key：<br>deviceNo：String，设备编号（Hex，4B）<br>deviceStatus：int，设备状态：<br>bit7：0-卡片存在，1-卡片不存在；<br>bit1：0-设备正常，1-设备失效（已拆卸）；<br>bit0：0-电量正常，1-设备低电；<br>sysInfoIssuerId：String，系统信息-发行方标志（Hex，8B）<br>sysInfoContractNo：String，系统信息-合同序列号（Hex，8B）<br>0015File：0015 文件内容（Hex，43B）<br>0015CardIssuerId：String，0015 文件-发卡方标志（Hex，8B）<br>0015CardTypeId：String，0015 文件-卡类型标志（Hex,1B）<br>0015CardVersion：String，0015 文件-卡片版本号（Hex,1B）<br>0015CardNetId：String，0015 文件-卡片网络标志（Hex，2B）<br>0015CardInternalNo：String，0015 文件-卡片内部编号（Hex，8B）<br>0015StartDate :String ,0015 文件-启用时间( Hex ,4B ,格式 YYYYMMDD）<br>0015EndDate :String ,0015 文件-到期时间( Hex ,4B ,格式 YYYYMMDD）<br>0015PlateNo：String，0015 文件-车牌号码（Hex，12B）<br>0015UserType :String ,0015 文件-用户类型（Hex，1B），00-普通车，01-绑定 OBU 普通车，其他见 GB/T20851.4<br>0015PlateColor :String，0015 文件-车牌颜色（Hex，1B），00-蓝色，01-黄色，02-黑色，03-白色，09-蓝白渐变色<br>0015VehicleType：String，0015 文件-车型（Hex，1B） |
| 返回值 | None | |
| 备注 | None | |

### 3.10.3.2 ETC 扣费-搜索 OBU 出错

| | | |
|---|---|---|
| 原型 | void onError(int code) | |
| 功能 | ETC 扣费-搜索 OBU 出错 | |
| 参数 | code [in] | 错误码 |
| 返回值 | None | |
| 备注 | None | |

# ● 4. 错误码

| 错误码 | 错误描述 |
|---|---|
| -100 | 参数个数或长度错误 |
| -101 | 不支持的命令 |
| -1000 | 参数错误 |
| -1001 | 功能不支持 |
| -1002 | 初始化失败 |
| -1003 | 系统时间年错误 |
| -1004 | 系统时间月错误 |
| -1005 | 系统时间日错误 |
| -1006 | 系统时间时错误 |
| -1007 | 系统时间分错误 |
| -1008 | 系统时间秒错误 |
| -1009 | 硬件失败 |
| -1010 | Buf 长度错误 |
| -2000 | 卡片参数错误 |
| -2001 | 无卡 |
| -2002 | 多卡 |
| -2032 | Mifare 卡片拒绝命令 |
| -2033 | Mifare 卡片应答数据字节数不是期望的数量 |
| -2034 | Mifare 卡片未认证密码 |
| -2035 | Mifare 认证失败 |
| -2036 | Mifare 卡片响应数据错误 |
| -2037 | Mifare 参数非法 |
| -2038 | Mifare Plus CMAC 计算错误 |
| -2039 | Mifare Plus CMAC 错误 |
| -2040 | Mifare Plus AES 解密失败 |
| -2041 | Mifare Plus AES 加密失败 |
| -2100 | 磁卡数据解码中 |
| -2500 | 模块检测失败 |
| -2501 | 驱动核心数据结构错误 |
| -2502 | 模块未上电 |

| | |
|---|---|
| -2503 | 载波未打开 |
| -2520 | 通信超时 |
| -2521 | 内部 FIFO 操作失败 |
| -2522 | 通信帧错误 |
| -2523 | 通信字符校验错 |
| -2524 | 通信冲突 |
| -2525 | 通信中信号不符合协议 |
| -2526 | 通信中 CRC 校验错 |
| -2527 | M1 卡密码认证错 |
| -2528 | 卡(Mifare)认证参数不正确 |
| -2529 | 卡片存在 |
| -2540 | A 卡通信应答的数据数量与期望的不符 |
| -2541 | A 卡通信应答 WUPA/REQA 命令的第一个字符非法 |
| -2542 | A 卡通信应答的卡号校验和错 |
| -2543 | A 卡通信应答的卡号的第一个字符错 |
| -2544 | A 卡通信应答的 ATS 的 TL 字节非法 |
| -2545 | A 卡通信应答的 ATS 的 T0 字节非法 |
| -2546 | A 卡通信应答的 ATS 的 TA1 字节非法 |
| -2547 | A 卡通信应答的 ATS 的 TB1 字节非法 |
| -2548 | A 卡通信应答的 ATS 的 TC1 字节非法 |
| -2550 | B 卡通信应答的数据数量与期望的不符 |
| -2551 | B 卡通信应答 WUPB/REQB 命令的第一个字符非 0x50 |
| -2552 | ATQB 中协议类型字节的第四位不为'0' |
| -2553 | B 卡通信应答 ATTRIB 命令中信道编码的与设置的不同 |
| -2554 | B 卡通信应答 HLTB 命令应答非 0x00 错误 |
| -2560 | 接收正确的情况下重传次数到限 |
| -2561 | 块类型编码错 |
| -2562 | I 块 PCB 错或后续数据长度错 |
| -2563 | PICC 使用 I 块响应链接块 |
| -2564 | 接收的 I 块序列号不正确 |
| -2565 | R 块 PCB 错或后续数据长度错 |
| -2566 | PICC 响应 NAK 块 |
| -2567 | 接收的 R 块序列号不正确 |
| -2568 | S 块 PCB 错或后续数据长度错 |
| -2569 | PICC 发送的 S 块非 S-WTX 请求 |
| -2570 | 请求的 WTX 参数错误(=0) |
| -2571 | 卡片回送数据超过 FSD |
| -2580 | 读二代证 GUID 错误 |
| -2581 | 按键取消 |
| -2582 | 刷卡或插卡取消 |
| -2800 | 校验错误 |
| -2801 | 通信超时 |
| -2802 | 模块没有上电 |
| -2803 | ATR 错误 |

| | |
|---|---|
| -2804 | 通信错误 |
| -2805 | PPS 错误 |
| -2806 | T0 参数错误 |
| -2807 | T0 响应过程字节错误 |
| -2808 | T1 参数错误 |
| -2809 | T1 校验错误 |
| -2810 | T1 块序列号错误 |
| -3000 | 安全模块参数错误 |
| -3001 | 根密钥错误 |
| -3002 | 安全系统被锁定 |
| -3003 | 安全文件读写错误 |
| -3004 | 密钥索引错误 |
| -3005 | 密钥校验错误 |
| -3006 | 没有 PIN 输入 |
| -3007 | PIN 输入取消 |
| -3008 | PIN 输入超时 |
| -3009 | PIN 输入间隔时间太短 |
| -3010 | KCV 模式错误 |
| -3011 | KCV 校验错误 |
| -3012 | KCV ODD 校验错误 |
| -3013 | 无匹配密钥 |
| -3014 | 密钥类型错误 |
| -3015 | 密钥长度错误 |
| -3016 | 密钥指数长度错误 |
| -3017 | 目的密钥索引错误 |
| -3018 | 源密钥索引错误 |
| -3019 | 源密钥类型错误 |
| -3020 | 组索引错误 |
| -3022 | 无 KCV |
| -3023 | DUKPT 溢出 |
| -3024 | DUKPT 密钥类型错误 |
| -3025 | DUKPT KSN 需要加 1 |
| -3026 | 试图在密钥的使用范围之外使用该密钥 |
| -3027 | 对密钥使用方式错误，限定只能解密的密钥用来加密数据，比如用主密钥去计算 mac |
| -3028 | 功能尚不支持 |
| -3029 | 功能密钥属性不匹配 |
| -3030 | 未认证 |
| -3031 | TR31 类型密钥下发加密密钥错误 |
| -3032 | TR31 类型密钥下发 MAC 密钥错误 |
| -3033 | CMAC 算法错误 |
| -3034 | 数据长度错误 |
| -3035 | 算法块错误 |
| -3036 | Des 算法异常 |
| -3037 | Aes 算法异常 |

| | |
|---|---|
| -3038 | Sm4 算法异常 |
| -3039 | Sm2 算法异常 |
| -3040 | Sm3 算法异常 |
| -3041 | Rsa 算法异常 |
| -3042 | hash 算法异常 |
| -3046 | pos 公私钥异常，触发丢失，kms 公钥异常等 |
| -3047 | 刷卡超时时间 |
| -3048 | 次数超出 |
| -3049 | 密码错误 |
| -3050 | 密钥未初始化 |
| -3051 | 未设置新密码 |
| -3052 | 要求敏感服务 |
| -3061 | PIN/PAN 防穷举保护 |
| -3062 | 存在相同的密钥 |
| -3081 | 扩展密钥文件读错误 |
| -3082 | 扩展密钥文件写错误 |
| -3083 | 扩展密钥读错误 |
| -3084 | 扩展密钥文件丢失 |
| -3085 | 扩展密钥文件打开失败 |
| -3086 | 扩展密钥自检失败 |
| -3087 | 不支持的扩展密钥写操作模式 |
| -3088 | 密钥未写入 |
| -3089 | 密钥访问超时（其他应用正在访问） |
| -3090 | 删除扩展密钥文件错误 |
| -3091 | 扩展密钥其它错误 |
| -4000 | 交易拒绝 |
| -4001 | 请使用其他界面 |
| -4002 | 交易终止 |
| -4005 | 最终选择数据错误 |
| -4100 | 交易终止（命令发送接收错误） |
| -4101 | 交易终止（命令接收超时） |
| -4102 | 交易终止（命令接收超时） |
| -4103 | 交易终止（状态码错误） |
| -4104 | 交易终止（卡片被锁） |
| -4105 | 交易终止（应用被锁） |
| -4106 | 交易终止（终端无应用） |
| -4107 | 交易终止（终端和卡片无共同支持的应用） |
| -4108 | 交易终止（卡片返回数据错误） |
| -4109 | 交易终止（卡片返回数据元重复） |
| -4110 | 交易终止（交易不被接收） |
| -4111 | 交易终止（卡片过期） |
| -4112 | 预处理参数列表为空 |
| -4113 | 交易终止（L1 读卡超时） |
| -4114 | 交易终止（L1 传输错误） |

| | |
|---|---|
| -4115 | 交易终止（L1 协议错误） |
| -4116 | 交易终止（L2 必备数据错误） |
| -4117 | 交易终止（L2 卡片认证失败（脱机数据认证失败）） |
| -4118 | 交易终止（L2 状态字错误） |
| -4119 | 交易终止（L2 数据解析失败） |
| -4120 | 交易终止（L2 交易金额超过非接交易限额） |
| -4121 | 交易终止（L2 卡片数据错误） |
| -4122 | 交易终止（L2 不支持磁条卡模式） |
| -4123 | 交易终止（L2 卡片无 PPSE） |
| -4124 | 交易终止（L2 PPSE 处理错误） |
| -4125 | 交易终止（L2 候选列表为空） |
| -4126 | 交易终止（L2 IDS 读错误） |
| -4127 | 交易终止（L2 IDS 写错误） |
| -4128 | 交易终止（L2 IDS 数据错误） |
| -4129 | 交易终止（L2 IDS 无匹配 AC） |
| -4130 | 交易终止（L2 终端数据错误） |
| -4131 | 交易终止（L3 超时） |
| -4132 | 交易终止（L3 取消） |
| -4133 | 交易终止（L3 交易金额不存在） |
| -4134 | 交易终止（重新出示卡片） |
| -4135 | 交易终止（使用其他卡片（有 Data Record）） |
| -4136 | 交易终止（使用其他卡片） |
| -4137 | 交易终止（GPO 响应错误） |
| -4138 | 交易终止（最终选择卡片数据错误） |
| -4139 | 交易终止（L3 无 DET 数据） |
| -4140 | 内核类型不支持 |
| -4141 | 非接限额超过 |
| -4142 | 金额为 0 |
| -4144 | 请使用其它界面（预处理失败） |
| -4500 | 无效参数 |
| -4501 | 下载公钥时校验码错误 |
| -4502 | 终端参数不存在 |
| -4503 | 终端参数数据错误 |
| -4504 | 交易日志不存在 |
| -4505 | 交易日志数据错误 |
| -4506 | EMV 数据不存在 |
| -4507 | PBOC LOG 格式不存在 |
| -4825 | MIR 二次拍卡命令 |
| -4854 | MIR 发送 COMPLETE 命令使用空数据 |
| -4855 | MIR 发送 COMPLETE 命令使用 ODOL 数据 |
| -4856 | MIR 重选组合应用后发送 COMPLETE 命令 |
| -4857 | MIR 重选组合应用后发送 READRECORD 命令 |
| -7001 | 打印错误 |
| -7002 | 电池电压低 |

| | |
|---|---|
| -7003 | 缺纸 |
| -7004 | 温度过高 |
| -7005 | 数据错误 |
| -7006 | 打印参数无效 |
| -7007 | 设备未打开或设备操作出错 |
| -7008 | 打印缓冲溢出 |
| -8001 | 写税控数据失败 |
| -8002 | 读税控数据失败 |
| -8300 | I2C 发送数据失败 |
| -8600 | I2C 接收数据超时 |
| -10100 | 串口关闭 |
| -10101 | 串口操作超时 |
| -10102 | LRC 校验错误 |
| -10103 | 失步 |
| -10104 | SP 初始化中 |
| -10105 | SP 重启中 |
| -10106 | SP 重连中 |
| -10107 | SP 忙碌中 |
| -10108 | SP 已休眠 |
| -10200 | 读取 OS 文件包错误 |
| -10201 | SP 正在升级中 |
| -10202 | 连接 SP 失败 |
| -10203 | 打开升级文件失败 |
| -10204 | 数据包超时 |
| -10205 | 数据包处理出错 |
| -10206 | 升级字符串过长 |
| -10207 | 升级失败 |
| -10208 | 未获取到本机的 sdk 版本号 |
| -10209 | 版本号与目标升级版本一致 |
| -10210 | 查询默认信息失败 |
| -10211 | 固件版本不允许降级 |
| -10212 | 升级被取消 |
| -10300 | 输入参数错误 |
| -10301 | 应答包数据区长度非法 |
| -10302 | 应答包数据解析出错 |
| -10400 | 内核已重启 |
| -11000 | BASE 错误开始位置 |
| -11001 | 操作不允许 |
| -11002 | 文件或目录不存在 |
| -11003 | 进程不存在 |
| -11004 | 系统调用被中断 |
| -11005 | I/O 错误 |
| -11006 | 设备或地址不存在 |
| -11007 | 参数列表太长 |

| | |
|---|---|
| -11008 | 可执行文件格式错误 |
| -11009 | 错误的文件编号 |
| -11010 | 子进程不存在 |
| -11011 | 重试 |
| -11012 | 内存溢出 |
| -11013 | 缺少权限 |
| -11014 | 地址错误 |
| -11015 | 需要块地址 |
| -11016 | 设备或资源忙碌 |
| -11017 | 文件已存在 |
| -11018 | 跨设备连接 |
| -11019 | 设备不存在 |
| -11020 | 不是目录 |
| -11021 | 是否是目录 |
| -11022 | 无效的参数 |
| -11023 | 文件索引表溢出 |
| -11024 | 打开文件过多 |
| -11025 | 不是打字机 |
| -11026 | 文本文件忙碌 |
| -11027 | 文件太大 |
| -11028 | 设备空间不足 |
| -11029 | 只读文件系统 |
| -11030 | 非法移动文件指针 |
| -11031 | 连接数太多 |
| -11032 | 管道已损坏 |
| -11033 | 数学参数超出函数边界 |
| -11034 | 数学结果无法展示 |
| -11107 | 通信状态未连接 |
| -11301 | ACK 响应包参数错误 |
| -11302 | SP 待发送 ACK 数据超过通信 buf 错误 |
| -11401 | 命令包数据长度溢出错误 |
| -11402 | 命令包校验异常，无 info 区 |
| -11403 | SP 接收缓冲区无存储空间，无 info 区 |
| -11404 | SP 接收数据超时，无 info 区 |
| -11406 | 命令包序号错误 |
| -11600 | 命令包参数错误 |
| -11601 | 未知命令包、不支持的命令包 |
| -11700 | 固件更新失败 |
| -11701 | 固件超过设计大小 |
| -11702 | 固件签名校验错误 |
| -11703 | 固件 boot 命名错误 |
| -11704 | 固件更新命令错误 |
| -11705 | 固件更新 FLASH 操作错误 |
| -11706 | 设备型号获取错误 |

| | |
|---|---|
| -11707 | SE 芯片型号错误 |
| -20001 | 重复调用 |
| -20002 | 固件升级中 |
| -20003 | 参数错误 |
| -20004 | 线程被异常中断 |
| -20005 | 固件升级失败 |
| -20006 | 固件校验失败 |
| -30001 | 读卡失败（未知原因导致的读卡失败，建议重新执行读卡操作） |
| -30002 | 未知的卡类型 |
| -30003 | NFC 检卡失败 |
| -30004 | IC 检卡失败 |
| -30005 | 读卡超时 |
| -30013 | 此卡为芯片卡,不可降级交易 |
| -30014 | 建立候选列表超时 |
| -30015 | 卡片交互失败 |
| -30016 | 卡片交互参数错 |
| -40002 | 密钥长度错误 |
| -40003 | checkValue 校验不通过 |
| -40004 | 密钥保存失败 |
| -40005 | 计算 MAC 失败 |
| -40006 | 加密失败 |
| -40007 | 回传数组长度错误 |
| -40008 | 不支持的 MAC 算法类型 |
| -40009 | checkValue 长度错误 |
| -40010 | 密钥索引错 |
| -40011 | 解密失败 |
| -40012 | 密钥长度错误 |
| -40013 | 获取随机数密钥错误 |
| -40014 | 指定加密索引密钥不存在 |
| -40016 | 验签失败 |
| -40017 | 获取报警信息码失败 |
| -40018 | 密钥分区已用完 |
| -40019 | 注入 BDK 错误 |
| -50002 | 交易预处理失败 |
| -50003 | 交易处理失败 |
| -50004 | 内核处理异常 |
| -50005 | PAN 格式错 |
| -50006 | PINPAD 回调为空 |
| -50007 | 内核数据为空 |
| -50008 | 键盘初始化异常,传递的键盘坐标参数为 Null |
| -50009 | EMV 流程未结束，无法进行下次操作 |
| -50010 | 交易处理失败，不支持的交易类型 |
| -50011 | 确认卡号信息失败，或者超时 |
| -50012 | 非接卡 CVM 错误 |

| | |
|---|---|
| -50013 | 数据库操作失败 |
| -50014 | 没有匹配的 CAPK |
| -50015 | 保存终端参数错 |
| -50016 | 没有匹配的 AID |
| -50017 | 检卡出错，cardinfo 为 null |
| -50018 | 函数调用顺序错误 |
| -50019 | transdata 非法 |
| -50020 | PIN 取消 |
| -50021 | PIN 出错 |
| -50022 | 应用选择索引错误 |
| -50023 | 身份认证出错 |
| -50024 | 联机处理出错 |
| -50025 | 最终选择超时 |
| -50026 | 最终选择出错 |
| -50027 | 签名出错 |
| -50028 | 未知的 CVM 类型 |
| -60001 | 输入 PIN 超时 |
| -60002 | 启动密码键盘失败 |
| -60003 | pinPadType 类型错误(当传入的键盘类型不为 1 和 2 时候返回该错误) |
| -60004 | 获取 PinBlock 失败 |
| -60005 | PIN 状态查询线程被打断 |
| -70001 | 缺少权限 com.sunmi.perm.MSR |
| -70002 | 缺少权限 com.sunmi.perm.ICC |
| -70003 | 缺少权限 com.sunmi.perm.CONTACTLESS_CARD |
| -70004 | 缺少权限 com.sunmi.perm.PINPAD |
| -70005 | 缺少权限 com.sunmi.perm.SECURITY |
| -70006 | 缺少权限 com.sunmi.perm.LED |

# ● 5. 实体类

## 5.1 EmvTermParamV2 – 终端参数实体类

**类成员变量说明：**

```
public String ifDsn = "3030303030393035";  // IFD 序列号 9F1E（hex 格式,定长 8 字节）
public String terminalType = "22";   // 终端类型 9F35（hex 格式,定长 1 字节）
public String countryCode = "0156";  // 终端国家代码 9F1A（hex 格式,定长 2 字节）
public boolean forceOnline = false;  // 商户强制联机(1 表示总是联机交易)
public boolean getDataPIN = true;    // 密码检测前是否读重试次数
public boolean surportPSESel = true; // 是否支持 PSE 选择方式
public boolean useTermAIPFlg = true;  // 是否基于卡片 AIP 进行风险管理
public boolean termAIP = true; // 终端是否强制进行风险管理
public boolean bypassAllFlg;    // 当以 byPass 模式处理某个 PIN 后，对其他 PIN 是否也以 bypass
```

```
模式处理
public boolean bypassPin = true; // 是否支持 bypass PIN
public boolean batchCapture;     // 是否批抓取数据
public boolean ectSiFlg = true;  // 电子现金终端支持指示器(EC Terminal Support Indicator)
是否存在
public boolean ectSiVal = true;  // 是否支持电子现金终端支持指示器
public boolean ectTlFlg = true;  // 电子现金终端交易限额(EC Terminal Transaction Limit)是
否存在
public String ectTlVal = "100000"; // 电子现金终端交易限额，单位分（变长，最多 6 字节）
public String capability = "E0F8C8";        // 终端性能 9F33（定长 3 字节）
public String addCapability = "0300C00000";// 终端扩展性能 9F40（定长 5 字节）
public boolean scriptMode;                  // scriptMode
public boolean adviceFlag = true;           // adviceFlag
public boolean isSupportSM = true;          // 是否支持 SM 算法
public boolean isSupportTransLog = true;   // 是否支持交易 LOG
public boolean isSupportMultiLang = true;  // 是否支持多语言
public boolean isSupportExceptFile = true; // 是否支持异常文件
public boolean isSupportAccountSelect = true;  // 是否支持账号选择
public String TTQ = "26000080";                // 终端交易属性(非接使用)（定长 4 字节）
public boolean IsReadLogInCard;     // 是否是读卡内交易记录的应用选择过程
private byte[] reserved = new byte[3];  // 保留字节值必须为 0
```

## 5.2 AidV2 -AID 实体类

```
public byte[] aid;//AID 标识（变长，最长 16 字节）
public byte[] cvmLmt = new byte[6];            //持卡人限额(定长 6 字节，大端存储)
public byte[] termClssLmt = new byte[6];        //终端非接交易限额(定长 6 字节，大端存储)
public byte[] termClssOfflineFloorLmt = new byte[6];//终端脱机非接最低限额定(定长 6 字节，
大端存储)
public byte[] termOfflineFloorLmt = new byte[6];//终端脱机最低限额(终端电子现金交易限额)(定
长 6 字节，大端存储)
public byte selFlag;  //选择标志(PART_MATCH 部分匹配 0;
public byte targetPer;//目标百分比数
public byte maxTargetPer; // 最大目标百分比数
public byte[] floorLimit; //最低限额,大端存储 9F1B（变长，最长 4 字节）
public byte randTransSel; //是否进行随机交易选择
public byte velocityCheck;//是否进行频度检测
public byte[] threshold = new byte[4];//阈值（定长 4 字节）
public byte[] TACDenial = new byte[5];//终端行为代码(拒绝)（定长 5 字节）
public byte[] TACOnline = new byte[5];//终端行为代码(联机)（定长 5 字节）
public byte[] TACDefault = new byte[5];   //终端行为代码(缺省)（定长 5 字节）
public byte[] AcquierId = new byte[6];//收单行标志 9F01（定长 6 字节）
public byte[] dDOL;   //终端缺省 DDOL（变长，最长 32 字节）
public byte[] tDOL;   //终端缺省 TDOL（变长，最长 32 字节）
public byte[] version = new byte[2];  //应用版本（定长 2 字节）
public byte rMDLen;   //风险管理数据长度
public byte[] riskManData = new byte[8];  //风险管理数据（定长 8 字节）
public byte[] merchName = new byte[128];  //商户名 9F4E（定长 128 字节）
public byte[] merchCateCode = new byte[2];//商户类别码(没要求可不设置)9F15（定长 2 字节）
public byte[] merchId = new byte[16]; //商户标志(商户号)9F16（定长 16 字节）
public byte[] termId = new byte[8];   //终端标志(POS 号)（定长 8 字节）
```

```
public byte[] referCurrCode = {0x01, 0x56};   //参考货币代码 9F3C（定长 2 字节）
public byte referCurrExp; //参考货币指数 9F3D
public byte[] referCurrCon = new byte[4]; //参考货币代码和交易代码的转换系数(交易货币对参考货币的汇率*1000)，用于境外消费，暂时不用（定长 4 字节）
public byte clsStatusCheck;//非接触状态检查
public byte zeroCheck;//零金额检查
public byte kernelType;//内核类型 DFC10A
public byte paramType;//AID 参数类型 DFC10B(0-默认,1-接触，2-非接)
public byte[] ttq = new byte[4]; //终端交易属性 9F66
public byte[] kernelID; //内核 ID DFC10C（变长，最长 8 字节）
```

## 5.3 CapkV2 -CAPK 实体类

```
public byte[] rid = new byte[5];//应用注册服务商 ID
public byte index;  //密钥索引
public byte hashInd;//HASH 算法标志
public byte arithInd;   //RSA 算法标志
public byte[] modul;   //模（变长，最长 248 字节）
public byte[] exponent; //指数（变长，最长 3 字节）
public byte[] expDate = new byte[3];//有效期(YYMMDD)（定长 3 字节）
public byte[] checkSum = new byte[20];  //密钥校验和（定长 20 字节）
```

## 5.4 EMVTransDataV2 - 交易处理实体类

```
public String amount;  //交易金额(单位分)，必须参数，不能为 null 和 "" ,当 amount="0"时表示查询余额

public String transType = "00"; //交易类型

public int flowType = 01; // 流程类型，参见 Aidl 常量. EMV FlowType 定义

public int cardType = 2; //卡类型 2:IC 4:NFC
```

## 5.5 EMVCandidateV2 – EMV 应用候选人

```
public short index;  // 索引用于与优先级列表对应
public String aid;   // 卡片 AID
public String appPreName;// 应用优先选择名称
public String appLabel;  // 应用标签
public String issDiscrData;  // tag 'BF0C'数据：1 个字节的长度字节+'BF0C'最大 222 个字节
public byte priority;// 优先级标志
public String appName;   // 本地应用名称
public byte kernelType;  // 非接应用内核类型
```

## 5.6 PinPadConfigV2 - 密码键盘配置实体类

```
privateintpinpadType;//密码键盘类型。0：预置密码键盘(由服务实现样式统一的键盘)1:调用方自己实现的密码键盘
privateintpinType=0;//pin 类型标识(0 是联机 pin，1 是脱机 pin)
privatebooleanisOrderNumKey=false;//true:顺序键盘 false:乱序键盘
privatebyte[]pan;//ascii 格式转换成的 byte 例如“123456”.getbytes("usascii")
privateintpinkeyIndex;//pik 索引(pin 密钥索引)
privateintmaxInput=6;//最大输入位数(最多允许输入 12 位)
privateintminInput=0;//最小输入位数
privateinttimeout=60000;//超时时间/毫秒
privatebooleanisSupportbypass=true;//是否支持 bypasspin
privateintpinblockFormat=0;//pinblockformat 格式，支持以下几种格式：
  SEC_PIN_BLK_ISO_FMT0(0)
  SEC_PIN_BLK_ISO_FMT1(1)
  SEC_PIN_BLK_ISO_FMT2(2)：脱机交易默认为此种格式不需要设置
  SEC_PIN_BLK_ISO_FMT3(3)
//pinblockISO-9564 标准中定义的 5 种格式，格式详细介绍:
//https://blog.csdn.net/weixin_43177881/article/details/82793956-Format_0_PIN_block_47
privateintalgorithmType=0;//加密 Pin 的算法类型 0-3DES(返回 8 字节),1-SM4（返回 16 字节）
privateintkeySystem=0;//当前 Pik 属于的密钥体系 0-SEC_MKSK,1-SEC_DUKPT。参考附录：Aidl 常量密钥体系常量
```

## 5.7 PinPadTextConfigV2- 密码键盘显示文字配置实体类

```
public String confirm;    //confirm 键文字
public String inputPin;   //输入联机 PIN 文字
public String inputOfflinePin;//输入脱机 PIN 文字
public String reinputOfflinePinFormat;//重新输入脱机 PIN（显示剩余次数）文字
```

## 5.8 DrlV2 – DRL LimitSet 实体类

```
public boolean isDefaultLmt = false;//是否是默认的 limitSet
public boolean statusCheck = false;//是否开启状态检查
public byte zeroCheck = 1;//是否开启零金额检查,0-联机，1-notAllow,2-关闭
public byte[] programID; //应用程序 ID
public byte[] cvmLmt = new byte[6];//持卡人限额(定长 6 字节，大端存储)
public byte[] termClssLmt = new byte[6];//终端非接交易限额(定长 6 字节，大端存储)
public byte[] termClssFloorLmt = new byte[6];//终端非接最低限额定(定长 6 字节，大端存储)
public byte[] termFloorLmt = new byte[6];//终端最低限额(定长 6 字节，大端存储)
public boolean cvmLmtActivate = true;//是否开启持卡人限额检查
public boolean termClssLmtActivate = false;//是否开启终端非接限额检查
public byte termClssFloorLmtActivate = 1;//是否开启终端非接最低限额定，0-关闭，1-开启且终端非接限额存在，2-开启但终端非接限额不存在}
```

## 5.9 RevocListV2– RevocationList 实体类

```
Public byte[] rid = newbyte[5];//应用注册服务商 ID（定长 5 字节）
Public byte index;//密钥索引
Public byte[] sn = new byte[3];//序列号（定长 3 字节）
public byte[] reserved = new byte[3];//保留字节值必须为 0
```

## 5.10 ETCInfoV2 实体类

```
public String deviceNo;          //设备编号
public String deviceStatus;      //设备状态
public String cardType;          //卡类型，00-储值卡，01-记账卡，02-非法卡片
public int amount;               //卡金额
public String licensePlateColor; //车牌颜色
public String licensePlateNo;    //车牌号
public int signal;               //信号强度
```

# ● 6. 访问权限

## 6.1 权限位置

在使用各接口前，应在 AndroidManifest 文件中声明合适的权限。

## 6.2 权限定义

### 6.2.1 磁条卡权限

<uses-permission android:name="com.sunmi.perm.MSR"/>

### 6.2.2 接触式 IC 卡权限

<uses-permission android:name="com.sunmi.perm.ICC"/>

### 6.2.3 非接触式 IC 卡权限

<uses-permission android:name="com.sunmi.perm.CONTACTLESS_CARD"/>

### 6.2.4 密码键盘权限

<uses-permission android:name="com.sunmi.perm.PINPAD"/>

### 6.2.5 安全模块权限

<uses-permission android:name="com.sunmi.perm.SECURITY"/>

### 6.2.6 LED 权限

<uses-permission android:name="com.sunmi.perm.LED" />

### 6.2.7 打印机权限(不支持)

<uses-permission android:name="com.sunmi.perm.PRINTER" />

### 6.2.8 串口权限(不支持)

<uses-permission android:name="com.sunmi.perm.SERIAL"/>

### 6.2.9 客显权限(不支持)

<uses-permission android:name="com.sunmi.perm.CUSTOMER_DISPLAY"/>

### 6.2.10 二代身份证模块权限(不支持)

<uses-permission android:name="com.sunmi.perm.IDCard"/>

### 6.2.11 钱箱权限(不支持)

<uses-permission android:name="com.sunmi.perm.MONEYBOX"/>

### 6.2.12 指纹模块权限(不支持)

<uses-permission android:name="com.sunmi.perm.FINGERPRINT"/>

# 7. 附录

## 7.1 Aidl 常量类（com.sunmi.pay.hardware.aidl.AidlConstants）

### 7.1.1 卡类型常量定义

```
// 磁卡
public static final int MAGNETIC= 1<<0;
// IC 卡
public static final int IC = 1<<1;
// 非接卡
public static final int NFC = 1<<2;
// Mifare 卡
public static final int MIFARE= 1<<3;
// PSAM 卡，卡座 0
public static final int PSAM0= 1<<4;
// Felica 卡
public static final int FELICA= 1<<5;
// SAM1 卡
public static final int SAM1= 1<<6;
// Mifare plus 卡
public static final int MIFARE_PLUS= 1<<7;
// Mifare desfire 卡
public static final int MIFARE_DESFIRE= 1<<8;
// AT24C01 卡
public static final int AT24C01= 1<<9;
// AT24C02 卡
public static final int AT24C02= 1<<10;
// AT24C04 卡
public static final int AT24C04= 1<<11;
// AT24C08 卡
public static final int AT24C08= 1<<12;
// AT24C16 卡
public static final int AT24C16= 1<<13;
// AT24C32 卡
public static final int AT24C32= 1<<14;
// AT24C64 卡
public static final int AT24C64= 1<<15;
// AT24C128 卡
public static final int AT24C128= 1<<16;
// AT24C256 卡
public static final int AT24C256= 1<<17;
// AT24C512 卡
public static final int AT24C512= 1<<18;
// SLE4442 卡
public static final int SLE4442= 1<<19;
// SLE4428 卡
public static final int SLE4428= 1<<20;
// AT88SC1608 卡
public static final int AT88SC1608= 1<<21;
// CTX512B 卡
```

```
public static final int CTX512B= 1<<22;
```

### 7.1.2 密钥类型常量定义

```
// 密钥类型，KEK (Key encrypt key)
public final static int KEY_TYPE_KEK = 0x01;
// 密钥类型，TMK (Terminal master key)
public final static int KEY_TYPE_TMK = 0x02;
// 密钥类型，PIK (PIN key)
public final static int KEY_TYPE_PIK = 0x03;
// 密钥类型，MAK (Mac key)
public final static int KEY_TYPE_MAK = 0x04;
// 密钥类型，TDK (Track data key)
public final static int KEY_TYPE_TDK = 0x05;
// 密钥类型，保留
public final static int KEY_TYPE_REC = 0x06;
// 密钥类型，dupkt 根密钥(Base derived key)
public static final int KEY_TYPE_DUPKT_BDK = 0x07;
// 密钥类型，初始 PIN 加密密钥(Initial PIN encryption key)
public static final int KEY_TYPE_DUPKT_IPEK = 0x08;
//密钥类型，TR31 密钥块保护密钥 KBPK(Key block protection key)
public static final int KEY_TYPE_KBPK = 0x09;
// 密钥类型，账户数据密钥 TADK
public static final int KEY_TYPE_TADK =0x0A;
```

### 7.1.3 密钥算法类型常量定义

```
// 密钥类型 3DES/DES 都是用这个
public final static int KEY_ALG_TYPE_3DES = 0x01;
// 加密类型 AES
public final static int KEY_ALG_TYPE_AES = 0x02;
// 加密类型 SM4
public final static int KEY_ALG_TYPE_SM4 = 0x03;
```

### 7.1.4 MAC 算法类型常量定义

```
// 固定用 8 字节长度的 key
public static final int MAC_ALG_ISO_9797_1_MAC_ALG1 = 1001;
//
public static final int MAC_ALG_ISO_9797_1_MAC_ALG3 = 1003;
// 与 9797-1-ALG3 相同
public static final int MAC_ALG_ISO_16609_MAC_ALG1 = 2000;
// FAST_MODE 银联标准计算 mac
public static final int MAC_ALG_FAST_MODE = 3000;
// X9_19 算法计算 mac
public static final int MAC_ALG_X9_19 = 3001;
// CBC 银联标准算法
public static final int MAC_ALG_CBC = 3002;
// 国密 SM4 计算 mac
public static final int MAC_ALG_CUP_SM4_MAC_ALG1 = 3003;
// FAST_MODE 国际标准计算 mac
```

```
public static final int MAC_ALG_FAST_MODE_INTERNATIONAL = 30000;
// CBC 国际标准计算 mac
public static final int MAC_ALG_CBC_INTERNATIONAL = 30001;
public static final int MAC_ALG_CUP_SM4_MAC_ALG2 = 3004;
```

### 7.1.5 DukptKeyType 常量定义

```
public static final int DUKPT_KEY_TYPE_2TDEA = 1;
public static final int DUKPT_KEY_TYPE_3TDEA = 2;
public static final int DUKPT_KEY_TYPE_AES128 = 3;
public static final int DUKPT_KEY_TYPE_AES192 = 4;
public static final int DUKPT_KEY_TYPE_AES256 = 5;
```

### 7.1.6 密钥体系常量

```
public static final int SEC_MKSK = 0x00;
public static final int SEC_DUKPT = 0x01;
public static final int SEC_RSA_KEY = 0x02;
public static final int SEC_SM2_KEY = 0x03;
```

### 7.1.7 数据加解密模式定义

```
// ECB 模式
public static final int DATA_MODE_ECB = 0;
// CBC 模式
public static final int DATA_MODE_CBC = 1;
// OFB 模式
public static final int DATA_MODE_OFB = 2;
// CFB 模式
public static final int DATA_MODE_CFB = 3;
```

### 7.1.8 Dukpt 密钥选择

```
// DUKPT PIN 密钥
public static final int DUKPT_KEY_SELECT_KEY_PIN = 0;
// DUKPT 请求和响应 MAC 密钥
public static final int DUKPT_KEY_SELECT_KEY_MAC_BOTH = 1;
// DUKPT 响应 MAC 密钥
public static final int DUKPT_KEY_SELECT_KEY_MAC_RSP = 2;
// DUKPT 请求和响应数据密钥
public static final int DUKPT_KEY_SELECT_KEY_DATA_BOTH = 3;
// DUKPT 响应数据密钥，既能加密，也能解密
public static final int DUKPT_KEY_SELECT_KEY_DATA_RSP = 4;
// DUKPT 计算 Mac 密钥（dukpt-aes）
public static final int DUKPT_KEY_SELECT_KEY_MAC_GEN = 5;
// DUKPT 数据加密密钥（dukpt-aes）
public static final int DUKPT_KEY_SELECT_KEY_DATA_ENC = 6;
// DUKPT 密钥加密密钥(key encryption key)（dukpt-aes）
public static final int DUKPT_KEY_SELECT_KEY_KEY_ENC_KEY = 7;
// DUKPT ipek key,aquire service use（dukpt-aes）
```

```
public static final int DUKPT_KEY_SELECT_KEY_DERIVATION = 8;
// DUKPT bdk key,aquire service use (dukpt-aes)
public static final int DUKPT_KEY_SELECT_KEY_DERIVATION_INIT = 9;
```

### 7.1.9 RSA transformation

```
public static final String RSA_TRANSFORMATION_1 = "RSA/None/NoPadding";
public static final String RSA_TRANSFORMATION_2 = "RSA/None/PKCS1Padding";
public static final String RSA_TRANSFORMATION_3 = "RSA/ECB/NoPadding";
public static final String RSA_TRANSFORMATION_4 = "RSA/ECB/PKCS1Padding";
public static final String RSA_TRANSFORMATION_5 = "RSA/ECB/OAEPWithSHA-1AndMGF1Padding";
public static final String RSA_TRANSFORMATION_6 = "RSA/ECB/OAEPWithSHA-
256AndMGF1Padding";
public static final String RSA_TRANSFORMATION_7 = "RSA/ECB/OAEPWithSHA-
512AndMGF1Padding";
```

### 7.1.10 RSA 签名算法

```
public static final String RSA_SIGN_ALG_1 = "NONEwithRSA";
public static final String RSA_SIGN_ALG_2 = "MD5withRSA";
public static final String RSA_SIGN_ALG_3 = "SHA1withRSA";
public static final String RSA_SIGN_ALG_4 = "SHA256withRSA";
public static final String RSA_SIGN_ALG_5 = "SHA512withRSA";
```

### 7.1.11 Hash 算法类型

```
public static final int HASH_SHA_TYPE_1 = 0x00;
public static final int HASH_SHA_TYPE_224 = 0x01;
public static final int HASH_SHA_TYPE_256 = 0x02;
public static final int HASH_SHA_TYPE_384 = 0x03;
public static final int HASH_SHA_TYPE_512 = 0x04;
public static final int HASH_SM3_TYPE = 0x05;
```

### 7.1.12 证件类型常量定义

```
// 身份证
public static final int IDCARD = 536911872;
// 军官证
public static final int ARMYCARD = 536911873;
// 护照
public static final int PASSPORT = 536911874;
// 入境证
public static final int ARRIVALCARD = 536911875;
// 临时身份证
public static final int TEMPIDCARD = 536911876;
// 其他证件
public static final int OTHERCARD = 536911877;
```

### 7.1.13 EMV 模块相关常量定义

```
// 强制联机
public static final int FORCE_ONLINE = 0;
// 非联机
public static final int NO_ONLINE = 1;
// CAPK AID 都不存在
public static final int EXIST_ALL_NOT = -1;
// CAPK AID 都存在
public static final int EXIST_ALL= 0;
// CAPK 不存在
public static final int EXIST_CAPK_NOT = 1;
// AID 不存在
public static final int EXIST_AID_NOT = 2;
// 交易完成
public static final int EMV_RESULT_FINISHED = 0x9000;
// 交易终止
public static final int EMV_RESULT_TERMINATION = 0x9001;
// 交易终止，PINBLOCK 获取失败
public static final int EMV_ERROR_PINBLOCK = 0x9002;
// 不支持交易
public static final int EMV_UNSUPPORTED_TRANS = 0X9003;
```

### 7.1.14 EMV FlowType 定义

```
// 标准的授权过程
public static final int TYPE_EMV_STANDARD = 0x01;
// 简易流程-读到卡号即结束
public static final int TYPE_EMV_BRIEF = 0x02;
// QPASS 流程-NFC 跳过输密
public static final int TYPE_NFC_SKIP_CVM = 0x03;
// 非接提速流程
public static final int TYPE_NFC_SPEEDUP = 0x04;
```

### 7.1.15 清除数据操作码定义

```
// 清除所有数据
public static final int OP_CLEAR_DATA_ALL=0;
// 清除终端数据
public static final int OP_CLEAR_DATA_TERMINAL=1;
/// 清除卡片数据
public static final int OP_CLEAR_DATA_CARD=2;
```

### 7.1.16 EMV TLV 操作类型定义

```
// 普通
public static final int OP_NORMAL = 0;
// PayPass
public static final int OP_PAYPASS = 1;
```

```
// PayWave
public static final int OP_PAYWAVE = 2;
// MIR
public static final int OP_MIR = 3;
// PAGO
public static final int OP_PAGO = 4;
// JCB
public static final int OP_JCB = 5;
// PURE
public static final int OP_PURE = 6;
// AE
public static final int OP_AE = 7;
// FLASH
public static final int OP_FLASH = 8;
// DPAS
public static final int OP_DPAS = 9;
// RUPAY
public static final int OP_RUPAY = 10;
// EFTPOS
public static final int OP_EFTPOS = 11;
// AID RELEVANT
public static final int OP_AID_RELEVANT = 101;
// 添加自定义 tag
public static final int OP_ADD_SELF_DEFINE_TAG = 102;
// 删除自定义 tag
public static final int OP_DEL_SELF_DEFINE_TAG = 103;
```

### 7.1.17 EMV 内核类型定义

```
// EMV(接触)
public static final int EMV = 0;
// QPBOC
public static final int QPBOC = 1;
// PAYPASS
public static final int PAYPASS = 2;
// PAYWAVE
public static final int PAYWAVE = 3;
// AE
public static final int AE = 4;
// DISCOVER
public static final int DISCOVER = 5;
// JCB
public static final int JCB = 6;
// FLASH
public static final int FLASH = 7;
// MIR
public static final int MIR = 8;
// MCCS
public static final int MCCS = 9;
// RUPAY
public static final int RUPAY = 10;
// PAGO
public static final int PAGO = 11;
// EFTPOS
public static final int EFTPOS = 12;
```

### 7.1.18 EMV 参数类型定义

```
// CONTACT/CONTACTLESS(默认)
public static final int DEFAULT = 0;
// CONTACT(接触)
public static final int CONTACT = 1;
// CONTACTLESS(非接)
public static final int CONTACTLESS = 2;
```

### 7.1.19 EMV 交易结果定义

```
// 成功(兼容)
public static final int SUCCESS = 0;
// 脱机批准
public static final int OFFLINE_APPROVAL = 1;
// 脱机拒绝
public static final int OFFLINE_DECLINE = 2;
// 预留
public static final int RESERVE = 3;
// 重新拍卡
public static final int TRY_AGAIN = 4;
// 联机批准
public static final int ONLINE_APPROVAL = 5;
// 联机拒绝
public static final int ONLINE_DECLINE = 6;
```

### 7.1.20 AID 函数行为常量定义

```
// 添加或者更新一条 AID
public static final int ACTION_AID_ADD = 0x00;
// 删除所有 AID
public static final int ACTION_AID_DEL = 0x01;
```

### 7.1.21 CAPK 函数行为常量定义

```
// 添加或者更新一条 CAPK
public static final int ACTION_CAPK_ADD = 0x00;
// 删除所有 CAPK
public static final int ACTION_CAPK_DEL = 0x01;
```

### 7.1.22 SysParam 常量定义

```
// “HardwareVer”-设备硬件版本
public static final String HARDWARE_VERSION = "HardwareVersion";
// “FirmwareVer”-设备固件版本
public static final String FIRMWARE_VERSION = "FirmwareVersion";
```

```
// “SMVersion”-国密固件版本
public static final String SM_VERSION = "SMVersion";
//ETC 固件版本
public static final String ETC_FIRM_VERSION = "ETCFirmVersion";
// “SN”-获取机器 SN 号
public static final String SN = "SN";
// “PN”-获取机器 SN1(PN)渠道自定义 SN 号
public static final String PN = "PN";
// “TUSN”-获取机器银联 TUSN 号
public static final String TUSN = "TUSN";
// “DeviceCode”-获取设备型号(如:w6900)
public static final String DEVICE_CODE = "DeviceCode";
// “DeviceModel”-获取机型（如:P1N）
public static final String DEVICE_MODEL = "DeviceModel";
// “Reserved”-预留字段（value 为 Json 格式）
public static final String RESERVED = " Reserved";
// 非接 A 卡参数
public static final String PCD_PARAM_A = "PCD_PARAM_A";
// 非接 B 卡参数
public static final String PCD_PARAM_B = "PCD_PARAM_B";
// 非接 Felica 卡参数
public static final String PCD_PARAM_C = "PCD_PARAM_C";
//是否支持 ETC,0-不支持,1-支持
public static final String SUPPORT_ETC = "SupportETC";
//Tusn 密钥 KCV
public static final String TUSN_KEY_KCV = "TusnKeyKcv";
//key same 是否开启
public static final String SEC_MODE = "SecMode";
//IC 驱动版本号
public static final String PCD_IFM_VERSION = "PCD_IFMVersion";

// EMV 版本信息
public static final String EMV_VERSION = "EMVVersion";
// Paypass 版本
public static final String PAYPASS_VERSION = "PaypassVersion";
// Paywave 版本
public static final String PAYWAVE_VERSION = "PaywaveVersion";
// QPBOC 版本
public static final String QPBOC_VERSION = "QPBOCVersion";
// Entry 版本
public static final String ENTRY_VERSION = "EntryVersion";
// Mir 版本
public static final String MIR_VERSION = "MirVersion";
// JCB 版本
public static final String JCB_VERSION = "JCBVersion";
// Pago 版本
public static final String PAGO_VERSION = "PAGOVersion";
// Pure 版本
public static final String PURE_VERSION = "PUREVersion";
// AE 版本
public static final String AE_VERSION = "AEVersion";
// FLASH 版本信息
public static final String FLASH_VERSION = "FLASHVersion";
// DPAS 版本信息
public static final String DPAS_VERSION = "DPASVersion";
// APEMV 版本信息
```

```
public static final String APEMV_VERSION = "APEMVVersion";
//EFTPOS 版本信息
public static final String EFTPOS_VERSION = "EFTPOSVersion";
// EMV kernel checksum 信息
public static final String EMV_KERNEL_CHECKSUM = "EmvKernelCheckSum";
// Pure 版本信息
public static final String PURE_VERSION_FULL = "PUREVersionFull";
// EFTPOS 版本信息
public static final String EFTPOS_VERSION_FULL = "EFTPOSVersionFull";
// APEMV 版本信息
public static final String APEMV_VERSION_FULL = "APEMVVersionFull";
```

### 7.1.23 LedLight 常量定义

```
// 红灯
public static final int RED_LIGHT = 1;
// 绿灯
public static final int GREEN_LIGHT = 2;
// 黄灯
public static final int YELLOW_LIGHT = 3;
// 蓝灯
public static final int BLUE_LIGHT = 4;
```

### 7.1.24 打印机状态常量

```
// 待命
public static final int IDLE = 1;
// 打印中
public static final int PRINTING = 2;
// 缺纸
public static final int PAPERLESS = 3;
// 过温
public static final int OVERTEMPERATURE = 4;
// 电池电压低
public static final int LOW_BATTERY_VOLTAGE = 5;
```

### 7.1.25 密码键盘模式

```
//普通
public static final String MODE_NORMAL = "Normal";
//美团
public static final String MODE_MEITUAN = "MeiTuan";
//静音
public static final String MODE_SILENT = "Silent";
```

### 7.1.26 PinBlock 格式

```
//（支持 DES、TDES、SM4）输入格式化后的 12 位 ASCII PAN
public static final int SEC_PIN_BLK_ISO_FMT0 = 0;
```

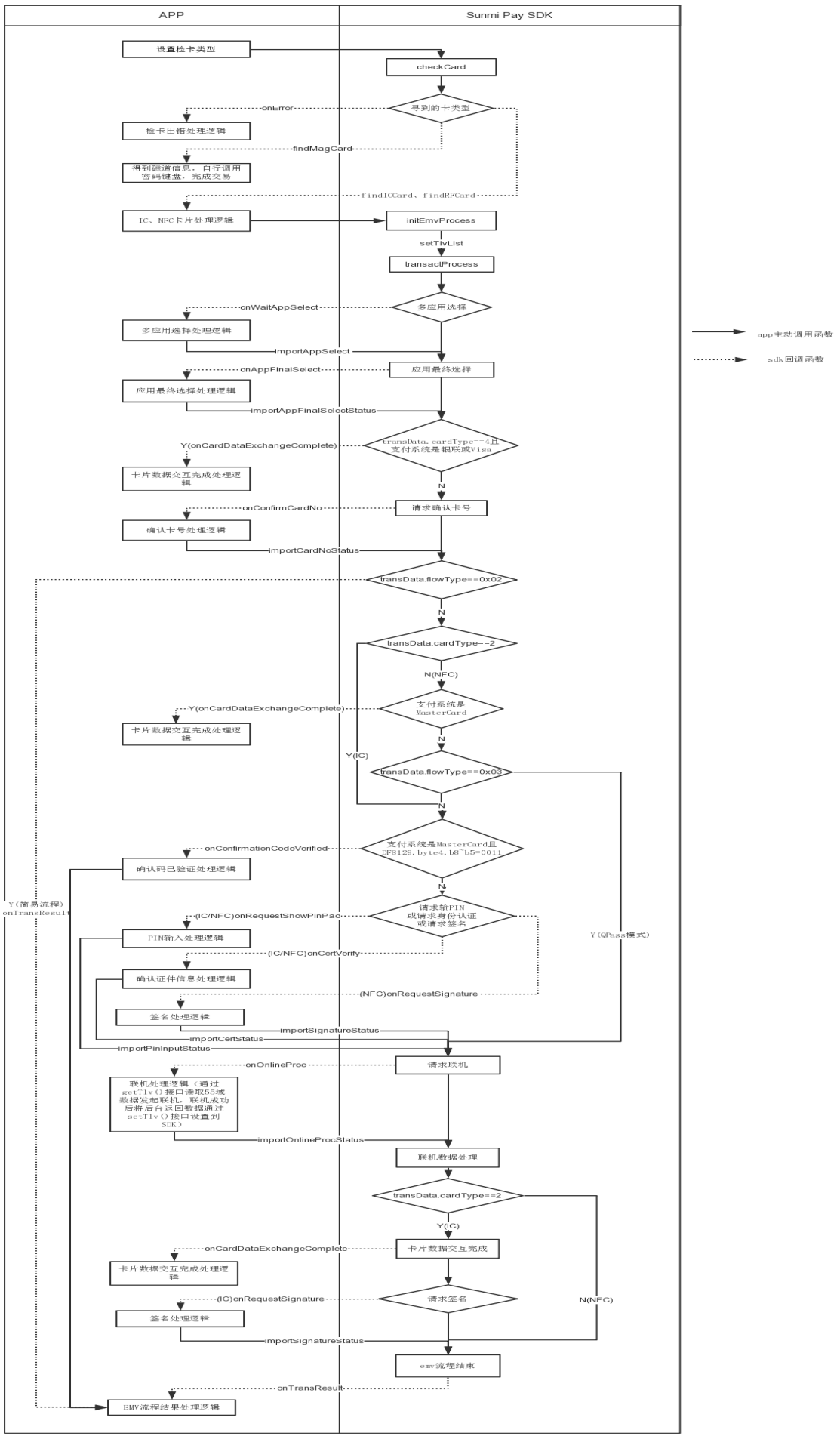
```
//（支持 DES、TDES）无意义
public static final int SEC_PIN_BLK_ISO_FMT1 = 1;
//（支持 DES、TDES）输入格式化后的 12 位 ASCII PAN
public static final int SEC_PIN_BLK_ISO_FMT3 = 3;
//（仅支持 AES） 输入格式化后的 12-19 位 ASCII PAN
public static final int SEC_PIN_BLK_ISO_FMT4 = 7;
```

## 7.2 PAN 数据截取

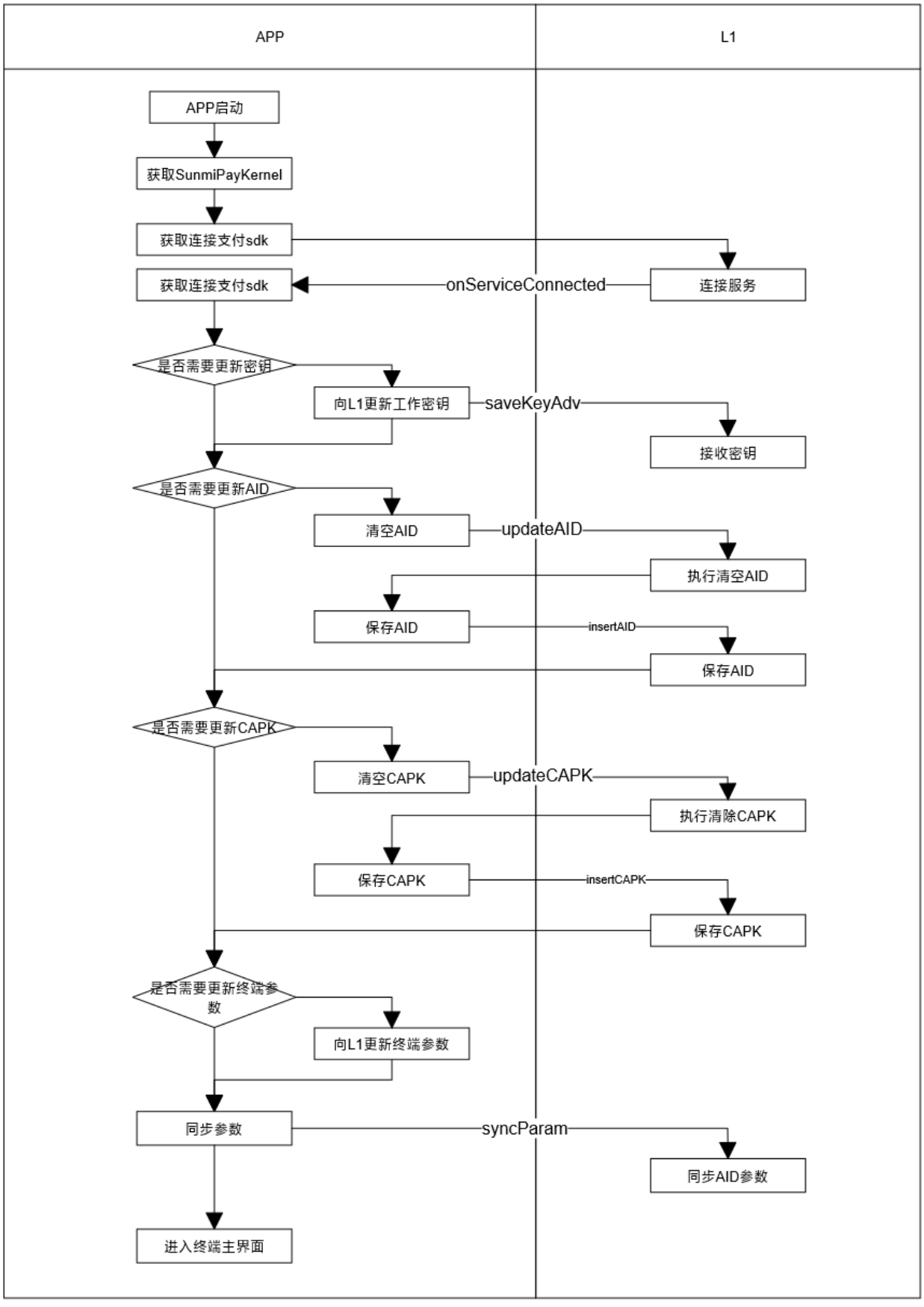
从卡号（2域）右边数第二位开始，向左取12位，作为参与PIN加、解密的PAN。
比如卡号为 6225882145611077
截取后的 PAN 数据 588214561107
转换为 byte[]数组“588214561107”.getBytes(“US-ASCII”);
SDK 中所有需要传入的 PAN 数据都是按照该种方式截取，如果 pinblock 结果错误在确认 pinKey 正确的情况下，按照上述规则检查传入的 PAN 数据

## 7.3 EMV 交易流程图

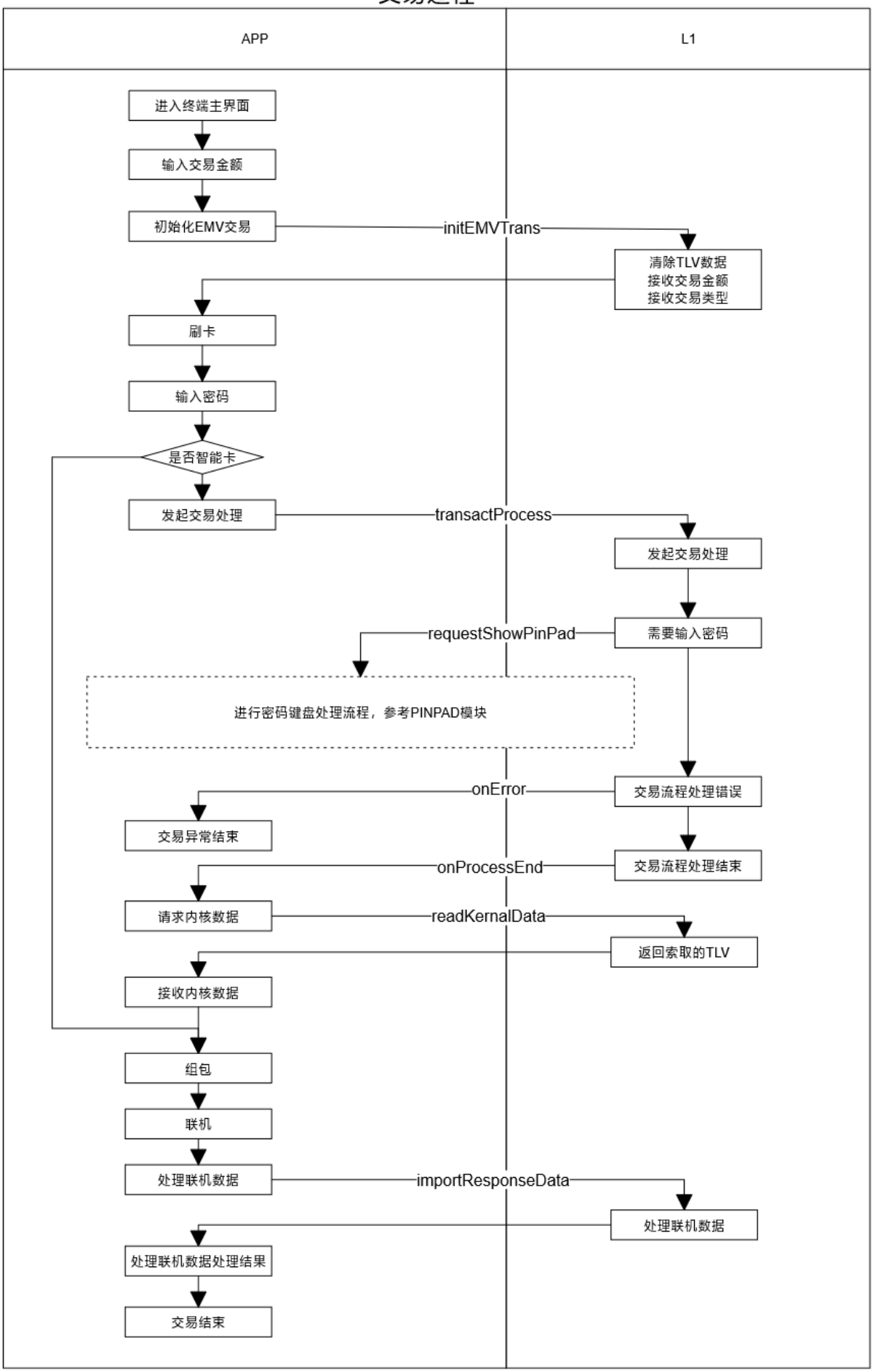
下面的流程图对 EMV 流程 SDK 与应用交互进行了描述，请按照流程进行开发


APP
Sunmi Pay SDK
设置检卡类型
checkCard
寻到的卡类型
onError
检卡出错处理逻辑
findMagCard
得到磁道信息，自行调用密码键盘，完成交易
findICCard、findRFCard
IC、NFC卡片处理逻辑
initEmvProcess
setTlvList
transactProcess
多应用选择
onWaitAppSelect
多应用选择处理逻辑
importAppSelect
应用最终选择
onAppFinalSelect
应用最终选择处理逻辑
importAppFinalSelectStatus
transData.cardType==1且支付系统是银联或Visa
Y(onCardDataExchangeComplete)
卡片数据交互完成处理逻辑
N
请求确认卡号
onConfirmCardNo
确认卡号处理逻辑
importCardNoStatus
transData.flowType==0x02
N
transData.cardType==2
N(NFC)
支付系统是MasterCard
Y(onCardDataExchangeComplete)
卡片数据交互完成处理逻辑
N
Y(IC)
transData.flowType==0x03
N
支付系统是MasterCard且DF8129.byte4.b8~b5=0011
onConfirmationCodeVerified
确认码已验证处理逻辑
N
请求输PIN或请求身份认证或请求签名
(IC/NFC)onRequestShowPinPad
Y(简易流程)onTransResult
PIN输入处理逻辑
(IC/NFC)onCertVerify
确认证件信息处理逻辑
(NFC)onRequestSignature
签名处理逻辑
importSignatureStatus
importCertStatus
importPinInputStatus
Y(QPass模式)
请求联机
onOnlineProc
联机处理逻辑（通过getTlv()接口读取55域数据发起联机，联机成功后将后台返回数据通过setTlv()接口设置到SDK）
importOnlineProcStatus
联机数据处理
transData.cardType==2
Y(IC)
卡片数据交互完成
onCardDataExchangeComplete
卡片数据交互完成处理逻辑
请求签名
(IC)onRequestSignature
N(NFC)
签名处理逻辑
importSignatureStatus
emv流程结束
onTransResult
EMV流程结果处理逻辑
app主动调用函数
sdk回调函数

# 准备过程


APP
L1
APP启动
获取SunmiPayKernel
获取连接支付sdk
获取连接支付sdk
onServiceConnected
连接服务
是否需要更新密钥
向L1更新工作密钥
saveKeyAdv
接收密钥
是否需要更新AID
清空AID
updateAID
执行清空AID
保存AID
insertAID
保存AID
是否需要更新CAPK
清空CAPK
updateCAPK
执行清除CAPK
保存CAPK
insertCAPK
保存CAPK
是否需要更新终端参数
向L1更新终端参数
同步参数
syncParam
同步AID参数
进入终端主界面

# 交易过程


APP
L1
进入终端主界面
输入交易金额
初始化EMV交易
initEMVTrans
清除TLV数据
接收交易金额
接收交易类型
刷卡
输入密码
是否智能卡
发起交易处理
transactProcess
发起交易处理
requestShowPinPad
需要输入密码
进行密码键盘处理流程，参考PINPAD模块
onError
交易流程处理错误
交易异常结束
onProcessEnd
交易流程处理结束
请求内核数据
readKernalData
返回索取的TLV
接收内核数据
组包
联机
处理联机数据
importResponseData
处理联机数据
处理联机数据处理结果
交易结束

# 7.4 密钥索引说明

### 7.4.1 密钥索引概述

**SP(SecureProcessor)**提供了 200 个普通密钥（MKSK 密钥，包含 3DES/AES/SM4 等）空间，10 个 Dukpt-3DES 密钥空间、10 个 Dukpt-AES 密钥空间、100 个 Dukpt-3DES 扩展密钥空间。各密钥类型对应的 SP 索引及 SunmiPayHardwareService(SPHS)的处理规则如下：

| 密钥体系 | 算法类型 | SP 提供的密钥索引 | SPHS 处理方法 |
|---|---|---|---|
| MKSK | 3DES/AES/SM4 | 1-200 | 使用**密钥分区(KeyPartition)**：每个分区专属于一个客户端 App，仅能由这个 App 访问。App 访问专属密钥分区时验证其开发者签名和包名，不同的 App 不能访问非自己专属的密钥分区。 |
| DUKPT | 3DES | 1-10 | -- |
| DUKPT | AES | 11-20 | -- |
| DUKPT | 3DES 扩展 | 1101-1200 | 使用**密钥专属**规则：每个密钥都有保存索引与 SP 索引的完整映射。<br>例：客户端 App1 保存密钥至 1100➔映射 SP 1101，App2 保存密钥至 1100➔映射 SP 1102。<br>密钥的删除、回收规则与**密钥分区**类同 |

### 7.4.2 支持密钥分区的机型

| 机型 | 是否默认支持密钥分区 | 能否代码开/关密钥分区 |
|---|---|---|
| P1N/P1_4G | 否 | 能 |
| P2lite | 是 | 能 |
| P2/P2_Pro | 是 | 能 |
| P2Mini | 是 | 能 |

### 7.4.3 密钥分区实施规则

SunmiPayHardwareService 在逻辑上将 200 个 MKSK 密钥索引划分为 10 个分区(KeyPartition)，第 1 个分区由 SunmiPayHardwareService 自己使用，其余 9 个分区提供给客户端 App 使用。每个分区的 keyIndex 范围为 0~19，客户端 App 可以在这个 keyIndex 范围内进行保存、使用、删除密钥等操作。当客户端 App 卸载后，如果有新的客户端 App 访问密钥分区且 9 个密钥分区均已分配完，则已卸载 App 的密钥分区会被回收，其保存的密钥会丢失（无法访问到）。

### 7.4.4 密钥索引映射规则

SP 密钥索引从 1 开始，1~20 由 SunmiPayHardwareService 使用
客户端 App 1（0~19）➔ SP(21~40)

客户端 App 2（0~19）➔ SP(41~60)

……

客户端 App 9（0~19）➔ SP(181~200)