

上海市杨浦区淞沪路388号创智天地7号楼3楼

SUNMI PAY SDK V2开发文档v3.2.2

上海商米科技有限公司

目录

[ 1. 版本控制 4](#_Toc155184372)

[ 2. 概述 15](#_Toc155184373)

[2.1 引言 15](#_Toc155184374)

[2.2 快速集成商米支付SDK 15](#_Toc155184375)

[ 3. API 16](#_Toc155184376)

[3.1 SunmiPayKernel SDK操作对象 16](#_Toc155184377)

[3.2 公共成员变量 17](#_Toc155184378)

[3.3 基础操作模块 18](#_Toc155184379)

[3.4 卡操作模块 30](#_Toc155184380)

[3.5 密码键盘模块（TOSS不支持） 50](#_Toc155184381)

[3.6 安全模块 56](#_Toc155184382)

[3.7 EMV操作模块 85](#_Toc155184383)

[3.8 税控模块（TOSS不支持） 99](#_Toc155184384)

[3.9 打印模块（仅内部使用） 100](#_Toc155184385)

[3.10 ETC模块（TOSS不支持） 104](#_Toc155184386)

[3.11 设备证书管理模块（TOSS不支持） 109](#_Toc155184387)

[ 4. 错误码 110](#_Toc155184388)

[ 5. 实体类 120](#_Toc155184389)

[5.1 EmvTermParamV2 – 终端参数实体类 120](#_Toc155184390)

[5.2 AidV2 -AID实体类 120](#_Toc155184391)

[5.3 CapkV2 -CAPK实体类 121](#_Toc155184392)

[5.4 EMVTransDataV2 - 交易处理实体类 121](#_Toc155184393)

[5.5 EMVCandidateV2 – EMV应用候选人 122](#_Toc155184394)

[5.6 PinPadConfigV2 - 密码键盘配置实体类 122](#_Toc155184395)

[5.7 PinPadTextConfigV2- 密码键盘显示文字配置实体类 122](#_Toc155184396)

[5.8 PinPadDataV2- 密码键盘UI布局数据 123](#_Toc155184397)

[5.9 PinPadDataV2Ex- 密码键盘UI布局数据（扩展） 123](#_Toc155184398)

[5.10 DrlV2 – DRL LimitSet实体类 123](#_Toc155184399)

[5.11 RevocListV2– RevocationList实体类 124](#_Toc155184400)

[5.12 ETCInfoV2实体类 124](#_Toc155184401)

[ 6. 访问权限 124](#_Toc155184402)

[6.1 权限位置 124](#_Toc155184403)

[6.2 权限定义 124](#_Toc155184404)

[ 7. 附录 126](#_Toc155184405)

[7.1 Aidl常量类（com.sunmi.pay.hardware.aidl.AidlConstants） 126](#_Toc155184406)

[7.2 PAN数据截取 137](#_Toc155184407)

[7.3 读卡流程图 138](#_Toc155184408)

[7.4 EMV交易流程图 138](#_Toc155184409)

[7.5 密钥注入/使用流程图 142](#_Toc155184410)

[7.6 密钥体系及密钥索引范围 142](#_Toc155184411)

[7.7 密钥分区说明 143](#_Toc155184412)

[7.8 KCV参数说明 143](#_Toc155184413)

## 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）  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  2.修改获取/设置TLV数据的接口  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  2.更新EMV流程图 | SunmiPayHardwareService\_v3.3.0 |  |
| 3.2.7 | 2018/11/15 | 1.更新PinPadConfigV2类说明  2.修改getKeyCheckValue出参说明  3.Emv流程增加以下接口：  (1)onAppFinalSelect()  (2)importAppFinalSelectStatus()  4.移除apduCommandNISO()接口 | SunmiPayHardwareService\_v3.3.0 |  |
| 3.2.8 | 201/11/16 | 1.smartCardExChangeNISO()接口改名为transmitApdu()  2.transmitApdu()直接返回卡片数据 | SunmiPayHardwareService\_v3.3.0 |  |
| 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()  2.增加错误码-50027  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()  2.添加错误码-50028  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  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()  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  2.增加获取初始化KSN接口：dukptGetInitKsn()  3.更新transmitApdu()接口，支持接触卡透传apdu  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  2.获取获取初始化KSN接dukptGetInitKsn()，获取ksn时不依赖saveKeyDukpt()接口  3.添加接口EMVOptV2. abortTransactProcess() | SunmiPayHardwareService\_v3.3.29 |  |
| 3.2.22 | 2019/04/10 | 1.添加MifarePlus卡数据交互接口  2.添加SLE4442/4428卡数据交互接口  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();  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  2.添加transmitApduEx()接口，改进Mifare卡apdu透传  3.移除修改记录中[适配的SDK版本]栏的Firmware版本  4.添加sysGetRandom()接口  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()  2. CheckCardCallbackV2()中增加回调接口findICCardEx()、findRFCardEx()、onErrorEx()  3.CheckCardCallbackV2. findMagCard()返回数据中增加cardType、trackErrorCode等字段  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  2.修改AidV2，增加clsStatusCheck、zeroCheck字段，支持DRL  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  2.修改输PIN提示文字  3.增减错误码：-70001~ -70006 | SunmiPayHardwareService\_v3.3.53 |  |
| 3.2.28 | 2019/10/28 | 1.EmvOptV2增加接口queryAidCapkLis()、transactPreProcess()  2.PinPadOptV2增加接口cancelInputPin()，支持代码取消输PIN  3.NFC检卡返回数据中增加卡类别、ATQA | SunmiPayHardwareService\_v3.3.55 |  |
| 3.2.29 | 2019/11/29 | 1.安全模块增加接口dataEncryptDukptEx()、dataDecryptDukptEx()、calcMacDukptEx()、verifyMacDukptEx()、saveTR31Key()、saveCiphertextKeyRSA()、saveRSAKey()  2.增加获取系统参数关键字：AEVersion  3.EMV flowType增加为0x04的类型  4.EMV模块增加接口函数addRevocList()、deleteRevocList()、sysSetTime()、sysGetTime()、clearData()  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  2.增加获取系统参数关键字FLASHVersion  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()  2.密码键盘模块增加设置显示文字接口：setPinPadText()  3.安全模块增加删除密钥接口deleteKey()、保存dukpt-AES密钥接口saveKeyDukptAES()  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()  2.系统模块增加接口setStatusBarDropDownMode()、setNavigationBarVisibility()、setHideNavigationBarItems()、sysPowerManage()  3.系统参数增加关键字：DPASVersion、APEMVVersion  4.增加EMV交易结果常量AidlConstants.EMV.TransResult  5.修改importOnlineProcStatus()接口参数status描述 | SunmiPayHardwareService\_v3.3.66 |  |
| 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()  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  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()  2.EMVOptV2增加接口setAccountDataSecParam()、getAccountSecData()  3.PinPadOptV2中增加接口：initPinPadEx()  4.系统参数增加关键字：PUREVersionFul、EFTPOSVersionFull、APEMVVersionFull  5.修改Dukpt-3DES密钥范围为0-9或1100-1199  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  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  2.AidlConstants.EMV.TLVOpCode中增加OP\_ADD\_SELF\_DEFINE\_TAG、OP\_DEL\_SELF\_DEFINE\_TAG  2.系统参数增加关键字SecMode、PCD\_IFMVersion | SunmiPayHardwareService\_v3.3.98 |  |
| 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  2.修改ETC模块扣费相关接口，搜索OBU改为回调方式  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()  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  2.安全模块增加校验Mac、RSA、RKI相关的扩展接口  3.安全模块更新MKSK、dukpt密钥索引范围  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  2.更新dukptIncreaseKSN()接口的描述 | SunmiPayHardwareService\_v3.3.137 |  |
| 3.2.51 | 2021/10/21 | 1.更新EMVListenerV2.onTransResult()接口中参数code值说明 | SunmiPayHardwareService\_v3.3.137 |  |
| 3.3.52 | 2021/11/11 | 1.EMV模块增加接口EMVListenerV2. onPreFirstGenAC()，EMVOptV2. importPreFirstGenACStatus()  2.Basic模块增加接口BasicOptV2. sysSetWakeup()  3.更新错误码表  4.更新接口 BasicOptV2.buzzerOnDevice() | SunmiPayHardwareService\_v3.3.149 |  |
| 3.2.53 | 2021/12/23 | 1.setPinPadMode()和getPinPadMode ()接口增加monitorClearKey关键字 | SunmiPayHardwareService\_v3.3.160 |  |
| 3.2.54 | 2022/01/25 | 1.EMVOptV2. setTermParamEx()接口增加关键字contactlessManualSelAppGeneralEx  2.AidV2增加extSelectSupFlg字段 | SunmiPayHardwareService\_v3.3.166 |  |
| 3.2.55 | 2022/03/08 | 1.SysParam增加KBBeepMode关键字 | SunmiPayHardwareService\_v3.3.170 |  |
| 3.2.56 | 2022/04/08 | 1.修改calcMac()接口返回值说明 | SunmiPayHardwareService\_v3.3.170 |  |
| 3.2.57 | 2022/05/07 | 1.SecurityOptV2增加接口secKeyIoControl(), apacsMac()  2.AidlConstants.Secuirty增加Mac类型MAC\_ALG\_X9\_19\_DEA  3.AidlConstants.SysParam增加关键“SAM” | SunmiPayHardwareService\_v3.3.178 |  |
| 3.2.58 | 2022/07/07 | 1. EMV增加接口EMVListenerV2.onDataStorageProc（），EMVOptV2.importDataStorage ();  2.EMVOptV2.setTermParamEx()参数增加key：dpasV2Support、dpasDeferredAuthSupport、dpasDataStorageSupport、dpasExtendedLoggingSupport、dpasTearingRecoverySupport | SunmiPayHardwareService\_v3.3.191 |  |
| 3.2.59 | 2022/08/17 | 1.SecurityOptV2中增加saveKeyEx()接口 | SunmiPayHardwareService\_v3.3.191 |  |
| 3.2.60 | 2022/08/20 | 1.SecurityOptV2中generateRSAKeys(),getRSAPublicKey(),getRSAPrivateKey(),dataEncryptRSA(),dataDecryptRSA(),removeRSAKey(),signingRSA(),verifySignatureRSA(),saveCiphertextKeyRSA(),saveRSAKey()接口标记为deprecated  2.添加读卡错误码-12549：A卡不支持ISO14443-4，激活流程终止  3.setPinPadMode()/getPinPadMode()支持cancelToClear模式  4.删除【7.4密钥索引说明】章节  5.修改rsaEncryptOrDecryptData()接口说明  6.添加【7.4 密钥注入/使用流程图】 | SunmiPayHardwareService\_v3.3.215 |  |
| 3.2.61 | 2022/11/09 | 更新EMV相关返回码描述 | SunmiPayHardwareService\_v3.3.215 |  |
| 3.2.62 | 2022/11/10 | 1.BasicOptV2增加接口installSharedLib()、deleteSharedLib()  2.injectCiphertextKeyEx()接口增加dataMode、iv参数  3. injectKeyDukptEx()接口增加dataMode、iv参数  4. saveKeyEx()接口增加dataMode、iv、isEncrypt参数  5. 增加错误码-20000：功能尚不支持、-2530：卡片拒绝命令 | SunmiPayHardwareService\_v3.3.225 |  |
| 3.2.63 | 2022/11/25 | 1.SecurityOptV2增加SecrutiyOptV2增加generateSymKey()、injectSymKey()、generateSymKeyEx()、injectSymKeyEx()接口  2.AidlConstants.Security增加 注入对称密钥模式常量 | SunmiPayHardwareService\_v3.3.228 |  |
| 3.2.64 | 2022/12/20 | 1.添加设备证书管理模块 | SunmiPayHardwareService\_v3.3.232 |  |
| 3.2.65 | 2022/12/22 | 1.接口中的“None”改为“无”  2.删除无参接口中的[in]/[out]描述 | SunmiPayHardwareService\_v3.3.233 |  |
| 3.2.66 | 2023/01/06 | 1.SecurityOptV2增加generateRSAKeypairEx()、injectRSAKeyEx()接口  2.AidlConstants.Security增加常量KEY\_TYPE\_RSA\_KPK、KEY\_TYPE\_RSA\_KEK  3.新增关闭密钥分区功能  4.更新安全模块接口参数keyIndex的说明  5.更新【7.4 密钥体系及密钥索引范围】 表 | SunmiPayHardwareService\_v3.3.234 |  |
| 3.2.66T02 | 2023/01/12 | 1. transmitMultiApdus()单次最大发送条数改为7条； 2. checkCardEnc()接口参数入参Bundle增加key: encKeyAlgType,表示密钥的算法类型; 3. calcSecHash(),sm2DecryptData(),sm2EncryptData(),sm2VerifySign(),sm2Sign()等接口，dataIn参数长度描述改为“小于等于”； 4. sm2Sign(),sm2VerifySign()接口dataIn参数最大支持长度改为1024字节； | \ |  |
| 3.2.67 | 2023/02/08 | 1. PinPadOptV2增加接口setVisualImpairmentModeParam()，getVisualImpairmentModeParam(); 2. initPinPadEx()接口参数pan,增加长度描述12~19位； 3. transmitApduExx()接口参数ctrCode增加bit6描述； 4. 错误码增加-2904，-2909； 5. rsaEncryptOrDecryptData()接口，更新参数长度范围描述； 6. sm2EncryptData()，sm2DecryptData接口，更新参数长度范围描述； 7. sm2VerifySign()接口，更新参数长度范围描述； 8. sm2Sign()接口，更新参数长度范围描述； 9. saveKeyDukpt()接口，增加备注； | SunmiPayHardwareService\_v3.3.238 |  |
| 3.2.68 | 2023/03/10 | 1. setGrayLevel()接口，设置灰度值增加备注说明； 2. 添加clearBuffer()，saveBaseKey()接口描述； 3. setPrintSpeed()，calcSecHash, smartCardIoControl接口，更新参数描述； 4. 错误码增加-80001，-2911； 5. savePlaintextKey()接口，更新参数描述 |  |  |
| 3.2.69 | 2023/04/21 | 1. injectCiphertextKey()接口，更新备注； 2. injectSymKey()接口，更新备注； 3. injectSymKeyEx()接口，更新备注； 4. NONEwithRSA模式签名增加备注说明； 5. saveCiphertextKeyRSA()接口，更新参数描述； |  |  |
| 3.2.70 | 2023/05/10 | 1. setVisualImpairmentModeParam()、getVisualImpairmentModeParam()接口更新参数ttsLanguage的值范围 2. 更新kcv参数描述 3. saveKeyEx()接口增加参数kcvMode 4. 更新saveTR31Key()接口描述 5. 更新错误码表，增加-16000~-16008错误码 | SunmiPayHardwareService\_v3.3.304 |  |
| 3.2.71 | 2023/05/12 | 1.更新dataEncryptRSA()、  dataDecryptRSA()、signingRSA()、verifySignatureRSA()接口描述  2.BasicOptV2、ReadCardOptV2、SecurityOptV2、PinPadOptV2、EMVOptV2增加巴西CKD、TOSS专用接口  3.更新【密钥体系及密钥索引范围】备注栏 | SunmiPayHardwareService\_v3.3.307 |  |
| 3.2.72 | 2023/05/31 | 1.更新打印机状态常量，增加打印机开盖状态码  2.更新错误码表，增加-7010、-7011错误码  3.更新EMV交易流程图  4.更新injectDeviceCertPrivateKey()接口说明  5. injectCiphertextKeyEx()、injectKeyDukptEx()接口增加kcvMode、kcvMacType、kcvInData参数 | SunmiPayHardwareService\_v3.3.308 |  |
| 3.2.73 | 2023/06/10 | 1. SecurityOptV2增加接口：  dataEncryptEx()、dataDecryptEx()  2. 更新【7.4 密钥体系及密钥索引范围】 表 | SunmiPayHardwareService\_v5.0.15 |  |
| 3.2.74 | 2023/07/05 | 1.更新setPreferredNetworkMode()、readRSAKey()、getKeyLength()、writeKeyVariable()、generateRSAKeypairEx()，injectRSAKeyEx()接口说明  2.更新setPrintHeatPoint()接口说明  3.更新dataEncryptEx()、dataDecryptEx()接口说明  4.更新【7.4 密钥体系及密钥索引范围】 表  5.更新错误码表 | SunmiPayHardwareService\_v5.0.16 |  |
| 3.2.75 | 2023/07/13 | 1.PinPadOptV2.setPinPadMode()、getPinPadMode()接口增加关键字 longTimeoutTime | SunmiPayHardwareService\_v3.3.311 |  |
| 3.2.76 | 2023/08/07 | 1. PinPadOptV2. setVisualImpairmentModeParam()、getVisualImpairmentModeParam()接口参数bundle增加关键字：rnibSelectMode、rnibHoldTime  2.更新checkCard()、checkCardEx()、checkCardEnc()、checkCardForToss()接口说明 | SunmiPayHardwareService\_v3.3.313 |  |
| 3.2.77 | 2023/08/22 | 1.更新接口setVisualImpairmentModeParam()说明 | SunmiPayHardwareService\_v3.3.313 |  |
| 3.2.78 | 2023/09/04 | 1.添加BasicOptV2.setSysParam()接口  2.更新模块或接口说明，对 TOSS不支持的接口增加说明 | SunmiPayHardwareService\_v3.3.313 |  |
| 3.2.79 | 2023/09/14 | 1.PinPadoptV2添加importPinPadData()接口  2.更新EMVOptV2. importPinInputStatus()接口，参数inputResult新增为4的值 | SunmiPayHardwareService\_v3.3.313 |  |
| 3.2.80 | 2023/10/11 | 1. PinPadOptV2添加importPinPadDataEx()接口 | SunmiPayHardwareService\_v3.3.318 |  |
| 3.2.81T | 2023/10/30 | 1. checkCard()、checkCardEx()接口增加磁道数据风险说明 | SunmiPayHardwareService\_v3.3.318 |  |
| 3.2.81 | 2023/11/15 | 1.toss检卡，磁条卡增加返回字段track2Raw | SunmiPayHardwareService\_v3.3.320 |  |
| 3.2.82 | 2023/11/29 | 1.更新7.1.30 LedLight常量定义 | SunmiPayHardwareService\_v5.0.22 |  |
| 3.2.83 | 2023/12/13 | 1.更新【2.2.1快速集成】，SDK支持机型中增加P2、P2\_SmartPad等机型，增加从maven仓库集成paylib.aar说明  2.更新getModuleAccessibility()、setModuleAccessibility()、getPedMode()、setPedMode()、setScheduleReboot()、clearScheduleReboot()、readRSAKey()接口说明，移除【巴西CKD专用】描述  3. 【7.1.11 RSA填充模式】中增加PADDING\_OAEP\_SHA1常量  4.增加【7.1.27 EMV联机结果定义】章节  5.更新【7.1.31 LedLight常量定义】，对四角灯、指示灯增加支持的机型说明  6.更新checkCard()、checkCardEx()、checkCardEnc()、checkCardForToss()接口说明，增加卡片下电描述  7.更新cardOff()接口描述  8.增加【7.3读卡流程图】章节  9.【3.3基础操作模块】增加getRtcBatVol()接口  10.【7.1.30 SysParam常量定义】增加关键字IfmLibVersion、MsrVersion | SunmiPayHardwareService\_v5.0.23 |  |
| 3.2.84 | 2023/12/26 | 1.【7.1.30 SysParam常量定义】增加关键字posapiVersion、RTCBATVOLDET | SunmiPayHardwareService\_v5.0.24 |  |
| 3.2.85 | 2024/01/30 | 1.保存密钥、设备证书接口增加密钥永久性说明、密钥可删除说明  2.更新3.7.1.13~3.7.1.19接口说明中增加“导入”描述  3.更新calcSecHash()接口说明  4.更新CheckCardEnc()接口，支持RSA加密磁道数据，参数bundle中增加key panAppendContent、panAppendMode  5.更新CheckCardCallbackV2. findMagCard()接口说明，增加appendedPanEnc、appendedPanEncBytes字段说明  6.更新transmitApdu()、transmitApduEx()、transmitApduExx()接口中参数sendBuff、recvBuff的说明  7.卡模块增加smartCardExChangePASS()、smartCardExChangePASSNoLength()接口  8.更新initPinPad()、initPinPadEx()、startInputPin()接口，参数pinPadType的范围从0-1扩展到0-5  9.PinPadListenerV2增加onHover()接口  10.更新【7.1.33 PinBlock格式】 | SunmiPayHardwareService\_v3.3.324 |  |
| 3.2.86 | 2024/03/07 | 1. 增加AidlConstants.EMV. TLVOpCode.OP\_CPACE, AidlConstants.EMV.KernelType.CPACE, AidlConstants.SysParam.CPACE\_VERSION、AidlConstants.SysParam.CPACE\_RELEASE\_DATE  2. setTermParamEx()参数bundle中增加关键字：supportPOI、CertifiedEP、AutoRun、KernelsForCertEP | SunmiPayHardwareService\_v3.3.326 |  |
| 3.2.87 | 2024/03/13 | 1.SecurityOptV2中增加queryKeyMappingRecordList()接口 | SunmiPayHardwareService\_v3.3.326 |  |
| 3.2.88 | 2024/03/20 | 1.对接口checkCard(), checkCardEx(), getTlv(),getTlvList(),setAccountDataSecParam(), onConfirmCardNo()增加sred说明  2. 增加【7.9 SRED说明】章节  3.SecurityOptV2中增加queryKeyMappingRecordListWL()和injectTR31Key()接口  4.BasicOptV2中增加readPuk()接口  5.更新BasicOptV2.setPedMode()、getPedMode()接口说明 | SunmiPayHardwareService\_v3.3.328  SunmiPayHardwareService\_v5.0.28 |  |
| 3.2.89 | 2024/04/5 | 1. SecuirtyOptV2中增加readSM2Key()、calcSM3HashWithID()、sm2SingleSign()接口 | SunmiPayHardwareService\_v3.3.328 |  |
| 3.2.90 | 2024/04/23 | 1.【7.1.14 注入对称密钥模式】增加注入对称密钥模式常量INJECT\_OAEP\_MODE、INJECT\_PKCS1\_MODE  2.更新SecurityOptV2.injectSymKeyEx()接口描述 | SunmiPayHardwareService\_v3.3.329 |  |
| 3.2.91 | 2024/05/17 | 1.【3.5密码键盘模块】接口增加对有物理键盘的设备的说明  2.更新【7.5 密钥注入/使用流程图】，增加svg格式的流程图  3.更新【7.1.30 SysParam常量定义】，增加emv库的发布日期关键字 | SunmiPayHardwareService\_v3.3.331  SunmiPayHardwareService\_v5.0.32 |  |
| 3.2.92 | 2024/06/07 | 1.更新getCardExistStatus()接口的参数cardType的描述 | SunmiPayHardwareService\_v3.3.331  SunmiPayHardwareService\_v5.0.32 |  |
| 3.2.93 | 2024/06/13 | 1.添加【7.10 EMV默认数据】章节  2.更新addAid()、deleteAid()、addCapk()、deleteCapk()接口说明  3.更新【7.1.4 MAC算法类型常量定义】，Mac算法类型增加MAC\_ALG\_HMAC\_SHA1、MAC\_ALG\_HMAC\_SHA256、MAC\_ALG\_CMAC | SunmiPayHardwareService\_v3.3.331  SunmiPayHardwareService\_v5.0.32 |  |
| 3.2.94 | 2024/06/14 | 1.更新checkCardEnc()接口参数描述  1.更新setAccountDataSecParam()接口的描述，参数bundle增加encKeyAlgType、panAppendContent、panAppendMode关键字  3.更新getAccountSecData()接口，返回值参数bundle中增加关键字appendedPanEncBytes、appendedPanEnc | SunmiPayHardwareService\_v5.0.33 |  |
| 3.3.95 | 2024/07/31 | 1.AidlConstants.SysParam中增加PCIPTSVersion、RNIBVersion、sred关键字  2.修改cardOff()接口，卡片下电时支持复合卡类型  3.更新startInputPin()、getPinBlock()接口的说明，移除巴西CKD专用限制  4.更新setAccountDataSecParam()接口描述  5.更新setTermParamEx()接口描述，增加关键字:  “quickChip”，  “noSignatureOrPINThreshold”，  “dpasContactlessSpeedupSupport”，  “jcbContactlessSpeedupSupport”，  “AEContactlessSpeedupSupport”，  “AEOnlineProcessSupport”，  "SupportAE4.1"  6.更新【7.9 SRED说明】章节  7.更新SetSysParam()接口，增加设置sred说明  8.更新transmitApdu()接口说明 | SunmiPayHardwareService\_v5.0.34 |  |
| 3.2.96 | 2024/08/17 | 1.增加【3.12 HCE模块】  2.更新错误码表，增加-2401~-2408 HCE功能相关的错误码 | SunmiPayHardwareService\_v5.0.35 |  |
| 3.2.97 | 2024/08/29 | 1.更新CheckCard()、CheckCardEx()、CheckCardEnc()、checkCardForToss()接口，检卡超时时间改为600s  2.更新PinPadListenerV2.onHover()接口，参数data增加event=6时的说明 | SunmiPayHardwareService\_v3.3.335 |  |

## 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（AndroidStudio）的基本使用方法，具有一定的Android 编程基础，熟悉金融规范、流程以及相关概念（密钥，pinblock，pan，emv，mac等概念）。目前SDK只支持在商米金融设备P1N, P1\_4G, P2，P2 Pro, P2Lite，P2mini，P2\_SmartPad，P2\_SE，P2\_Xpro，TOSS，P2liteSE，P3\_MIX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，P2Pro，P2Lite，P2mini，P2\_SmartPad，P2\_SE，P2\_Xpro，TOSS，P2liteSE，P3\_MIX中的任一种（在设置-关于设备-型号 可以查看设备型号）。
2. 打开设置—>应用—>（右上角）点击选择显示系统应用，确认设备已安装了v3.3.xx版本的 SunmiPayHardwareService。
3. AndroidStudio快速集成SDK aar库：

（1）本地引用：将PayLib\_vx.x.x.aar文件放在libs目录下，并在build.gradle中添加以下代码：

|  |
| --- |
| repositories {  flatDir {  dirs ‘libs’  }  }  dependencies {  …...  compile(name: ‘PayLib’, ext: ‘aar’)  } |

（2）maven仓库引用：在项目根目录的build.gradle中增加以下代码：

|  |
| --- |
| buildscript {  repositories {  mavenCentral()  }  }  allprojects {  repositories {  mavenCentral()  }  } |

然后在app/build.gradle中增加以下代码：

|  |
| --- |
| dependencies {  implementation 'com.sunmi:PayLib-release:2.0.07'  } |

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 | 无 |
| 返回值 | SunmiPayKernel |
| 备注 |  |

#### 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](#ConnectCallback) |
| 返回值 | 无 | |
| 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 | 无 |
| 返回值 | 无 |
| 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 | 无 |
| 返回值 | 无 |
| 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 | 无 |
| 返回值 | 无 |
| 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  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 |
| mBasicOptV2 | 基础信息模块 | 包含获取基础信息，控制LED灯，蜂鸣器等API |
| mReadCardOptV2 | 卡操作模块 | 包含读银行卡，会员卡，M1卡等相关操作API |
| mPinPadOptV2 | 密码键盘模块 | 包含安全的密码键盘相关API |
| mSecurityOptV2 | 安全模块 | 包含保存密钥，mac计算，加密等API |
| mEMVOptV2 | EMV模块 | 包含EMV流程的相关API |
| mTaxOptV2 | 税控模块 | 包含税控操作API |
| mETCOptV2 | ETC模块 | 包含保存ETC功能相关的API |
| mDevCertManagerV2 | 设备证书管理模块 | 包含设备证书功能相关的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](#_SysParam__SysParam常量定义) |
| 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 | 设置系统参数 | |
| 参数 | key[in] | 用户参数关键字见[AidlConstantsV2. SysParam](#_SysParam__SysParam常量定义) |
| value[in] | 值 |
| 返回值 | 0-成功  非0-错误码 | |
| 备注 | 1. 参数key、value参见[AidlConstantsV2. SysParam](#_SysParam__SysParam常量定义)定义的值：   |  |  |  | | --- | --- | --- | | key | value | 功能 | | SysParam.RESERVED | JSON字符串 | 配置非SDK默认项 | | SysParam.TERM\_STATUS | 以下值之一：  SysParam.CLEAR\_TAMPER\_LOG  SysParam.CLEAR\_TAMPER | 清除触发log  清除触发 | | SysParam.PINPAD\_MODE | 以下值之一：  PinPadMode.MODE\_NORMAL  PinPadMode.MODE\_MEITUAN  PinPadMode.MODE\_SILENT  PinPadMode.MODE\_LEDOFF | 设置PinPad模式 | | SysParam.PCD\_PARAM\_A | Hex字符串，例：95018F0F | 设置A卡非接参数 | | SysParam.PCD\_PARAM\_B | Hex字符串，例：8D018F03 | 设置B卡非接参数 | | SysParam.PCD\_PARAM\_C | Hex字符串，例：8D018F03 | 设置Felica非接参数 | | SysParam.SEC\_MODE | “0”或“1” | 设置开启/关闭密钥相同性检查标志，开启后不能保存相同key值的MKSK或DUKPT密钥 | | SysParam.KB\_BEEP\_MODE | 以下值之一：  KBBeepMode.MODE\_ON  KBBeepMode.MODE\_OFF | 设置实体键盘按键音模式 | | SysParam.EMV\_MASK | Hex字符串 | 设置EMV掩码 | | SysParam. SRED | “0” | 关闭sred，开启sred请调用[setAccountDataSecParam ()](#_3.7.1.35_设置账户数据安全参数)接口 |   2.key、value说明：  （1）所有key、value值建议使用SDK默认值，不建议修改  （2）TERM\_STATUS 为设备生产用参数，客户端请勿使用  （3）PINPAD\_MODE不建议使用，设置PinPad模式请使用PinPadOptV2.setPinPadMode()接口  （4）PCD\_PARAM\_A、PCD\_PARAM\_B、PCD\_PARAM\_C在P2、P2\_PRO机型上支持，在其他机型上不支持  （5）KB\_BEEP\_MOD仅在P2\_SmartPad上支持  （6）EMV\_MASK 仅用于EMV认证 | |

#### 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  非0-错误码 | |
| 备注 | P2lite/P2Pro/P2/P2mini/P2\_xpro设备上无实体蜂鸣器，蜂鸣声通过播放音频文件由扬声器发出，1.**freq**按以下规则处理：  （1）0-750，播放音频文件频率为750Hz  （2）751-1000，播放音频文件频率为1000Hz  （3）1001-1500，播放音频文件频率为1500Hz  （4）1501-1750，播放音频文件频率为1750Hz  （5）1751-2500，播放音频文件频率为2500Hz  （6）2501-2750，播放音频文件频率为2750Hz  （7）2751-3250，播放音频文件频率为3250Hz  （8）3250以上，播放音频文件频率为4000Hz  2.**duration** 默认为200ms，其他值无效 | |

#### 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](#LedLight常量定义) |
| ledStatus[in] | LED状态：0-亮，1-灭 |
| 返回值 | 0-成功  非0-错误码 | |
| 备注 | **1.P3上index6-8表示的物理LED灯为一组灯，index表示控制此组灯的某一种颜色，且颜色可以叠加。如同时亮6,7的灯，则灯的颜色显示为红色和绿色的叠加，为黄色**  **2.本接口在TOSS机型上不支持** | |

#### 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  非0-错误码 | |
| 备注 | 1.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  **2.本接口在TOSS机型上不支持** | |

#### 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  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  非0-错误码 | |
| 备注 | **1.本接口在TOSS机型上不支持** | |

#### 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  非0-错误码 | |
| 备注 | **1.本接口在TOSS机型上不支持** | |

#### 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  非0-错误码 | |
| 备注 | **1.本接口在TOSS机型上不支持** | |

#### 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：  STATUS\_BAR\_DISABLE\_HOME=0x00200000;//隐藏home键  STATUS\_BAR\_DISABLE\_BACK = 0x00400000;//隐藏back键  STATUS\_BAR\_DISABLE\_RECENT = 0x01000000;//隐藏recent键  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  非0-错误码 | |
| 备注 | **1.本接口在TOSS机型上不支持** | |

#### 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  非0-错误码 | |
| 备注 |  | |

#### 3.3.12 APP动态权限配置（巴西CKD专用）

|  |  |  |
| --- | --- | --- |
| 原型 | int allowDynamicPermission(String packageName) | |
| 功能 | APP动态权限配置 | |
| 参数 | packageName [in] | 目标APP包名 |
| 返回值 | 0-成功  非0-错误码 | |
| 备注 | **1.本接口为巴西CKD机型上专用接口，在非巴西CKD机型上不支持** | |

#### 3.3.13 设置全局wifi代理（巴西CKD专用）

|  |  |  |
| --- | --- | --- |
| 原型 | int setGlobalProxy(String proxy) | |
| 功能 | 设置全局wifi代理 | |
| 参数 | proxy[in] | 代理，格式为ip:port 或 url:port |
| 返回值 | 0-成功  非0-错误码 | |
| 备注 | **1.本接口为巴西CKD机型上专用接口，在非巴西CKD机型上不支持** | |

#### 3.3.14 安装CA证书（巴西CKD专用）

|  |  |  |
| --- | --- | --- |
| 原型 | int installApplicationCertificate(String name, String contents) | |
| 功能 | 安装CA证书 | |
| 参数 | name[in] | 证书名称 |
| contents[in] | 证书的内容 |
| 返回值 | 0-成功  非0-错误码 | |
| 备注 | **1.本接口为巴西CKD机型上专用接口，在非巴西CKD机型上不支持** | |

#### 3.3.15 卸载CA证书（巴西CKD专用）

|  |  |  |
| --- | --- | --- |
| 原型 | int uninstallApplicationCertificate(String name) | |
| 功能 | 卸载CA证书 | |
| 参数 | name[in] | 证书名称 |
| 返回值 | 0-成功  非0-错误码 | |
| 备注 | **1.本接口为巴西CKD机型上专用接口，在非巴西CKD机型上不支持** | |

#### 3.3.16 获取CPU使用率（巴西CKD专用）

|  |  |  |
| --- | --- | --- |
| 原型 | String getCpuUsage() | |
| 功能 | 获取CPU使用率 | |
| 参数 | [in] |  |
| 返回值 | 非空-CPU使用率，百分比格式，如返回20表示使用率为20%  空-失败 | |
| 备注 | **1.本接口为巴西CKD机型上专用接口，在非巴西CKD机型上不支持** | |

#### 3.3.17 获取CPU温度（巴西CKD专用）

|  |  |  |
| --- | --- | --- |
| 原型 | String getCpuTemperature() | |
| 功能 | 获取CPU温度 | |
| 参数 | [in] |  |
| 返回值 | 非空-CPU温度，单位：℃  空-失败 | |
| 备注 | **1.本接口为巴西CKD机型上专用接口，在非巴西CKD机型上不支持** | |

#### 3.3.18 设置定时重启

|  |  |  |
| --- | --- | --- |
| 原型 | int setScheduleReboot(int hour, int minute, int second, int millisecond) | |
| 功能 | 设置设备定时重启时间 | |
| 参数 | hour[in] | 时，范围：0~23 |
| minute[in] | 分，范围：0~59 |
| second[in] | 秒，范围：0~59 |
| millisecond[in] | 毫秒，范围：0~999 |
| 返回值 | 0-成功  非0-错误码 | |
| 备注 | **1.本接口的功能和Android系统的定时重启接口无关联关系，亦即，如果同时调用本接口和Android系统中的定时重启接口，则两处调用都会生效** | |

#### 3.3.19 清除定时重启

|  |  |  |
| --- | --- | --- |
| 原型 | int clearScheduleReboot() | |
| 功能 | 清除已设置的设备定时重启时间 | |
| 参数 | [in] |  |
| 返回值 | 0-成功  非0-错误码 | |
| 备注 |  | |

#### 3.3.20 自定义功能键（巴西CKD专用）

|  |  |  |
| --- | --- | --- |
| 原型 | int customizeFunctionKey(Bundle bundle) | |
| 功能 | 自定义功能键 | |
| 参数 | bundle[in] | 包含以下key：  key：String，按键名称，范围：“volume\_1”、“volume\_2”、“function\_1”、“function\_2”  type：String，功能类型，范围： “function”、 “native”、“app\_launch”  value：String，参数type对应的值：  1. 当type为”function”时，value的值可以为“volume\_up”或“volume\_down”  2. 当 type为“native”时， value的值只能为“native”  1.当type为“app\_launch“时，value的值为想要启动的APP的包名 |
| 返回值 | 0-成功  非0-错误码 | |
| 备注 | **1.本接口为巴西CKD机型上专用接口，在非巴西CKD机型上不支持** | |

#### 3.3.21 添加APP至LMK白名单（巴西CKD专用）

|  |  |  |
| --- | --- | --- |
| 原型 | int setLMKPackage(String packageName) | |
| 功能 | 添加APP至LMK白名单 | |
| 参数 | packageName [in] | 目标APP包名 |
| 返回值 | 0-成功  非0-错误码 | |
| 备注 | **1.本接口为巴西CKD机型上专用接口，在非巴西CKD机型上不支持** | |

#### 3.3.22 将APP从LMK白名单移除（巴西CKD专用）

|  |  |  |
| --- | --- | --- |
| 原型 | int removeLMKPackage(String packageName) | |
| 功能 | 将APP从LMK白名单移除 | |
| 参数 | packageName [in] | 目标APP包名 |
| 返回值 | 0-成功  非0-错误码 | |
| 备注 | **1.本接口为巴西CKD机型上专用接口，在非巴西CKD机型上不支持** | |

#### 3.3.23 设置设备唤醒源

|  |  |  |
| --- | --- | --- |
| 原型 | int sysSetWakeup(int channel, int mode, Bundle attr) | |
| 功能 | 设置设备唤醒源 | |
| 参数 | channel[in] | 唤醒源，1-IC卡唤醒，2-磁卡唤醒，3-按键唤醒 |
| mode[in] | 模式，0-关，1-开 |
| attr[in] | 其他属性，预留 |
| 返回值 | 0-成功  非0-错误码 | |
| 备注 | **1.本接口在TOSS机型上不支持** | |

#### 3.3.24 设置首选的网络类型（巴西CKD专用）

|  |  |  |
| --- | --- | --- |
| 原型 | int setPreferredNetworkMode(int mode, int slotIndex) | |
| 功能 | 设置首选的网络类型 | |
| 参数 | mode [in] | 网络类型，包含以下值：  0-GSM/WCDMA (WCDMA首选)  1-仅GSM  2-仅WCDMA  3-GSM/WCDMA (自动模式, 根据PRL) AVAILABLE Application Settings menu  4-CDMA and EvDo (自动模式, 根据PRL) AVAILABLE Application Settings menu  5-仅CDMA  6-及EvDo  7-GSM/WCDMA、CDMA,、EvDo (自动模式, 根据PRL) AVAILABLE Application Settings menu  8-LTE、CDMA、EvDo  9-LTE、GSM/WCDMA  10-LTE、CDMA、EvDo、GSM/WCDMA  11-仅LTE  12-LTE/WCDMA  13-仅TD-SCDMA  14-仅TD-SCDMA、WCDMA  15-TD-SCDMA、LTE  16-TD-SCDMA、GSM  17-TD-SCDMA、GSM、LTE  18-TD-SCDMA、GSM/WCDMA  19-TD-SCDMA、WCDMA、LTE  20-TD-SCDMA、GSM/WCDMA、LTE  21-TD-SCDMA、EvDo、CDMA、GSM/WCDMA  22-TD-SCDMA/LTE/GSM/WCDMA、CDMA、EvDo  30-LTE/GSM  31-仅LTE TDD  32-CDMA、GSM(2G Global)  33-CDMA、EVDO、GSM  34-LTE、CDMA、EVDO、GSM(4G Global, 4M) |
| slotIndex[in] | sim卡槽索引，范围：0~1 |
| 返回值 | 0-成功  非0-错误码 | |
| 备注 | **1.本接口为巴西CKD机型上专用接口，在非巴西CKD机型上不支持**  **2.参数mode的值必须为getSupportedNetworkType()接口返回值中的其中一种** | |

#### 3.3.25 获取支持的网络类型（巴西CKD专用）

|  |  |  |
| --- | --- | --- |
| 原型 | String getSupportedNetworkType(int slotIndex) | |
| 功能 | 获取支持的网络类型 | |
| 参数 | slotIndex[in] | sim卡槽索引，范围：0~1 |
| 返回值 | 非空-网络类型  空-失败 | |
| 备注 | **1.本接口为巴西CKD机型上专用接口，在非巴西CKD机型上不支持** | |

#### 3.3.26 打开或关闭飞行模式（巴西CKD专用）

|  |  |  |
| --- | --- | --- |
| 原型 | int setAirplaneMode(boolean enable) | |
| 功能 | 打开或关闭飞行模式 | |
| 参数 | enable[in] | 开关，true-打开飞行模式，false-关闭飞行模式 |
| 返回值 | 0-成功  非0-错误码 | |
| 备注 | **1.本接口为巴西CKD机型上专用接口，在非巴西CKD机型上不支持** | |

#### 3.3.27 打开或关闭数据漫游（巴西CKD专用）

|  |  |  |
| --- | --- | --- |
| 原型 | int setDataRoamingEnable(int slotIndex, boolean enable) | |
| 功能 | 打开或关闭数据漫游 | |
| 参数 | slotIndex[in] | sim卡槽索引，范围：0~1 |
| enable[in] | 开关，true-开启数据漫游，false-关闭数据漫游 |
| 返回值 | 0-成功  非0-错误码 | |
| 备注 | **1.本接口为巴西CKD机型上专用接口，在非巴西CKD机型上不支持** | |

#### 3.3.28 启用/禁用电话功能（巴西CKD专用）

|  |  |  |
| --- | --- | --- |
| 原型 | int enablePhoneCall(boolean enable) | |
| 功能 | 启用/禁用电话功能 | |
| 参数 | enable[in] | 开关，true-启用电话功能，false-禁用电话功能 |
| 返回值 | 0-成功  非0-错误码 | |
| 备注 | **1.本接口为巴西CKD机型上专用接口，在非巴西CKD机型上不支持** | |

#### 3.3.29 获取检卡成功/失败的的次数

|  |  |  |
| --- | --- | --- |
| 原型 | int getCardUsageCount(int cardType, boolean isSuccess) | |
| 功能 | 获取MAG刷卡、IC/NFC检卡成功/失败的的次数 | |
| 参数 | cardType [in] | 卡类型，值为MAG/IC/NFC |
| isSuccess[in] | true-成功的次数，false-失败的次数 |
| 返回值 | >=0-成功/失败的次数  <0-错误码 | |
| 备注 |  | |

#### 3.3.30 获取模块的可访问性

|  |  |  |
| --- | --- | --- |
| 原型 | int getModuleAccessibility(int module) | |
| 功能 | 获取模块的可访问性 | |
| 参数 | module[in] | 模块，1-MAG，2-ICC，3-PICC，4-PinPad |
| 返回值 | 0-禁用，1-启用  <0-错误码 | |
| 备注 |  | |

#### 3.3.31 设置模块的可访问性

|  |  |  |
| --- | --- | --- |
| 原型 | int setModuleAccessibility(int module, int ability) | |
| 功能 | 设置模块的可访问性 | |
| 参数 | module [in] | 模块，1-MAG，2-ICC，3-PICC，4-PinPad |
| ability[in] | 可访问性，0-禁用，1-启用 |
| 返回值 | 0-成功  非0-错误码 | |
| 备注 |  | |

#### 3.3.32 获取PED模式

|  |  |  |
| --- | --- | --- |
| 原型 | int getPedMode() | |
| 功能 | 获取PED（PIN Entry Device）模式 | |
| 参数 | [in] |  |
| 返回值 | >=0-PED模式，1-共享模式，2-隔离模式，3-混合模式  <0-错误码非 | |
| 备注 | 1.巴西CKD机型上返回1或2，其他机型仅返回2 | |

#### 3.3.33 设置PED模式

|  |  |  |
| --- | --- | --- |
| 原型 | int setPedMode(int mode) | |
| 功能 | 设置PED（PIN Entry Device）模式 | |
| 参数 | mode[in] | PED模式，1-共享模式，2-隔离模式，3-混合模式 |
| 返回值 | 0-成功  非0-错误码 | |
| 备注 | **1.巴西CKD机型上支持设置PED模式为1或2，且切换模式时会清除设备上的所有密钥**  2非巴西CKD机型不支持设置PED模式，此接口总是返回-2147483648（未知错误） | |

#### 3.3.34 获取所有PED密钥信息（巴西CKD专用）

|  |  |  |
| --- | --- | --- |
| 原型 | int getPedKeysInfo(Bundle info) | |
| 功能 | 获取所有PED（PIN Entry Device）密钥信息 | |
| 参数 | info[out] | 密钥信息数据 |
| 返回值 | 0-成功  非0-错误码 | |
| 备注 | 1.当前SDK不支持获取PED密钥信息，此接口总是返回-2147483648（未知错误）  **2.本接口为巴西CKD机型上专用接口，在非巴西CKD机型上不支持** | |

#### 3.3.35 安装共享库

|  |  |  |
| --- | --- | --- |
| 原型 | int installSharedLib(String path) | |
| 功能 | 安装共享库 | |
| 参数 | path [in] | 共享库的绝对路径，如：”/sdcard/emlib/libAE.so” |
| 返回值 | 0-成功  非0-错误码 | |
| 备注 |  | |

#### 3.3.36 删除共享库

|  |  |  |
| --- | --- | --- |
| 原型 | int deleteSharedLib(String name) | |
| 功能 | 删除共享库 | |
| 参数 | name[in] | 共享库的名称，如：” libAE.so” |
| 返回值 | 0-成功  非0-错误码 | |
| 备注 |  | |

#### 3.3.37 安装Liteso文件（TOSS专用）

|  |  |  |
| --- | --- | --- |
| 原型 | int litesoInstaller(int index, String filePath) | |
| 功能 | 安装Liteso文件 | |
| 参数 | index [in] | Liteso安装的位置，范围：0~11 |
| filePath[in] | Liteso的完整文件路径（长度不超过128字节） |
| 返回值 | 0-成功  非0-错误码 | |
| 备注 | 1.Liteso文件大小<=200KB  2.安装Liteso会覆盖之前安装在index处（如果存在）的Liteso文件  3.安装前要保证SDK有权限访问**filePath**指定的Liteso文件  **4.本接口为TOSS机型上专用接口，在非TOSS机型上不支持** | |

#### 3.3.38 运行Liteso（TOSS专用）

|  |  |  |
| --- | --- | --- |
| 原型 | int litesoRun(int index) | |
| 功能 | 运行指定index处的Liteso | |
| 参数 | index [in] | Liteso安装的位置，范围：0~11 |
| 返回值 | 0-成功  非0-错误码 | |
| 备注 | 1.运行Liteso前要确保已在**index**处安装的Liteso文件  2. 运行前会清除上次运行的Liteso及其存储空间所有数据,并重启SE芯片. 运行成功后, SE芯片会保存当前索引,直到下一次执行LitesoRun接口  3.在SE芯片的声明周期内，运行Liteso的次数不能超过10000次  **4.本接口为TOSS机型上专用接口，在非TOSS机型上不支持** | |

#### 3.3.39 获取Liteso信息（TOSS专用）

|  |  |  |
| --- | --- | --- |
| 原型 | int litesoInfo(int index, Bundle info) | |
| 功能 | 获取Liteso信息 | |
| 参数 | index [in] | Liteso安装的位置，范围：0~11 |
| Info[out] | Liteso信息，包含如下key：  name：String， Liteso名称  desc：String，Liteso应用描述  vender：String，Liteso厂商描述  version：String，Liteso版本信息 |
| 返回值 | 0-成功  非0-错误码 | |
| 备注 | **1.本接口为TOSS机型上专用接口，在非TOSS机型上不支持** | |

#### 3.3.40 获取当前运行的Liteso信息（TOSS专用）

|  |  |  |
| --- | --- | --- |
| 原型 | int litesoRunInfo(Bundle info) | |
| 功能 | 获取当前运行的Liteso信息 | |
| 参数 | info[out] | Liteso信息，包含如下key：  name，String， Liteso名称  desc，String，Liteso应用描述  vender，String，Liteso厂商描述  version，String，Liteso版本信息 |
| 返回值 | 0-成功  非0-错误码 | |
| 备注 | **1.本接口为TOSS机型上专用接口，在非TOSS机型上不支持** | |

#### 3.3.41 与Liteso进行数据交互（TOSS专用）

|  |  |  |
| --- | --- | --- |
| 原型 | int litesoCustomCmd(int cmd, byte[] dataIn, byte[] dataOut) | |
| 功能 | 与当前选择的Liteso进行数据交互 | |
| 参数 | cmd[in] | 指令码 |
| dataIn[in] | 传给Liteso的数据区，长度范围：0~1500B |
| dataOut[out] | 指令的结果数据，输出数据长度范围：0~1960B |
| 返回值 | >=0-dataOut中有效数据的长度  <0-错误码 | |
| 备注 | **1.本接口为TOSS机型上专用接口，在非TOSS机型上不支持** | |

#### 3.3.42 删除Liteso（TOSS专用）

|  |  |  |
| --- | --- | --- |
| 原型 | int litesoRemove(int index) | |
| 功能 | 删除Liteso | |
| 参数 | name[in] | Liteso安装的位置，范围：0~11 |
| 返回值 | 0-成功  非0-错误码 | |
| 备注 | **1.本接口为TOSS机型上专用接口，在非TOSS机型上不支持** | |

#### 3.3.43 获取RTC电池电压

|  |  |  |
| --- | --- | --- |
| 原型 | int getRtcBatVol(Bundle info) | |
| 功能 | 获取RTC电池电压 | |
| 参数 | info[out] | 电池电压信息，包含如下key：  vol：int，电压值，单位：mv  fromAdc: 电压值来源，0-缓存值，1-从ADC中直接读取的值，非缓存值 |
| 返回值 | 0-成功  非0-错误码 | |
| 备注 | 1.若累计时间已超时，将使能检测电路读取纽扣电池的电压，更新缓存值并在返回的数据中将fromAdc标志置为1;  2.若累计时间未超时，返回给上层的数据包中使用纽扣电池电压的缓存值，并将fromAdc标志置为0 | |

#### 3.3.44 读取验签Puk（巴西CKD专用）

|  |  |  |
| --- | --- | --- |
| 原型 | int readPuk(int pukType, Bundle info) | |
| 功能 | 读取验签Puk | |
| 参数 | pukType[in] | Puk类型，1-manufacturer，2-resaller (不支持) |
| info[out] | Puk信息，包含如下key：  pubKey：byte[]，公钥数据，长度264B，格式为：模长(4B，MSB，值为LEN)+模(LEN B)+指数(4B, MSB)  digestTime：long，生效时间，单位：ms  validDate：long，失效时间，单位：ms  owner：String，拥有者名称 |
| 返回值 | 0-成功  非0-错误码 | |
| 备注 | **1.本接口为巴西CKD机型上专用接口，在非巴西CKD机型上不支持** | |

### 3.4 卡操作模块

#### 3.4.1 卡模块接口

##### 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：  CardType. MAGNETIC.getValue() | CardType. NFC.getValue() | CardType. IC.getValue() |
| callback[in] | 检卡回调，详见[CheckCardCallbackV2](#_3.4.2_CheckCardCallbackV2_检卡回调对象) |
| timeout[in] | 超时时间，（单位为秒） 参数取值范围：1-600（秒） |
| 返回值 | 无 | |
| 备注 | 1.不区分银行卡和非银行卡  2.当参数timeout<=0时，SDK使用默认值60，timeout>120时，SDK使用默认值120  3.当cardType为复合类型时，默认检卡流程为出错不停止（接触卡除外，接触卡总是出错即停止）  **4.当检磁卡时，此接口会返回1,2,3磁道明文数据（若有），使用时请注意风险**  **5.当检接触卡时，此接口会对卡片上电，请在读卡完成后调用[cardOff()](#_3.4.1.6_卡片下电)接口对卡片下电。长时间不下电可能导致设备功耗高**  **6.当检非接触卡时，此接口会打开读卡器载波，请在读卡完成后调用[cardOff()](#_3.4.1.6_卡片下电)接口关闭载波。长时间不关闭载波可能导致设备功耗高且发热**  7.当开启sred功能后，CheckCardCallbackV2.onError()返回错误码-30001：读卡失败，参考[SRED说明](#sred说明) | |

##### 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 | 无 |
| 返回值 | 无 |
| 备注 | 检卡未返回（[CheckCardCallbackV2](#_3.4.2_CheckCardCallbackV2_检卡回调对象)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；  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 {  byte[] command; //命令  short lc; //dataIn中有效数据的长度(0~256)  byte[] dataIn; //数据体（最长256字节）  short le; //le (0~256)  } |
| recv[out] | class ApduRecvV2 {  short outLen; // outData中的有效数据的长度(0~256)  byte[] outData; //返回数据（最长256字节）  byte swa; //swa  byte swb; //swb  } |
| 返回值 | 0-成功  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** | |

##### 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；  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：  Command(4B)+LC(1B，值为len)+inData(len B)+LE(1B)  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：  outLen(2B,大端模式，值为len)+outData(len B)+SWA(1B)+SWB(1B) |
| 返回值 | 0-成功  非0-错误码 | |
| 备注 |  | |

##### 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；  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1929B |
| recvBuff [out] | 卡片的应答数据，应答数据的最大长度为2046B，recvBuff.length>=2046 |
| 返回值 | >=0- recvBuff中有效数据的长度  <0-错误码 | |
| 备注 | 1.本接口不支持mifare UltralightC、Mifare Ultralight EV系列卡透传，mifare UltralightC、Mifare Ultralight EV系列卡透传请使用[transmitApduEx()](#_3.4.1.14_透传APDU指令（扩展）)、[transmitApduExx()](#_3.4.1.17_透传APDU指令（二次扩展）)接口 | |

##### 3.4.1.6 卡片下电

|  |  |  |
| --- | --- | --- |
| 原型 | int cardOff(int cardType) | |
| 功能 | 对接触式IC卡下电或对非接触式卡关闭载波。 | |
| 参数 | cardType[in] | 卡类型（支持复合卡类型），值可以为以下两种形式种的任一种：  1. AidlConstants.CardType.IC.getValue()  2. AidlConstants.CardType.IC.getValue() | AidlConstants.CardType.NFC.getValue() |
| 返回值 | 0-卡片已经下电(接触式IC)或移走(非接触式IC卡)  非0-错误码 | |
| 备注 | 1.当cardType为复合卡类型时，若复合卡类型中的任一种卡下电失败，则接口调用失败，返回值为第一种下电失败的卡类型对应的错误码 | |

##### 3.4.1.7 判断卡片是否在位

|  |  |  |
| --- | --- | --- |
| 原型 | int getCardExistStatus(int cardType) | |
| 功能 | 判断卡片是否在位 | |
| 参数 | cardType[in] | 卡类型（非复合类型），每次只可传一种卡类型 |
| 返回值 | >=0-卡在位状态，参考：[卡片在位状态(AidlConstants.CardExistStatus)](#_7.1.36_卡片在位状态)  <0-错误码 | |
| 备注 |  | |

##### 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  非0-错误码 | |
| 备注 |  | |

###### 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  <0-错误码 | |
| 备注 |  | |

###### 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  非0-错误码 | |
| 备注 |  | |

###### 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  非0-错误码 | |
| 备注 |  | |

###### 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  非0-错误码 | |
| 备注 |  | |

###### 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  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 | |

###### 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  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 | |

###### 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  非0-错误码 | |
| 备注 |  | |

###### 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  非0-错误码 | |
| 备注 |  | |

##### 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  -1-认证失败 | |
| 备注 |  | |

###### 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  <0-错误码 | |
| 备注 |  | |

###### 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  非0-错误码 | |
| 备注 |  | |

##### 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,  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  <0-错误码 | |
| 备注 |  | |

###### 3.4.1.10.2 Mifare Plus写数据

|  |  |  |
| --- | --- | --- |
| 原型 | int mifarePlusWriteBlock(int block, 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,  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 |  | |

###### 3.4.1.10.3 Mifare Plus修改block密钥

|  |  |  |
| --- | --- | --- |
| 原型 | int mifarePlusChangeBlockKey(int block, byte[] oldKey, 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,  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  非0-错误码 | |
| 备注 |  | |

##### 3.4.1.11 SLE4442/4428

###### 3.4.1.11.1 SLE卡片认证

|  |  |  |
| --- | --- | --- |
| 原型 | int sleAuthKey(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  非0-错误码 | |
| 备注 |  | |

###### 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  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  <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  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 | 无 |
| 返回值 | >=0-剩余认证次数  <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  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  <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  <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  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  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  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  <0-错误码 | |
| 备注 |  | |

###### 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  非0-错误码 | |
| 备注 | 1.验密通过后才可写相应的区  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  <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；  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1929B |
| recvBuff [out] | 卡片的应答数据，应答数据的最大长度为2046B，recvBuff.length>=2046 |
| 返回值 | >=0- recvBuff中有效数据的长度  <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：   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：   Bit0: 1-enable rx crc, 0-disable rx crc  Bit1: 1-enable tx crc, 0-disable tx crc  Bit2: 0-enable rx parity, 1-disable rx parity  Bit3: 0-enable tx parity, 1-disable tx parity  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.  2.**transmitApdu接口中不需要传入B1，SDK默认发送0x03+sendBuff给卡片**  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  <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  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  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  <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  <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：  CardType. MAGNETIC.getValue() | CardType. NFC.getValue() | CardType. IC.getValue() |
| ctrCode[in] | 卡片激活控制码,默认值0，格式如下：  Bit0-bit1：接触卡工作电压  0：VCC\_3000mV  1：VCC\_1800mV  2：VCC\_5000mV  3：预留  Bit2：接触CPU卡及SAM 卡上电复位速率  0：SPD\_1X  1：SPD\_4X  Bit3： pps是否支持  0：不支持  1：支持  Bit4：接触CPU卡及SAM 卡协议流程  0：ICC\_SPEC  1：ICC\_EMV  Bit5: 是否选择卡片支持的第二协议  0：不选择第二协议  1：选择第二协议 |
| stopOnError[in] | 是否出错即停止，0-不停止，1-停止（仅当cardType为复合类型值时有效） |
| callback[in] | 检卡回调，详见[CheckCardCallbackV2](#_3.4.2_CheckCardCallbackV2_检卡回调对象) |
| timeout[in] | 超时时间，（单位为秒） 参数取值范围：1-600（秒） |
| 返回值 | 无 | |
| 备注 | 1.不区分银行卡和非银行卡  2.当参数timeout<=0时，SDK使用默认值60，timeout>120时，SDK使用默认值120  3.无论stopOnError为何值，接触卡总是出错即停止  **4.当检磁卡时，此接口会返回1,2,3磁道明文数据（若有），使用时请注意风险**  **5.当检接触卡时，此接口会对卡片上电，请在读卡完成后调用[cardOff()](#_3.4.1.6_卡片下电)接口对卡片下电。长时间不下电可能导致设备功耗高**  **6.当检非接触卡时，此接口会打开读卡器载波，请在读卡完成后调用[cardOff()](#_3.4.1.6_卡片下电)接口关闭载波。长时间不关闭载波可能导致设备功耗高且发热**  7.当开启sred功能后，CheckCardCallbackV2.onError()返回错误码-30001：读卡失败，参考[SRED说明](#sred说明) | |

##### 3.4.1.17 透传APDU指令（二次扩展）

|  |  |  |
| --- | --- | --- |
| 原型 | int transmitApduExx(int cardType, int ctrCode, 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；  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 |
| ctrCode[in] | 卡片数据交互控制码，说明如下：  Bit0-bit3: 设置非接CPU卡apdu帧等待时间（fwi）   |  |  |  |  | | --- | --- | --- | --- | | fwi(bit0–bit3) | 等待时间  (单位: ms) | fwi(bit0–bit3) | 等待时间  (单位: ms) | | 0x0-0x3 | 卡指定时间 | 0xA | 309 | | 0x4 | 4.832 | 0xB | 618.5 | | 0x5 | 9.664 | 0xC | 1237 | | 0x6 | 19.3 | 0xD | 2474 | | 0x7 | 38.7 | 0xE | 4948 | | 0x8 | 77.3 | Other | 卡指定时间 | | 0x9 | 154.3 |  |  |   bit4-bit5：非接cpu卡apdu 重试次数  0：不重试  1：重试1次  2：重试2次  3：预留  bit6:  0：本次开启自动获取应答（默认）  1：关闭此次自动获取应答（SCC0 SAMx） |
| sendBuff [in] | 要透传给卡片的数据，最大为1929B |
| recvBuff [out] | 卡片的应答数据，应答数据的最大长度为2046B，recvBuff.length>=2046 |
| 返回值 | >=0- recvBuff中有效数据的长度  <0-错误码 | |
| 备注 | **当卡类型为Mifare时，参数sendBuff的格式与transmitApduEx()中sendBuff的格式相同** | |

##### 3.4.1.18 透传多条APDU指令

|  |  |  |
| --- | --- | --- |
| 原型 | int transmitMultiApdus(int cardType, int ctrCode, List<String> sendList, List<String>recvList) | |
| 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；  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7条 | |
| 参数 | cardTypy[in] | 当前操作的卡片类型 |
| ctrCode[in] | 卡片数据交互控制码，说明如下：  Bit0-bit3: 设置非接CPU卡apdu帧等待时间（fwi）   |  |  |  |  | | --- | --- | --- | --- | | fwi(bit0–bit3) | 等待时间  (单位: ms) | fwi(bit0–bit3) | 等待时间  (单位: ms) | | 0x0-0x3 | 卡指定时间 | 0xA | 309 | | 0x4 | 4.832 | 0xB | 618.5 | | 0x5 | 9.664 | 0xC | 1237 | | 0x6 | 19.3 | 0xD | 2474 | | 0x7 | 38.7 | 0xE | 4948 | | 0x8 | 77.3 | Other | 卡指定时间 | | 0x9 | 154.3 |  |  |   Bit4-bit5：非接cpu卡apdu 重试次数  0：不重试  1：重试1次  2：重试2次  3：预留 |
| sendList[in] | 要透传给卡片的APDU列表(Hex list)，list中每个item代表一条APDU |
| recvList[out] | 卡片的应答数据(Hex list)，list中每个item代表一条应答数据 |
| 返回值 | >=0- 成功  <0-错误码 | |
| 备注 | **当卡类型为Mifare时，参数sendBuff的格式与transmitApduEx()中sendBuff的格式相同** | |

##### 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。  **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：  cardType：int，卡类型,同时支持NFC,IC,MAG卡检卡  encKeySystem：int，密钥体系，参见 [密钥体系常量](#密钥体系常量)，仅支持SEC\_MKSK，SECK\_DUKPT，SEC\_RSA  encKeyIndex：int，磁道数据加密密钥索引，一般传入TDK索引，参考附录：[密钥体系及密钥索引范围](#_7.6_密钥体系及密钥索引范围)  encMode：int，加密模式，参考附录：Aidl常量.[加密模式常量](#加密模式常量)，仅当encKeySystem为SEC\_MKSK，SECK\_DUKPT时有效  encIv：byte[]，初始向量，加密模式为ECB时传null，为其他加密模式时传入8或16字节向量。仅当encKeySystem为SEC\_MKSK，SECK\_DUKPT时有效  encPaddingMode：byte，当磁道数据进行DES/AES/SM4加密时，若数据长度不是8/16的倍数，则在后面补齐EncPaddingMode至长度为8/16的倍数；若磁道数据进行RSA加密，则填充模式参考 [RSA填充模式](#RSA填充模式)，且不支持 NOTHING\_PADDING模式  encMaskStart：int，表示账号前EncMaskStart位为明文，范围是0~6  encMaskEnd：int，表示账号后EncMaskEnd位为明文，范围是0~4  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‘\*’  panAppendContent：String，RSA对track2加密时前缀/后缀数据（TID）  panAppendMode：int，RSA对track2加密时前后缀模式，0-前缀式（输入数据为TID+track2），1-后缀式（输入数据为track2+TID）  ctrCode：int，卡片激活控制码,默认值0  bit0~bit1：接触卡工作电压：  0：VCC\_3000m  1：VCC\_1800mV  2：VCC\_5000mV  3：预留  bit2：接触CPU卡及SAM卡上电复位速率  0：SPD\_1X  1：SPD\_4X  bit3：是否支持PPS  0：不支持  1：支持  bit4：接触CPU卡及SAM卡协议流程  0：ICC\_SPEC  1：ICC\_EMV  stopOnError：int，是否出错即停：0-不停止，1-停止（仅当cardType为复合类型值时有效）  encKeyAlgType：int，密钥的算法类型，参见[密钥算法类型常量定义](#密钥算法类型常量定义) |
| callback[in] | 检卡回调，详见[CheckCardCallbackV2](#CheckCardCallbackV2) |
| timeout[in] | 超时时间，（单位为秒） 参数取值范围：1-600（秒） |
| 返回值 | 无 | |
| 备注 | 1.本接口不区分银行卡和非银行卡  2.本接口在TOSS机型上不支持  3.当参数timeout<=0时，SDK使用默认值60，timeout>120时，SDK使用默认值120  4.无论stopOnError为何值，接触卡总是出错即停止  **5.当检接触卡时，此接口会对卡片上电，请在读卡完成后调用[cardOff()](#_3.4.1.6_卡片下电)接口对卡片下电。长时间不下电可能导致设备功耗高**  **6.当检非接触卡时，此接口会打开读卡器载波，请在读卡完成后调用[cardOff()](#_3.4.1.6_卡片下电)接口关闭载波。长时间不关闭载波可能导致设备功耗高且发热**  7.开启或关闭sred功能时，此接口不受影响 | |

##### 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：  0-设置Felica卡轮询指令的系统码(默认为0xffff)，2字节，高位在前  1-设置非接指令交互的超时时间，单位：ms，4字节，高位在前  2-获取非接寄存器配置(TLV格式)  3-设置非接寄存器配置(TLV格式)  4-获取非接参数配置(TLV格式)  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  <0-错误码 | |
| 备注 |  | |

##### 3.4.1.21 TOSS检卡（TOSS专用）

|  |  |  |
| --- | --- | --- |
| 原型 | void checkCardForToss(Bundle bundle, 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 | bundle[in] | 检卡参数，包含如下key：  cardType：int，卡类型,同时支持NFC,IC,MAG卡检卡  ctrCode：int，卡片激活控制码,默认值0  bit0~bit1：接触卡工作电压：  0：VCC\_3000m  1：VCC\_1800mV  2：VCC\_5000mV  3：预留  bit2：接触CPU卡及SAM卡上电复位速率  0：SPD\_1X  1：SPD\_4X  bit3：是否支持PPS  0：不支持  1：支持  bit4：接触CPU卡及SAM卡协议流程  0：ICC\_SPEC  1：ICC\_EMV  code：int，vanCode值：0x01-KICC，0x02-NICE，0x03-KIS，0x04-SMATRO，0x05-KSNET  type：int，卡类型：'I'-IC，'M'-MS/Fallback，'K'-KeyIn，'B'-Barcode  maskStart：int，磁卡账号前maskStart位为明文，范围0-8  maskEnd：int，磁卡账号后maskEnd位为明文，范围0-4，默认最后一位加掩码  maskChar：char，磁卡掩码字符，默认为‘\*’，传0表示使用默认字符，传‘0’~‘9’之间的字符，将会被替换成‘\*’，其他情况下使用传入的字符。  stopOnError：int，是否出错即停止，0-不停止，1-停止（仅当cardType为复合类型值时有效） |
| callback[in] | 检卡回调，详见[CheckCardCallbackV2](#CheckCardCallbackV2) |
| timeout[in] | 超时时间，（单位为秒） 参数取值范围：1-600（秒） |
| 返回值 | 无 | |
| 备注 | 1.本接口不区分银行卡和非银行卡  2.本接口仅返回2磁道数据，不返回1、3磁道数据  3.若读取到正确的2磁道数据，检卡将返回：  （1）加密后的2磁道数据  （2）2磁道中加了掩码的账号  （3）2磁道中的服务代码  4.当2磁道无数据或解码出错且3磁道有数据时，将以3磁道的数据代替2磁道的数据在步骤3中返回  5.当参数timeout<=0时，SDK使用默认值60，timeout>120时，SDK使用默认值120  6.不论stopOnError为何值，接触卡总是出错即停止  **7.当检接触卡时，此接口会对卡片上电，请在读卡完成后调用[cardOff()](#_3.4.1.6_卡片下电)接口对卡片下电。长时间不下电可能导致设备功耗高**  **8.当检非接触卡时，此接口会打开读卡器载波，请在读卡完成后调用[cardOff()](#_3.4.1.6_卡片下电)接口关闭载波。长时间不关闭载波可能导致设备功耗高且发热**  9.当开启sred功能后，CheckCardCallbackV2.onError()返回错误码-30001：读卡失败，参考[SRED说明](#sred说明)  **10.本接口为TOSS机型上专用接口，在非TOSS机型上不支持** | |

##### 3.4.1.22 PASS模式透传APDU指令

|  |  |  |
| --- | --- | --- |
| 原型 | int smartCardExChangePASS(int cardType, in byte[] apduSend, byte[] apduRecv) | |
| 功能 | **以PASS模式透传数据的方式**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；  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] | 要透传给卡片的数据，最大为255B |
| apduRecv[out] | 应答数据单元，apduRecv.length>=260，格式为：  outLen(2B,大端模式，值为len)+outData(len B)+SWA(1B)+SWB(1B) |
| 返回值 | 0-成功  非0-错误码 | |
| 备注 |  | |

##### 3.4.1.22 PASS模式透传APDU指令

|  |  |  |
| --- | --- | --- |
| 原型 | int smartCardExChangePASS(int cardType, in byte[] apduSend, byte[] apduRecv) | |
| 功能 | **以PASS模式透传数据的方式**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；  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] | 要透传给卡片的数据，最大为255B |
| apduRecv[out] | 应答数据单元，apduRecv.length>=260，格式为：  outLen(2B,大端模式，值为len)+outData(len B)+SWA(1B)+SWB(1B) |
| 返回值 | 0-成功  非0-错误码 | |
| 备注 |  | |

##### 3.4.1.23 PASS模式APDU指令交互，返回数据不包含outLen字段

|  |  |  |
| --- | --- | --- |
| 原型 | int smartCardExChangePASSNoLength(int cardType, byte[] apduSend, byte[] apduRecv) | |
| 功能 | **以PASS模式透传数据的方式**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；  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] | 要透传给卡片的数据，最大为255B |
| apduRecv[out] | 应答数据单元，apduRecv.length>=260，格式为：  outData+SWA(1B)+SWB(1B) |
| 返回值 | 0-成功  非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：  cardType：int，卡片类型  TRACK1：Strin，磁道1数据  TRACK2：String，磁道2数据  TRACK3：String，磁道3数据  track2Raw:byte[] ,track2 data(renturn by  checkCardForToss())  pan：String，PAN数据（**checkCardEnc()**返回）  name：String，持卡人姓名（**checkCardEnc()**返回）  expire：String, 卡片有效期（**checkCardEnc()**返回）  servicecode: String, 卡片服务码 (**checkCardEnc()**返回)  appendedPanEnc：String，RSA加密后的TID+track2或track2+TID数据，Hex格式（**checkCardEnc()**返回）  appendedPanEncBytes：byte[]，RSA加密后的TID+track2或track2+TID数据（**checkCardEnc()**返回）  track1ErrorCode：int，磁道1错误码  track2ErrorCode：int，磁道2错误码  track3ErrorCode：int，磁道3错误码  磁道错误码取值如下：  -1：磁道无数据  -2：磁道奇偶校验错  -3：磁道LRC校验错 |
| 返回值 | 无 | |
| 备注 |  | |

##### 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 | 无 | |
| 备注 |  | |

##### 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 | 无 | |
| 备注 |  | |

##### 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 | 无 | |
| 备注 |  | |

##### 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：  cardType：int，检到的卡类型  atr：String，卡片的ATR |
| 返回值 | 无 | |
| 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(Bundle info) | |
| 功能 | 检到非接卡 | |
| 参数 | info[in] | 包含如下数据：  cardType：int，检到的卡类型  uuid：String，卡片的UUID  ats：String，卡片的ATS  cardCategory：int，卡片类别（‘A’或‘B’）  atqa；byte[]，卡片的ATQA |
| 返回值 | 无 | |
| 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：  cardType：int，当前出错的卡类型  code：int，错误码  message：String，错误信息 |
| 返回值 | 无 | |
| 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（TOSS不支持）

#### 3.5.1 密码键盘模块接口

##### 3.5.1.1 初始化密码键盘

|  |  |  |
| --- | --- | --- |
| 原型 | String 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，参见[PinPadConfigV2](#PinPadConfigV2) |
| listerner[in] | 输PIN回调，参见[PinPadListenerV2](#PinPadListenerV2) |
| 返回值 | 顺序或乱序数字键值 | |
| 备注 | 1.有实体键盘的设备（如P2\_SmartPad、P3K）上仅支持实体PIN键盘，不支持虚拟PIN键盘和视障模式PIN键盘，调用本接口启动视障模式键盘失败 | |

##### 3.5.1.2 导入密码键盘布局数据

|  |  |  |
| --- | --- | --- |
| 原型 | void importPinPadData(PinPadDataV2 data) | |
| 功能 | 导入密码键盘UI布局数据 | |
| 参数 | data[in] | 密码键盘UI布局数据，参见[PinPadDataV2](#PinPadDataV2) |
| 返回值 | 无 | |
| 备注 | 1.客户端APP仅当使用自定义键盘时(调用initPinPad()时，config.pinpadType=1)，才需要调用此接口  2.自定义键盘的UI布局、按键映射均由客户端APP自己实现  3.有实体键盘的设备（如P2\_SmartPad、P3K）上仅支持实体PIN键盘，不支持虚拟PIN键盘和视障模式PIN键盘，调用本接口导入虚拟键盘布局数据无效果 | |

##### 3.5.1.3 取消输PIN

|  |  |
| --- | --- |
| 原型 | void cancelInputPin() |
| 功能 | 取消输PIN |
| 参数 | 无 |
| 返回值 | 无 |
| 备注 |  |

##### 3.5.1.4 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](#PinPadTextConfigV2) |
| 返回值 | 无 | |
| 备注 | 1.有实体键盘的设备（如P2\_SmartPad、P3K）上仅支持实体PIN键盘，不支持虚拟PIN键盘和视障模式PIN键盘，调用本接口设置键盘显示的文字无效果 | |

##### 3.5.1.5 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：  normal：int，普通模式（0-关闭，1-开启。本模式和其他模式互斥）  longPressToClear：int，长按Clear按钮清除输入的内容（0-关闭，1-开启）  silent：int，输PIN时静音（0-关闭，1-开启）  greenLed：int，输PIN时LED绿灯亮（0-关闭，1-开启）  monitorClearKey：int，监视是否按下清除按钮（0-关闭，1-开启）  cancelToClear: int，按取消键清除输PIN（0-关闭，1-开启）  visualImpairment：int，视障模式（0-关闭，1-开启.本模式和其他模式互斥）  longTimeoutTime：输PIN最大超时时间为10min（类型：int，0-关闭，1-开启） |
| 返回值 | 0-成功  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）  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  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**  4.有实体键盘的设备（如P2\_SmartPad、P3K）上仅支持实体PIN键盘，不支持虚拟PIN键盘和视障模式PIN键盘，调用本接口设置**视障模式**无效果 | |

##### 3.5.1.6 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 | bundle [out] | 密码键盘模式参数，包含如下key：  normal：int，普通模式（0-关闭，1-开启）  longPressToClear：int，长按Clear按钮清除输入的内容（0-关闭，1-开启）  silent：int，输PIN时静音（0-关闭，1-开启）  greenLed：int，输PIN时LED绿灯亮（0-关闭，1-开启）  monitorClearKey：int，监视是否按下清除按钮（0-关闭，1-开启）  cancelToClear: int，按取消键清除输PIN（0-关闭，1-开启）  longTimeoutTime：输PIN最大超时时间为10min（类型：int，0-关闭，1-开启） |
| 返回值 | 0-成功  非0-错误码 | |
| 备注 |  | |

##### 3.5.1.7 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：  pinPadType: int，密码键盘类型，范围：0-预置普通键盘(默认值) ，1-自定义普通键盘，2-预置盲人模式键盘，3-预置rnib认证盲人模式键盘，4-预置rnib认证普通键盘，5-自定义盲人模式键盘  pinType：int，PIN类型标识，范围：0-联机PIN，1-脱机PIN  isOrderNumKey：int，是否顺序键盘，范围：0-乱序键盘(默认值)，1-顺序键盘  pan：byte[]，PAN数据，ASCII格式转换成的byte 例如“123456”.getBytes(“US-ASCII”)，长度12~19位  pinKeyIndex：int，PIK(PIN key)索引  minInput：int，最小输入位数，默认0  maxInput：int，最大输入位数，默认6  inputStep：int，PIN步长，默认1  timeout：int，超时时间，单位：ms，默认60000（1min）  isSupportbypass：int，是否支持bypass PIN，范围：0-不支持，1-支持(默认值)  pinblockFormat：int，PinBlock格式，默认0，参考 [PinBlock格式](#PinBlock格式)  algorithmType：int，加密Pin的算法类型，0-3DES(返回8字节PinBlock)，1-SM4(返回16字节PinBlock)，2-AES(返回16字节PinBlock)  keySystem：int，密钥体系，范围：0-SEC\_MKSK(默认值)，1-SEC\_DUKPT  diversify：byte[]，分散因子 ，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](#PinPadListenerV2) |
| 返回值 | 无 | |
| 备注 | 1.当前不支持pinPadType为3、4类型  2.有实体键盘的设备（如P2\_SmartPad、P3K）上仅支持实体PIN键盘，不支持虚拟PIN键盘和视障模式PIN键盘，调用本接口启动视障模式键盘失败 | |

##### 3.5.1.8 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  <0-错误码 | |
| 备注 |  | |

##### 3.5.1.9 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 | 无 |
| 返回值 | >=0-当前生效的模式，范围1-5  <0-错误码 |
| 备注 |  |

##### 3.5.1.10 设置视障模式参数

|  |  |  |
| --- | --- | --- |
| 原型 | int setVisualImpairmentModeParam(Bundle param) | |
| 功能 | 设置视障模式参数 | |
| 参数 | bundle [in] | 密码键盘模式参数，包含如下key：  timeoutGap1：int，手指触碰屏幕时间，取值范围0~100，单位：100ms，默认值：10  timeoutGap2：int，两次点击屏幕的间隔时间，取值范围0~100，单位：100ms，默认值：10  ttsLanguage：int，语音播报的语言（0-跟随系统（默认值），1-英语，2-波兰语，3-法语，4-葡萄牙语(巴西)，5-中文(中国)，6-西班牙语)  rnibSelectMode：int，PIN数字确认模式（0-双击确认（默认值），1-长按确认）  rnibHoldTime：int，长按确认模式下长按时间，单位：100ms(范围：(0,100]，默认值：30) |
| 返回值 | 0-成功  非0-错误码 | |
| 备注 | 1.如果key缺省，SDK使用key的默认值设置，例：key **rnibHoldTime**缺省，SDK使用30（**rnibHoldTime**的默认值）设置  2.有实体键盘的设备（如P2\_SmartPad、P3K）上仅支持实体PIN键盘，不支持虚拟PIN键盘和视障模式PIN键盘，调用本接口设置视障模式参数无效果 | |

##### 3.5.1.11 获取视障模式参数

|  |  |  |
| --- | --- | --- |
| 原型 | int getVisualImpairmentModeParam(Bundle param) | |
| 功能 | 获取视障模式参数 | |
| 参数 | bundle [out] | 密码键盘模式参数，包含如下key：  timeoutGap1：int，手指触碰屏幕时间，取值范围0~100，单位：100ms  timeoutGap2：int，两次点击屏幕的间隔时间，取值范围0~100，单位：100ms  ttsLanguage：int，语音播报的语言(类型：int，0-跟随系统（默认值），1-英语，2-波兰语，3-法语，4-葡萄牙语(巴西)，5-中文(中国))，6-西班牙语)  rnibSelectMode: PIN数字确认模式（类型：int，0-双击确认（默认值），1-长按确认）  rnibHoldTime：长按确认时模式下长按时间，单位：100ms(类型：int，范围：(0,100]，默认值：30) |
| 返回值 | 0-成功  非0-错误码 | |
| 备注 |  | |

##### 3.5.1.12 启动输PIN

|  |  |  |
| --- | --- | --- |
| 原型 | int startInputPin(Bundle param, PinPadListenerV2 listener) | |
| 功能 | 启动输PIN，输PIN完成后不返回PinBlock | |
| 参数 | param [in] | 密码键盘配置，包含key：  pinPadType：int，密码键盘类型，范围：0-预置普通键盘(默认值) ，1-自定义普通键盘，2-预置盲人模式键盘，3-预置rnib认证盲人模式键盘，4-预置rnib认证普通键盘，5-自定义盲人模式键盘  pinType：int，PIN类型标识，范围：0-联机PIN，1-脱机PIN  isOrderNumKey：int，是否顺序键盘，范围：0-乱序键盘(默认值)，1-顺序键盘  minInput：int，最小输入位数，默认0  maxInput：int，最大输入位数，默认6  inputStep：int，PIN步长，默认1  expLen：String，允许输入的PIN数字个数，以","分割，例如"0,4,6"表示允许输入0个/4个/6个数字，此字段与minInput/maxInput/inputStep互斥，若都存在，则优先使用expLen  isSupportbypass：int，是否支持bypass PIN，范围：0-不支持，1-支持(默认值)  timeout：int，超时时间，单位：ms，默认60000 |
| listener[in] | 参见[PinPadListenerV2](#PinPadListenerV2) |
| 返回值 | 0-成功  非0-错误码 | |
| 备注 | 1.本接口输PIN完成后不返回PinBlock，查询PinBlock请调用getPinBlock()接口  2.当前不支持pinPadType为3、4类型  3.有实体键盘的设备（如P2\_SmartPad、P3K）上仅支持实体PIN键盘，不支持虚拟PIN键盘和视障模式PIN键盘，调用本接口启动视障模式键盘失败 | |

##### 3.5.1.13 获取PinBlock

|  |  |  |
| --- | --- | --- |
| 原型 | int getPinBlock(Bundle param, byte[] dataOut) | |
| 功能 | 获取PinBlock | |
| 参数 | param [in] | 密码键盘配置，包含key：  keySystem：int，密钥体系，范围： 0-SEC\_MKSK(默认值), 1-SEC\_DUKPT  pinKeyIndex：int，PIK(PIN key)索引  algorithmType：int，加密Pin的算法类型，范围：0-3DES(返回8字节PinBlock)，1-SM4(返回16字节PinBlock)，2-AES(返回16字节PinBlock)  pinblockFormat：int，PinBlock格式，默认0  pan：byte[]，PAN数据，ASCII格式转换成的byte 例如 “123456”.getBytes("US-ASCII") |
| dataOut[out] | Buffer，存放返回的PinBlock数据 |
| 返回值 | >=0-dataOut中有效数据的长度  <0-错误码 | |
| 备注 | **1.调用此接口前必须先调用startInputPin()接口，且调用本接口的App和调用startInputPin()接口的App必须是同一个** | |

##### 3.5.1.14 验证脱机PIN（巴西CKD专用）

|  |  |  |
| --- | --- | --- |
| 原型 | int offlinePinVerify(Bundle paramIn, Bundle paramOut) | |
| 功能 | 验证脱机PIN | |
| 参数 | paramIn [in] | 脱机PIN参数，包含key：  offlineType：int，脱机类型，范围：0-脱机明文，1-脱机密文  modulus：byte[]，RSA密钥模  exponent：byte[]，RSA密钥指数  random：byte[]，随机数 |
| paramount[out] | 出参数据，包含key：  sw1：int，SW1  sw2：int，SW2 |
| 返回值 | 0-成功  非0-错误码 | |
| 备注 | **1.本接口为巴西CKD机型上专用接口，在非巴西CKD机型上不支持** | |

##### 3.5.1.15 导入密码键盘布局数据（扩展）

|  |  |  |
| --- | --- | --- |
| 原型 | void importPinPadDataEx(PinPadDataV2Ex data) | |
| 功能 | 导入密码键盘UI布局数据 | |
| 参数 | data[in] | 密码键盘UI布局数据，参见[PinPadDataV2Ex](#PinPadDataV2Ex) |
| 返回值 | 无 | |
| 备注 | 1.客户端APP仅当使用自定义键盘时(调用initPinPad()时，config.pinpadType=1)，才需要调用此接口  2.自定义键盘的UI布局、按键映射均由客户端APP自己实现  3. PinPadDataV2Ex相对于PinPadDataV2增加了enterX、enterY、enterH、enterW字段用于自定义Enter键的位置，增加了clearX、clearY、clearH、clearW用于自定义Clear键的位置  4.有实体键盘的设备（如P2\_SmartPad、P3K）上仅支持实体PIN键盘，不支持虚拟PIN键盘和视障模式PIN键盘，调用本接口导入键盘布局数据无效果 | |

#### 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 | 无 | |
| 备注 |  | |

##### 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.  （1）type = 0(联机PIN)：  （a）pinBlock=null，bypass模式（用户未输pin并直接按下确认键）  （b）pinBlock！=null，PinBlock数据，长度为8B（3DES）或16B（SM4/AES）  （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 | 无 | |
| 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 | 无 |
| 返回值 | 无 |
| 备注 |  |

##### 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](#ErrorCode) |
| 返回值 | 无 | |
| 备注 |  | |

##### 3.5.2.5 视障模式手指滑过按键

|  |  |  |
| --- | --- | --- |
| 原型 | void onHover (int event, byte[] data) | |
| 功能 | 视障模式手指滑过按键回调 | |
| 参数 | event[in] | 事件类型：  4-视障模式-触碰到ENTER键  5-视障模式-触碰到CLEAR键  6-视障模式-触碰到CANCEL键  7-视障模式-触碰到无效键值区域（已废弃）  8-视障模式-当前输入PIN长度已达上限  9-视障模式-触碰到数字键，需要播放beep  10-视障模式-手指触碰上方无效区域  11-视障模式-手指触碰下方无效区域（暂不支持）  12-视障模式-手指触碰左侧无效区域  13-视障模式-手指触碰右侧无效区域 |
| data[in] | 与event相关的数据：  （1）event=5或6，data[0]为已输入的PIN长度  （2）event !=5和6，data值为null，无效值 |
| 返回值 | 无 | |
| 备注 |  | |

### 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](#密钥类型常量定义) |
| keyValue [in] | 密钥数据 |
| checkValue[in] | 密钥校验值，参考：[KCV参数说明](#KCV参数说明) |
| 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](#密钥算法类型常量定义) |
| keyIndex[in] | 保存的索引，范围：参考附录：[密钥体系及密钥索引范围.SEC\_MKSK](#密钥体系及密钥索引范围_SEC_MKSK) |
| 返回值 | 0-成功  非0-错误码 | |
| 备注 | 1.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  2.本接口保存的密钥永久性存储，设备关机/重启、APP重启后密钥不丢失  3.客户端App可调用[deleteKey()](#删除密钥)、[deleteKeyEx()](#删除密钥_扩展)接口删除已保存的密钥 **4.本接口在TOSS机型上不支持** | |

#### 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](#密钥类型常量定义)，不能密文保存KEY\_TYPE\_KEK类型 |
| keyValue[in] | 密钥数据 |
| checkValue[in] | 密钥校验值，参考：[KCV参数说明](#KCV参数说明) |
| 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](#密钥算法类型常量定义) |
| keyIndex[in] | 保存的索引，范围：参考附录：[密钥体系及密钥索引范围.SEC\_MKSK](#密钥体系及密钥索引范围_SEC_MKSK) |
| 返回值 | 0-成功  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  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  2.本接口保存的密钥永久性存储，设备关机/重启、APP重启后密钥不丢失  3.客户端App可调用[deleteKey()](#删除密钥)、[deleteKeyEx()](#删除密钥_扩展)接口删除已保存的密钥  **4.本接口在TOSS机型上不支持** | |

#### 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：参考附录：[密钥体系及密钥索引范围.SEC\_MKSK](#密钥体系及密钥索引范围_SEC_MKSK) |
| macType[in] | mac加密算法，参考附录：Aidl常量.[MAC算法类型](#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-成功  <0-错误码，详见[AidlErrorCode](#_PinPad-PinPad错误定义) | |
| 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];  **2.本接口在TOSS机型上不支持** | |

#### 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：参考附录：[密钥体系及密钥索引范围.SEC\_MKSK](#密钥体系及密钥索引范围_SEC_MKSK) |
| 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](#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  非0-错误码 | |
| 备注 | 无 | |
| 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  2.dataOut的长度应和dataIn的长度相等  例如：byte[] dataIn = new byte[16];  那么dataOut的长度也应该为16字节  **3.本接口在TOSS机型上不支持** | |

#### 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：参考附录：[密钥体系及密钥索引范围.SEC\_MKSK](#密钥体系及密钥索引范围_SEC_MKSK) |
| 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](#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  非0-错误码 | |
| 备注 | 无 | |
| 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  2.dataOut的长度应和dataIn的长度相等  例如：byte[] dataIn = new byte[16];  那么dataOut的长度也应该为16字节  **3.本接口在TOSS机型上不支持** | |

#### 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：  BDK-基础分散密钥:对应Aidl常量.[密钥类型常量](#密钥类型常量定义)KEY\_TYPE\_DUPKT\_BDK  IPEK-初始PIN加密密钥：对应Aidl常量.[密钥类型常量](#密钥类型常量定义)KEY\_TYPE\_DUPKT\_IPEK |
| keyValue[in] | 密钥数据 |
| checkValue[in] | 密钥校验值（当注入BDK传空），参考：[KCV参数说明](#KCV参数说明) |
| ksn[in] | KSN |
| encryptType[in] | 密钥算法 |
| keyIndex[in] | 保存的索引，范围：参考附录：[密钥体系及密钥索引范围.SEC\_DUKPT.3DES](#密钥体系及密钥索引范围_SEC_DUKPT_3DES) |
| 返回值 | 0-成功  <0-错误码 | |
| 备注 | 此接口仅可注入DES类型密钥，且密钥长度须等于16 | |
| 示例 | 1.本接口保存的密钥永久性存储，设备关机/重启、APP重启后密钥不丢失  2.客户端App可调用[deleteKey()](#删除密钥)、[deleteKeyEx()](#删除密钥_扩展)接口删除已保存的密钥  **3.本接口在TOSS机型上不支持** | |

#### 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：参考附录：[密钥体系及密钥索引范围.SEC\_DUKPT](#密钥体系及密钥索引范围_SEC_DUKPT) |
| macType [in] | mac算法，参考附录：Aidl常量.[MAC算法类型](#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  <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];  2. keyIndex范围为1100-1199(Dukpt扩展密钥)时，dataIn长度<=1016  **3.本接口在TOSS机型上不支持** | |

#### 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：参考附录：[密钥体系及密钥索引范围.SEC\_DUKPT](#密钥体系及密钥索引范围_SEC_DUKPT) |
| 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](#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  <0-错误码 | |
| 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  2.dataOut的长度应和dataIn的长度相等  例如：byte[] dataIn = new byte[16];  那么dataOut的长度也应该为16字节  3. keyIndex范围为1100-1199(Dukpt扩展密钥)时，dataIn长度<=1024  **4.本接口在TOSS机型上不支持** | |

#### 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：参考附录：[密钥体系及密钥索引范围.SEC\_DUKPT](#密钥体系及密钥索引范围_SEC_DUKPT) |
| 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](#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  <0-错误码 | |
| 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  2.dataOut的长度应和dataIn的长度相等  例如：byte[] dataIn = new byte[16];  那么dataOut的长度也应该为16字节  3. keyIndex范围为1100-1199(Dukpt扩展密钥)时，dataIn长度<=1024  **4.本接口在TOSS机型上不支持** | |

#### 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：参考附录：[密钥体系及密钥索引范围.SEC\_DUKPT](#密钥体系及密钥索引范围_SEC_DUKPT) |
| 返回值 | 0-成功  <0-错误码 | |
| 备注 | **1.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**  **2.本接口在TOSS机型上不支持** | |

#### 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：参考附录：[密钥体系及密钥索引范围.SEC\_DUKPT](#密钥体系及密钥索引范围_SEC_DUKPT) |
| 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  <0-错误码 | |
| 备注 | 1.用于Dukpt密钥体系密钥  **2.本接口在TOSS机型上不支持** | |

#### 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](#密钥体系常量) |
| keyIndex[in] | 密钥索引，值范围如下：  keySystem 为 SEC\_DUKPT 时，范围：参考附录：[密钥体系及密钥索引范围.SEC\_DUKPT](#密钥体系及密钥索引范围_SEC_DUKPT)  keySystem 为 SEC\_MKSK 时，范围：参考附录：[密钥体系及密钥索引范围.SEC\_MKSK](#密钥体系及密钥索引范围_SEC_MKSK) |
| dataOut[out] | 固定为4个字节 |
| 返回值 | 0-成功  <0-错误码 | |
| 备注 | 1.通过注入BDK时设定的密钥索引得到的是BDK对应IPEK的CheckValue  **2.本接口在TOSS机型上不支持** | |

#### 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  非0-错误码 | |
| 备注 | 1.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）  dataOut 固定传进去4个字节的byte[] 如: byte[] dataOut = new byte[4];  **2.本接口在TOSS机型上不支持** | |

#### 3.6.14 生成RSA公私钥对（废弃）

|  |  |  |
| --- | --- | --- |
| 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  非0-错误码 | |
| 备注 | 1.本接口保存的密钥永久性存储，设备关机/重启、APP重启后密钥不丢失  2.客户端App可调用[removeRSAKey()](#移除RSA密钥_废弃)接口删除已保存的密钥  **3.本接口废弃，生成RSA密钥请参考generateRSAKeypair()**  **4.本接口在TOSS机型上不支持** | |

#### 3.6.15 获取RSA公钥（废弃）

|  |  |  |
| --- | --- | --- |
| 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  <0-错误码 | |
| 备注 | **1.本接口废弃**  **2.本接口在TOSS机型上不支持** | |

#### 3.6.16 RSA算法加密数据（废弃）

|  |  |  |
| --- | --- | --- |
| 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](#RSATransformation) |
| 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  <0 – 错误码 | |
| 备注 | 1.当transformation为RSA/None/NoPadding或RSA/ECB/NoPadding时，dataIn长度要等于模长，dataOut长度要大于等于模长，例如keySize为2048时，moduleLen为256B，此时dataIn的长度要等于256B，dataOut长度大于等于256B  2.transformation RSA/ECB/OAEPWithSHA-512AndMGF1Padding仅适用于2048长度的RSA密钥  **3.本接口废弃**  **4.本接口在TOSS机型上不支持** | |

#### 3.6.17 RSA算法解密数据（废弃）

|  |  |  |
| --- | --- | --- |
| 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](#RSATransformation) |
| 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  <0 – 错误码 | |
| 备注 | 1.当transformation为RSA/None/NoPadding或RSA/ECB/NoPadding时，dataIn长度等于模长，dataOut长度要大于等于模长，例如keySize为2048时，moduleLen为256B，此时dataIn的长度等于256B，dataOut长度大于等于256B  2.transformation RSA/ECB/OAEPWithSHA-512AndMGF1Padding仅适用于2048长度的RSA密钥  **3.本接口废弃**  **4.本接口在TOSS机型上不支持** | |

#### 3.6.18 移除RSA密钥（废弃）

|  |  |  |
| --- | --- | --- |
| 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  非0 – 错误码 | |
| 备注 | **1.本接口废弃**  **2.本接口在TOSS机型上不支持** | |

#### 3.6.19 保存数字证书（废弃）

|  |  |  |
| --- | --- | --- |
| 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  非0 – 错误码 | |
| 备注 | 1.本接口保存的证书数据永久性存储，设备关机/重启、APP重启后密钥不丢失  **2.本接口废弃**  **3.本接口在TOSS机型上不支持** | |

#### 3.6.20 获取数字证书（废弃）

|  |  |  |
| --- | --- | --- |
| 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  <0 – 错误码 | |
| 备注 | **1.本接口废弃**  **2.本接口在TOSS机型上不支持** | |

#### 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  <0 – 错误码 | |
| 备注 | **1.本接口在TOSS机型上不支持** | |

#### 3.6.22 RSA算法签名数据（废弃）

|  |  |  |
| --- | --- | --- |
| 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](#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  <0 – 错误码 | |
| 备注 | 1.当签名算法为NONEwithRSA时，私钥加密时默认使用的padding方式为PKCS1Padding，dataIn的长度不超过moduleLen-11字节，dataOut的长度大于等于模长。例如keySize为2048时，moduleLen为256B，此时dataIn的长度不超过256-11=245B，dataOut的长度大于等于256B  **2.本接口废弃**  **3.本接口在TOSS机型上不支持** | |

#### 3.6.23 RSA算法验签（废弃）

|  |  |  |
| --- | --- | --- |
| 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](#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 | 1.当签名算法为NONEwithRSA时，私钥加密时默认使用的padding方式为PKCS1Padding，srcData的长度不超过moduleLen-11字节，signature的长度等于模长。例如keySize为2048时，moduleLen为256B，此时srcData的长度不超过256-11=245B，signature的长度等于256B  **2.本接口废弃**  **3.本接口在TOSS机型上不支持** | |

#### 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](#密钥类型常量定义) |
| keyValue [in] | 密钥数据 |
| checkValue[in] | 密钥校验值，参考：[KCV参数说明](#KCV参数说明) |
| keyAlgType[in] | 密钥的算法类型，参考附录：Aidl常量类.[密钥算法类型](#密钥算法类型常量定义) |
| keyIndex[in] | 密钥索引，范围：参考附录：[密钥体系及密钥索引范围.SEC\_MKSK](#密钥体系及密钥索引范围_SEC_MKSK) |
| 返回值 | 0 – 成功  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  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.本接口保存的密钥永久性存储，设备关机/重启、APP重启后密钥不丢失  4.客户端App可调用[deleteKey()](#删除密钥)、[deleteKeyEx()](#删除密钥_扩展)接口删除已保存的密钥  **5.本接口在TOSS机型上不支持** | |

#### 3.6.25 注入密文密钥

|  |  |  |
| --- | --- | --- |
| 原型 | int injectCiphertextKey(String targetPkgName, int keyType, byte[] keyValue, 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](#密钥类型常量定义)，不能密文保存KEY\_TYPE\_KEK类型 |
| keyValue[in] | 密钥数据 |
| checkValue[in] | 密钥校验值，参考：[KCV参数说明](#KCV参数说明) |
| encryptIndex[in] | 解密密钥的索引，注意，这里是TMK的索引，范围：参考附录：[密钥体系及密钥索引范围.SEC\_MKSK](#密钥体系及密钥索引范围_SEC_MKSK) |
| keyAlgType[in] | 密钥的算法类型，参考附录：Aidl常量类.[密钥算法类型](#密钥算法类型常量定义) |
| keyIndex[in] | 保存的索引，范围：参考附录：[密钥体系及密钥索引范围.SEC\_MKSK](#密钥体系及密钥索引范围_SEC_MKSK) |
| 返回值 | 0 – 成功  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  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  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  4. 调用injectCiphertextKey()接口的前置条件是调用injectPlaintextKey()接口先保存dependKey在targetAPP包名下  5.本接口保存的密钥永久性存储，设备关机/重启、APP重启后密钥不丢失  6.客户端App可调用[deleteKey()](#删除密钥)、[deleteKeyEx()](#删除密钥_扩展)接口删除已保存的密钥  **7.本接口在TOSS机型上不支持** | |

#### 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](#Dukpt密钥选择) |
| keyIndex[in] | 密钥索引，范围：参考附录：[密钥体系及密钥索引范围.SEC\_DUKPT](#密钥体系及密钥索引范围_SEC_DUKPT) |
| 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](#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  <0-失败 | |
| 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  2.dataOut的长度应和dataIn的长度相等  例如：byte[] dataIn = new byte[16];  那么dataOut的长度也应该为16字节  3. keyIndex范围为1100-1199(Dukpt扩展密钥)时，dataIn长度<=1024  **4.本接口在TOSS机型上不支持** | |

#### 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](#Dukpt密钥选择) |
| keyIndex[in] | 密钥索引，范围：参考附录：[密钥体系及密钥索引范围.SEC\_DUKPT](#密钥体系及密钥索引范围_SEC_DUKPT) |
| 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](#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  <0-失败 | |
| 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  2.dataOut的长度应和dataIn的长度相等  例如：byte[] dataIn = new byte[16];  那么dataOut的长度也应该为16字节  3. keyIndex范围为1100-1199(Dukpt扩展密钥)时，dataIn长度<=1024  **4.本接口在TOSS机型上不支持** | |

#### 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](#Dukpt密钥选择) |
| keyIndex [in] | 密钥索引，范围：参考附录：[密钥体系及密钥索引范围.SEC\_DUKPT](#密钥体系及密钥索引范围_SEC_DUKPT) |
| 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  <0-失败 | |
| 备注 | 1. keyIndex范围为1100-1199(Dukpt扩展密钥)时，dataIn长度<=1016  **2.本接口在TOSS机型上不支持** | |

#### 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](#Dukpt密钥选择) |
| keyIndex [in] | 密钥索引，范围：参考附录：[密钥体系及密钥索引范围.SEC\_DUKPT](#密钥体系及密钥索引范围_SEC_DUKPT) |
| 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  非0-失败 | |
| 备注 | 1. keyIndex范围为1100-1199(Dukpt扩展密钥)时，dataIn长度<=1016  **2.本接口在TOSS机型上不支持** | |

#### 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：参考附录：[密钥体系及密钥索引范围.SEC\_MKSK](#密钥体系及密钥索引范围_SEC_MKSK) |
| keyIndex[in] | 密钥索引，范围：参考附录：[密钥体系及密钥索引范围.SEC\_MKSK](#密钥体系及密钥索引范围_SEC_MKSK)、[密钥体系及密钥索引范围.SEC\_DUKPT](#密钥体系及密钥索引范围_SEC_DUKPT) |
| 返回值 | 0-成功  <0-错误码 | |
| 备注 | 1.保存TR31密钥前必须先保存KBPK  2.目前仅支持保存TR31 dukpt-IPEK，不支持保存TR31 dukpt-BDK  3.本接口保存的密钥永久性存储，设备关机/重启、APP重启后密钥不丢失  4.客户端App可调用[deleteKey()](#删除密钥)、[deleteKeyEx()](#删除密钥_扩展)接口删除已保存的密钥  **5.本接口在TOSS机型上不支持** | |

#### 3.6.31 保存密文密钥-解密密钥为RSA密钥（废弃）

|  |  |  |
| --- | --- | --- |
| 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](#密钥类型常量定义) |
| 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](#密钥算法类型常量定义) |
| keyIndex [in] | 保存的索引，范围：参考附录：[密钥体系及密钥索引范围.SEC\_MKSK](#密钥体系及密钥索引范围_SEC_MKSK) |
| encryptIndexRSA [in] | 解密密钥的索引，注意，这里是RSA私钥的索引，范围0-19 |
| transformation [in] | 算法/工作模式/填充模式, 参考附录：Aidl常量类.[RSA transformation](#RSATransformation) ， 仅支持RSA\_TRANSFORMATION\_4、RSA\_TRANSFORMATION\_5、RSA\_TRANSFORMATION\_6、RSA\_TRANSFORMATION\_7； |
| 返回值 | 0-成功  <0-错误码 | |
| 备注 | 1.本接口保存的密钥永久性存储，设备关机/重启、APP重启后密钥不丢失  2.客户端App可调用[deleteKey()](#删除密钥)、[deleteKeyEx()](#删除密钥_扩展)接口删除已保存的密钥  **3.本接口已废弃**  **4.本接口在TOSS机型上不支持** | |

#### 3.6.32 保存RSA密钥（废弃）

|  |  |  |
| --- | --- | --- |
| 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,  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  <0-错误码 | |
| 备注 | 1.本接口保存的密钥永久性存储，设备关机/重启、APP重启后密钥不丢失  2.客户端App可调用[removeRSAKey()](#移除RSA密钥_废弃)接口删除已保存的密钥  **3.本接口废弃**  **4.本接口在TOSS机型上不支持** | |

#### 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](#密钥体系常量) |
| keyIndex [in] | 密钥索引，值范围如下：  keySystem为SEC\_MKSK时，范围：参考附录：[密钥体系及密钥索引范围.SEC\_MKSK](#密钥体系及密钥索引范围_SEC_MKSK)  keySystem为SEC\_DUKPT 时，范围：参考附录：[密钥体系及密钥索引范围.SEC\_DUKPT](#密钥体系及密钥索引范围_SEC_DUKPT)  keySystem为SEC\_RSA\_KEY时，范围：参考附录：[密钥体系及密钥索引范围.SEC\_RSA\_KEY](#密钥体系及密钥索引范围_SEC_RSA_KEY) |
| 返回值 | 0-成功  <0-错误码 | |
| 备注 | **1.本接口在TOSS机型上不支持** | |

#### 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](#DukptKeyType常量定义) |
| keyType[in] | 密钥类型。包含：  BDK-基础分散密钥:对应Aidl常量.[密钥类型常量](#密钥类型常量定义)KEY\_TYPE\_DUPKT\_BDK  IPEK-初始PIN加密密钥：对应Aidl常量.[密钥类型常量](#密钥类型常量定义)KEY\_TYPE\_DUPKT\_IPEK |
| keyValue[in] | 密钥数据 |
| checkValue[in] | 密钥校验值，参考：[KCV参数说明](#KCV参数说明) |
| 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：参考附录：[密钥体系及密钥索引范围.SEC\_DUKPT.AES](#密钥体系及密钥索引范围_SEC_DUKPT_AES) |
| 返回值 | 0-成功  <0-失败 | |
| 备注 | 1.本接口保存的密钥永久性存储，设备关机/重启、APP重启后密钥不丢失  2.客户端App可调用[deleteKey()](#删除密钥)、[deleteKeyEx()](#删除密钥_扩展)接口删除已保存的密钥  **3.本接口在TOSS机型上不支持** | |

#### 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：参考附录：[密钥体系及密钥索引范围.SEC\_MKSK](#密钥体系及密钥索引范围_SEC_MKSK) |
| 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，  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](#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  <0-错误码 | |
| 备注 | 1.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  **2.本接口在TOSS机型上不支持** | |

#### 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：参考附录：[密钥体系及密钥索引范围.SEC\_SM2\_KEY](#密钥体系及密钥索引范围_SEC_SM2_KEY) |
| pubKey[out] | 包含以下key：  data：byte[]（长度64B），公钥数据  kcv：byte[]（长度5B），公钥kcv  rfu: byte[]（长度10B），RFU数据 |
| 返回值 | 0-成功  <0-错误码 | |
| 备注 | 1.本接口保存的密钥永久性存储，设备关机/重启、APP重启后密钥不丢失  2.客户端App可调用[deleteKey()](#删除密钥)、[deleteKeyEx()](#删除密钥_扩展)接口删除已保存的密钥  **3.本接口在TOSS机型上不支持** | |

#### 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：参考附录：[密钥体系及密钥索引范围.SEC\_SM2\_KEY](#密钥体系及密钥索引范围_SEC_SM2_KEY) |
| keyData[in] | 密钥数据，包含以下key：  data：byte[]（公钥长度64B，私钥长度32B），公钥/私钥数据  kcv：byte[]（长度5B，可选），公钥/私钥kcv  rfu: byte[]（长度10B，可选），公钥/私钥RFU数据 |
| 返回值 | 0-成功  <0-错误码 | |
| 备注 | 1.本接口保存的密钥永久性存储，设备关机/重启、APP重启后密钥不丢失  2.客户端App可调用[deleteKey()](#删除密钥)、[deleteKeyEx()](#删除密钥_扩展)接口删除已保存的密钥  **3.本接口在TOSS机型上不支持** | |

#### 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：参考附录：[密钥体系及密钥索引范围.SEC\_SM2\_KEY](#密钥体系及密钥索引范围_SEC_SM2_KEY) |
| pvkIndex[in] | 私钥索引，范围：参考附录：[密钥体系及密钥索引范围.SEC\_SM2\_KEY](#密钥体系及密钥索引范围_SEC_SM2_KEY) |
| userId[in] | 签名者ID，小于等于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，长度小于等于1024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  <0-错误码 | |
| 备注 | **1.本接口在TOSS机型上不支持** | |

#### 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：参考附录：[密钥体系及密钥索引范围.SEC\_SM2\_KEY](#密钥体系及密钥索引范围_SEC_SM2_KEY) |
| userId[in] | 签名者ID，小于等于512字节 |
| dataIn[in] | 待验签的数据，长度小于等于1024字节 |
| signData[in] | 签名数据，定长64B |
| 返回值 | 0-成功  <0-错误码 | |
| 备注 | **1.本接口在TOSS机型上不支持** | |

#### 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：参考附录：[密钥体系及密钥索引范围.SEC\_SM2\_KEY](#密钥体系及密钥索引范围_SEC_SM2_KEY) |
| dataIn[in] | 待加密的数据，长度大于0字节，小于等于1024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  <0-错误码 | |
| 备注 | **1.本接口在TOSS机型上不支持** | |

#### 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：参考附录：[密钥体系及密钥索引范围.SEC\_SM2\_KEY](#密钥体系及密钥索引范围_SEC_SM2_KEY) |
| 参数 | dataIn[in] | 待解密的数据，长度大于96字节，小于等于1120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  <0-错误码 | |
| 备注 | **1.本接口在TOSS机型上不支持** | |

#### 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) | |
|  | mode [in] | 摘要模式，参见 [Hash算法类型](#Hash算法类型) |
| 参数 | dataIn[in] | 待计算的数据，长度无限制 |
| 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  <0-错误码 | |
| 备注 | **1.本接口在TOSS机型上不支持** | |

#### 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：参考附录：[密钥体系及密钥索引范围.SEC\_MKSK](#密钥体系及密钥索引范围_SEC_MKSK) |
| macAlgType [in] | Mac算法类型，参考附录：Aidl常量.[MAC算法类型](#MAC算法类型) |
| dataIn [in] | 待验证数据 |
| mac[in] | mac数据 |
| 返回值 | 0-成功  <0-错误码 | |
| 备注 | **1.本接口在TOSS机型上不支持** | |

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

|  |  |  |
| --- | --- | --- |
| 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：参考附录：[密钥体系及密钥索引范围.SEC\_RSA\_KEY](#密钥体系及密钥索引范围_SEC_RSA_KEY) |
| 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  <0-错误码 | |
| 备注 | 1.本接口保存的密钥永久性存储，设备关机/重启、APP重启后密钥不丢失  2.客户端App可调用[deleteKey()](#删除密钥)、[deleteKeyEx()](#删除密钥_扩展)接口删除已保存的密钥  **3.本接口在TOSS机型上不支持** | |

#### 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] | 公钥/私钥索引，范围：参考附录：[密钥体系及密钥索引范围.SEC\_RSA\_KEY](#密钥体系及密钥索引范围_SEC_RSA_KEY) |
| 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  <0-错误码 | |
| 备注 | 1.本接口保存的密钥永久性存储，设备关机/重启、APP重启后密钥不丢失  2.客户端App可调用[deleteKey()](#删除密钥)、[deleteKeyEx()](#删除密钥_扩展)接口删除已保存的密钥  **3.本接口在TOSS机型上不支持** | |

#### 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：参考附录：[密钥体系及密钥索引范围.SEC\_RSA\_KEY](#密钥体系及密钥索引范围_SEC_RSA_KEY) |
| padding[in] | 填充模式，0-NoPadding，1-PKCS1Padding，2-PKCS7Padding |
| dataIn [in] | 待加密/解密数据，长度小于等于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  <0-错误码 | |
| 备注 | 1．该函数对input进行 RSA 加解密运算，运算结果输出到output；  2．当index索引指向私钥时：  （1）如果dataIn**是**对应公钥的加密密文，则dataOut为dataIn的明文（私钥解密）  （2）如果dataIn**不是**对应公钥的加密密文，则dataOut为dataIn 的密文（私钥加密）  3. 当index索引指向公钥时：  （1）如果dataIn**是**对应私钥的加密密文，则dataOut为dataIn的明文（公钥解密）  （2）如果dataIn**不是**对应私钥的加密密文，则dataOut为dataIn的密文（公钥加密）  4. 该函数可实现长度不超过 2048 bits 的 RSA 运算（RSA2048）  5.如果应用填充，则传入不填充参数  **6.本接口在TOSS机型上不支持** | |

#### 3.6.47 随机生成对称密钥

|  |  |  |
| --- | --- | --- |
| 原型 | int generateSymKey(int keyIndex, int keyType, int keyAlgType) | |
| 功能 | 随机生成对称密钥 | |
| 参数 | keyIndex[in] | 密钥索引，范围：参考附录：[密钥体系及密钥索引范围.SEC\_MKSK](#密钥体系及密钥索引范围_SEC_MKSK) |
| keyType[in] | 密钥类型，参考附录：Aidl常量类.[密钥类型定义](#密钥类型常量定义) |
| 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](#密钥算法类型常量定义) |
| 返回值 | 0-成功  非0-错误码 | |
| 备注 | 1.本接口保存的密钥永久性存储，设备关机/重启、APP重启后密钥不丢失  2.客户端App可调用[deleteKey()](#删除密钥)、[deleteKeyEx()](#删除密钥_扩展)接口删除已保存的密钥  **3.本接口在TOSS机型上不支持** | |

#### 3.6.48 注入对称密钥

|  |  |  |
| --- | --- | --- |
| 原型 | int injectSymKey(int keyIndex, int keyType, byte[] keyValue, byte[] checkValue, int keyAlgType); | |
| 功能 | 注入对称密钥 | |
| 参数 | keyIndex[in] | 密钥索引，范围：参考附录：[密钥体系及密钥索引范围.SEC\_MKSK](#密钥体系及密钥索引范围_SEC_MKSK) |
| keyType[in] | 密钥类型，参考附录：Aidl常量类.[密钥类型定义](#密钥类型常量定义) |
| keyValue [in] | 密钥数据 |
| checkValue[in] | 密钥校验值，参考：[KCV参数说明](#KCV参数说明) |
| 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](#密钥算法类型常量定义) |
| keyIndex[in] | 保存的索引 范围：0~199 |
| 返回值 | 0-成功  非0-错误码 | |
| 备注 | 1.仅支持对称明文注入MKSK密钥  2.本接口保存的密钥永久性存储，设备关机/重启、APP重启后密钥不丢失  3.客户端App可调用[deleteKey()](#删除密钥)、[deleteKeyEx()](#删除密钥_扩展)接口删除已保存的密钥  **4.本接口在TOSS机型上不支持** | |

#### 3.6.49 获取密钥checkValue（扩展）

|  |  |  |
| --- | --- | --- |
| 原型 | int getKeyCheckValueEx( Bundle bundle, byte[] dataOut) | |
| 功能 | 获取密钥的checkValue | |
| 参数 | bundle[in] | 入参，包含如下key：  keySystem：int，密钥体系，参考附录：Aidl常量类.[密钥体系常量](#密钥体系常量)  keyIndex：int，密钥索引，值范围如下：  keySystem为SEC\_MKSK时，范围：参考附录：[密钥体系及密钥索引范围.SEC\_MKSK](#密钥体系及密钥索引范围_SEC_MKSK)；  keySystem为SEC\_DUKPT 时，范围：参考附录：[密钥体系及密钥索引范围.SEC\_DUKPT](#密钥体系及密钥索引范围_SEC_DUKPT)  kcvMode：int，kcv模式，范围：参考附录：Aidl常量类.[kcv模式](#kcv模式)  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  <0-错误码 | |
| 备注 | **1.本接口在TOSS机型上不支持** | |

#### 3.6.50 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：  keySystem：int，密钥体系，参考附录：Aidl常量类.[密钥体系常量](#密钥体系常量)  keyIndex：int，密钥索引，值范围如下：  keySystem为SEC\_MKSK时，范围：参考附录：[密钥体系及密钥索引范围.SEC\_MKSK](#密钥体系及密钥索引范围_SEC_MKSK)；  keySystem为SEC\_DUKPT 时，范围：参考附录：[密钥体系及密钥索引范围.SEC\_DUKPT](#密钥体系及密钥索引范围_SEC_DUKPT) ；  keySystem为SEC\_RSA\_KEY时，范围：参考附录：[密钥体系及密钥索引范围.SEC\_RSA\_KEY](#密钥体系及密钥索引范围_SEC_RSA_KEY)  targetAppPkgName：String，目标应用包名 |
| 返回值 | 0-成功  <0-错误码 | |
| 备注 | **1.本接口在TOSS机型上不支持** | |

#### 3.6.51 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：  targetAppPkgName：String，目标应用包名  keyType：int，密钥类型：KEK/TMK/PIK/TDK/MAK/REC  keyValue：byte[]，密钥数据  kcvMode：int，kcv模式，参考：[Aidl常量类.kcv模式定义](#kcv模式)  kcvMacType：int，kcvMac算法类型，参见Aidl常量.[MAC算法类型](#MAC算法类型)  kcvInData：byte[]，用于计算kcv的数据  checkValue：byte[]，密钥校验值，参考：[KCV参数说明](#KCV参数说明)  encryptIndex：int，解密密钥的索引，范围：参考附录：[密钥体系及密钥索引范围.SEC\_MKSK](#密钥体系及密钥索引范围_SEC_MKSK)  keyAlgType：int，密钥算法类型，1-3Des，2-AES，3-SM4  keyIndex：int，密钥保存的位置索引，范围：参考附录：[密钥体系及密钥索引范围.SEC\_MKSK](#密钥体系及密钥索引范围_SEC_MKSK)  keyLength：int，密钥长度（明文）  dataMode：int，加密模式，参考附录：Aidl常量.[加密模式常量](#加密模式常量)  iv：byte[]，初始化向量 |
| 返回值 | 0-成功  <0-错误码 | |
| 备注 | 1.本接口保存的密钥永久性存储，设备关机/重启、APP重启后密钥不丢失  2.客户端App可调用[deleteKey()](#删除密钥)、[deleteKeyEx()](#删除密钥_扩展)接口删除已保存的密钥  **3.本接口在TOSS机型上不支持** | |

#### 3.6.52 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：  targetAppPkgName：String，目标应用包名  keyValue：byte[]，密钥数据  kcvMode：int，kcv模式，参考：[Aidl常量类.kcv模式定义](#kcv模式)  kcvMacType：int，kcvMac算法类型，参见Aidl常量.[MAC算法类型](#MAC算法类型)  kcvInData：byte[]，用于计算kcv的数据  checkValue：byte[]，密钥校验值，参考：[KCV参数说明](#KCV参数说明)  ksn：byte[]，密钥序列号  encryptIndex：int，解密密钥的索引，范围：参考附录：[密钥体系及密钥索引范围.SEC\_MKSK](#密钥体系及密钥索引范围_SEC_MKSK)  keyAlgType：int，密钥算法类型，1-3Des，2-AES  keyIndex：int，密钥保存的位置索引，范围：参考附录：[密钥体系及密钥索引范围.SEC\_DUKPT](#密钥体系及密钥索引范围_SEC_DUKPT)  isEncrypt：bool，是否密文  keyLength：int，密钥长度（明文）  dataMode：int，加密模式，参考附录：Aidl常量.[加密模式常量](#加密模式常量)  iv：byte[]，初始化向量 |
| 返回值 | 0-成功  <0-错误码 | |
| 备注 | 1.本接口保存的密钥永久性存储，设备关机/重启、APP重启后密钥不丢失  2.客户端App可调用[deleteKey()](#删除密钥)、[deleteKeyEx()](#删除密钥_扩展)接口删除已保存的密钥  **3.本接口在TOSS机型上不支持** | |

#### 3.6.53 保存明文或密文密钥（扩展）

|  |  |  |
| --- | --- | --- |
| 原型 | int saveKeyEx( Bundle bundle) | |
| 功能 | 保存明文或密文密钥 | |
| 参数 | bundle[in] | 入参，包含如下key：  keyType：int，密钥类型，参考附录：[Aidl常量类.密钥类型定义](#密钥类型常量定义)  keyValue：byte[]，密钥数据  checkValue：byte[]，密钥校验值，参考：[KCV参数说明](#KCV参数说明)  kcvMode：int，kcv模式，参考：[Aidl常量类.kcv模式定义](#kcv模式)  kcvMacType：int，kcvMac算法类型，参见Aidl常量.[MAC算法类型](#MAC算法类型)  kcvInData：byte[]，用于计算kcv的数据  encryptIndex：int，用于解密密钥密文的索引，注意，这里是TMK的索引，范围：参考附录：[密钥体系及密钥索引范围.SEC\_MKSK](#密钥体系及密钥索引范围_SEC_MKSK)  keyAlgType：int，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](#密钥算法类型常量定义)  keyIndex：int，密钥保存的位置索引，范围：参考附录：[密钥体系及密钥索引范围.SEC\_MKSK](#密钥体系及密钥索引范围_SEC_MKSK)  variantUsage：int，扩展变量的用法  keyVariant：byte[]，扩展变量  dataMode：int，加解密模式，参考附录：Aidl常量.[加密模式常量](#加密模式常量)  iv：byte[]，初始化向量  isEncrypt：bool，是否密文 |
| 返回值 | 0-成功  <0-错误码 | |
| 备注 | 1.本接口保存的密钥永久性存储，设备关机/重启、APP重启后密钥不丢失  2.客户端App可调用[deleteKey()](#删除密钥)、[deleteKeyEx()](#删除密钥_扩展)接口删除已保存的密钥  **3.参数kcvMacType、kcvInData仅在巴西CKD机型上支持，在其他机型上不支持**  **4.本接口在TOSS机型上不支持** | |

#### 3.6.54 计算mac（扩展）

|  |  |  |
| --- | --- | --- |
| 原型 | int calcMacExtended(Bundle bundle, byte[] dataOut) | |
| 功能 | 计算mac | |
| 参数 | bundle[in] | 入参，包含如下key：  keyIndex：int，mac密钥索引，范围：参考附录：[密钥体系及密钥索引范围.SEC\_MKSK](#密钥体系及密钥索引范围_SEC_MKSK)  keyLength：int，密钥长度，0-整个密钥，非0-密钥的前keyLength字节  macTypeint：mac算法，参见Aidl常量.[MAC算法类型](#MAC算法类型)  diversify：byte[]，分散因子，值为null，暂不支持  dataIn：byte[]，用于进行mac计算的源数据  iv：byte[]：初始化向量 |
| dataOut[out] | 计算生成的mac数据 |
| 返回值 | 0-成功  <0-错误码 | |
| 备注 | **1.本接口在TOSS机型上不支持** | |

#### 3.6.55 Dukpt密钥计算mac(扩展)

|  |  |  |
| --- | --- | --- |
| 原型 | int calcMacDukptExtended(Bundle bundle, byte[] dataOut) | |
| 功能 | dukpt密钥计算MAC | |
| 参数 | bundle[in] | 入参，包含如下key：  keySelect：密钥选择，参见Aidl常量.[Dukpt密钥选择](#Dukpt密钥选择)  keyIndex：int，密钥索引，范围：参考附录：[密钥体系及密钥索引范围.SEC\_DUKPT](#密钥体系及密钥索引范围_SEC_DUKPT)  keyLength：int，密钥长度，范围：0-整个密钥，非0-密钥的前keyLength字节  macType：int，mac算法，参见Aidl常量.[MAC算法类型](#MAC算法类型)  dataIn：byte[]，用于进行mac计算的源数据  iv：byte[]，初始化向量 |
| dataOut[out] | 计算生成的mac数据 |
| 返回值 | 0-成功  <0-错误码 | |
| 备注 | **1.本接口在TOSS机型上不支持** | |

#### 3.6.56 读取RSA密钥信息

|  |  |  |
| --- | --- | --- |
| 原型 | int readRSAKey(int keyIndex, Bundle keyInfo) | |
| 功能 | 读取RSA密钥信息 | |
| 参数 | keyIndex [in] | 密钥索引，范围：参考附录：[密钥体系及密钥索引范围.SEC\_MKSK](#密钥体系及密钥索引范围_SEC_MKSK) |
| keyInfo [out] | 出参，包含如下key：  modulus：byte[]，模  exponent：byte[]，指数 |
| 返回值 | 0-成功  <0-错误码 | |
| 备注 |  | |

#### 3.6.57 获取密钥长度（巴西CKD专用）

|  |  |  |
| --- | --- | --- |
| 原型 | int getKeyLength(int keySystem, int keyIndex) | |
| 功能 | 获取密钥长度 | |
| 参数 | keySystem [in] | 密钥体系，参考附录：Aidl常量类.[密钥体系常量](#密钥体系常量) |
| keyIndex [in] | 密钥索引，范围如下：  keySystem为SEC\_MKSK时，范围：参考附录：[密钥体系及密钥索引范围.SEC\_MKSK](#密钥体系及密钥索引范围_SEC_MKSK)  keySystem为SEC\_DUKPT 时，范围：参考附录：[密钥体系及密钥索引范围.SEC\_DUKPT](#密钥体系及密钥索引范围_SEC_DUKPT) |
| 返回值 | >=0：密钥长度  <0：错误码 | |
| 备注 | **1.本接口为巴西CKD机型上专用接口，在非巴西CKD机型上不支持**  **2.本接口在巴西CKD机型上暂不支持** | |

#### 3.6.58 保存可变密钥（巴西CKD专用）

|  |  |  |
| --- | --- | --- |
| 原型 | int writeKeyVariable(Bundle bundle) | |
| 功能 | 保存可变密钥 | |
| 参数 | bundle[in] | 入参，包含如下key：  密钥信息，包含如下key：  keyType：int，密钥类型，参考附录：Aidl常量类.[密钥类型定义](#密钥类型常量定义)  checkValue：byte[]，密钥校验值，参考：[KCV参数说明](#KCV参数说明)  kcvMode：int，kcv模式，参考：[Aidl常量类.kcv模式定义](#kcv模式)  kcvMacType：int，kcvMac算法类型，参见Aidl常量.[MAC算法类型](#MAC算法类型)  kcvInData：byte[]，用于计算kcv的数据  keyAlgType：int，加密类型，1-3DES，2-AES，3-SM4  srcKeyIndex：int，源密钥索引，范围：参考附录：[密钥体系及密钥索引范围.SEC\_MKSK](#密钥体系及密钥索引范围_SEC_MKSK)  destKeyIndex：int，目标密钥索引，范围：参考附录：[密钥体系及密钥索引范围.SEC\_MKSK](#密钥体系及密钥索引范围_SEC_MKSK)  xorData：byte[]，异或数据 |
| 返回值 | 0-成功  <0-错误码 | |
| 备注 | 1.本接口保存的密钥永久性存储，设备关机/重启、APP重启后密钥不丢失  2.客户端App可调用[deleteKey()](#删除密钥)、[deleteKeyEx()](#删除密钥_扩展)接口删除已保存的密钥  **3.本接口为巴西CKD机型上专用接口，在非巴西CKD机型上不支持**  **4.本接口在巴西CKD机型上暂不支持** | |

#### 3.6.59 密钥杂项操作

|  |  |  |
| --- | --- | --- |
| 原型 | int secKeyIoControl(int keyIndex, int ctrCode, byte[] dataIn, byte[] dataOut) | |
| 功能 | 密钥杂项操作 | |
| 参数 | keyIndex [in] | 密钥索引，范围：参考附录：[密钥体系及密钥索引范围.SEC\_MKSK](#密钥体系及密钥索引范围_SEC_MKSK) |
| ctrCode[in] | 密钥操作类型，参考 [密钥操作类型](#密钥操作类型) |
| dataIn[in] | 输入数据 |
| dataOut[out] | 输出数据 |
| 返回值 | >=0-dataOut中有效数据的长度  <0-错误码 | |
| 备注 | **1.本接口在TOSS机型上不支持** | |

#### 3.6.60 计算APACS mac

|  |  |  |
| --- | --- | --- |
| 原型 | int apacsMac(int initMakIndex, int makIndex, int pikIndex, int ctrCode, byte[] dataIn, byte[] dataOut) | |
| 功能 | 计算APACS Mac | |
| 参数 | initMakIndex[in] | 初始Mac密钥索引，范围：参考附录：[密钥体系及密钥索引范围.SEC\_MKSK](#密钥体系及密钥索引范围_SEC_MKSK) |
| makIndex[in] | 派生Mac密钥索引，范围：参考附录：[密钥体系及密钥索引范围.SEC\_MKSK](#密钥体系及密钥索引范围_SEC_MKSK) |
| pikIndex[in] | 派生PIN密钥索引，范围：参考附录：[密钥体系及密钥索引范围.SEC\_MKSK](#密钥体系及密钥索引范围_SEC_MKSK) |
| ctrCode[in] | 密钥操作类型，参考 [密钥操作类型](#密钥操作类型) |
| dataIn[in] | 输入数据 |
| dataOut[out] | 输出数据 |
| 返回值 | >=0-dataOut中有效数据的长度  <0-错误码 | |
| 备注 | **1.本接口在TOSS机型上不支持** | |

#### 3.6.61 保存密文密钥，解密密钥为RSA私钥

|  |  |  |
| --- | --- | --- |
| 原型 | int saveCiphertextKeyUnderRSA(int keyIndex, int rsaKeyIndex, int keyType, int keyAlgType, byte[] checkValue, byte[] keyData) | |
| 功能 | 保存MKSK密文密钥，解密密钥为RSA私钥 | |
| 参数 | keyIndex[in] | 密钥索引，范围：参考附录：[密钥体系及密钥索引范围.SEC\_MKSK](#密钥体系及密钥索引范围_SEC_MKSK) |
| rsaKeyIndex [in] | RSA私钥索引，范围：参考附录：[密钥体系及密钥索引范围.SEC\_RSA\_KEY](#密钥体系及密钥索引范围_SEC_RSA_KEY) |
| keyType [in] | 密钥类型，参考附录：Aidl常量类.[密钥类型定义](#密钥类型常量定义) |
| keyAlgType [in] | 密钥的算法类型，参考附录：Aidl常量类.[密钥算法类型](#密钥算法类型常量定义) |
| checkValue [in] | 密钥校验值，参考：[KCV参数说明](#KCV参数说明) |
| keyData [in] | 密钥数据 |
| 返回值 | 0-成功  <0-错误码 | |
| 备注 | 1.本接口保存的密钥永久性存储，设备关机/重启、APP重启后密钥不丢失  2.客户端App可调用[deleteKey()](#删除密钥)、[deleteKeyEx()](#删除密钥_扩展)接口删除已保存的密钥  **3.本接口在TOSS机型上不支持** | |

#### 3.6.62 注入密文密钥，解密密钥为RSA私钥

|  |  |  |
| --- | --- | --- |
| 原型 | int injectCiphertextKeyUnderRSA(String targetPkgName, int keyIndex, int rsaKeyIndex, int keyType, int keyAlgType, byte[] checkValue, byte[] keyData) | |
| 功能 | 注入MKSK密文密钥，解密密钥为RSA私钥 | |
| 参数 | targetPkgName[in] | 目标APP的包名 |
| keyIndex[in] | 密钥索引，范围：参考附录：[密钥体系及密钥索引范围.SEC\_MKSK](#密钥体系及密钥索引范围_SEC_MKSK) |
| rsaKeyIndex [in] | RSA私钥索引，范围：参考附录：[密钥体系及密钥索引范围.SEC\_RSA\_KEY](#密钥体系及密钥索引范围_SEC_RSA_KEY) |
| keyType [in] | 密钥类型，参考附录：Aidl常量类.[密钥类型定义](#密钥类型常量定义) |
| keyAlgType [in] | 密钥的算法类型，参考附录：Aidl常量类.[密钥算法类型](#密钥算法类型常量定义) |
| checkValue [in] | 密钥校验值，参考：[KCV参数说明](#KCV参数说明) |
| keyData [in] | 密钥数据 |
| 返回值 | 0-成功  <0-错误码 | |
| 备注 | 1.本接口保存的密钥永久性存储，设备关机/重启、APP重启后密钥不丢失  2.客户端App可调用[deleteKey()](#删除密钥)、[deleteKeyEx()](#删除密钥_扩展)接口删除已保存的密钥  **3.本接口在TOSS机型上不支持** | |

#### 3.6.63 随机生成对称密钥（扩展）

|  |  |  |
| --- | --- | --- |
| 原型 | int generateSymKeyEx(Bundle bundle) | |
| 功能 | 随机生成对称密钥 | |
| 参数 | bundle[in] | 入参，包含如下key：  keyIndex：int，密钥保存的位置索引，范围：参考附录：[密钥体系及密钥索引范围.SEC\_MKSK](#密钥体系及密钥索引范围_SEC_MKSK)  keyType：int，密钥类型，参考附录：[Aidl常量类.密钥类型定义](#密钥类型常量定义)  keyAlgType：int，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](#密钥算法类型常量定义)  keyLength：int，密钥长度(3DES-16/24字节, AES-16/24/32字节) |
| 返回值 | 0-成功  <0-错误码 | |
| 备注 | 1.本接口保存的密钥永久性存储，设备关机/重启、APP重启后密钥不丢失  2.客户端App可调用[deleteKey()](#删除密钥)、[deleteKeyEx()](#删除密钥_扩展)接口删除已保存的密钥  **3.本接口在TOSS机型上不支持** | |

#### 3.6.64 注入对称密钥（扩展）

|  |  |  |
| --- | --- | --- |
| 原型 | int injectSymKeyEx(Bundle bundle) | |
| 功能 | 注入对称密钥 | |
| 参数 | bundle[in] | 入参，包含如下key：  keyIndex：int，密钥保存的位置索引，范围：参考附录：[密钥体系及密钥索引范围.SEC\_MKSK](#密钥体系及密钥索引范围_SEC_MKSK)  keyType：int，密钥类型，参考附录：[Aidl常量类.密钥类型定义](#密钥类型常量定义)  keyValue：byte[]，密钥数据  checkValue：byte[]，密钥校验值，参考：[KCV参数说明](#KCV参数说明)  keyAlgType：int，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](#密钥算法类型常量定义)  keyLength 密钥长度(int，3DES-16/24字节，AES-16/24/32字节)  encryptIndex1：int，依赖的解密密钥索引1，范围：参考附录：[密钥体系及密钥索引范围.SEC\_MKSK](#密钥体系及密钥索引范围_SEC_MKSK)、[密钥体系及密钥索引范围.SEC\_RSA\_KEY](#密钥体系及密钥索引范围_SEC_RSA_KEY)、[密钥体系及密钥索引范围.SEC\_DEVICE\_CERT](#密钥体系及密钥索引范围_SEC_DEVICE_CERT)  encryptIndex2：int，依赖的解密密钥索引2（GOWF算法需要依赖两个解密密钥），范围：参考附录：[密钥体系及密钥索引范围.SEC\_MKSK](#密钥体系及密钥索引范围_SEC_MKSK)  dataMode：int，加解密模式，参考附录：Aidl常量.[加密模式常量](#加密模式常量)  iv：byte[]，初始化向量  injectMode：int，注入模式，参考附录：Aidl常量.[注入对称密钥模式](#注入对称密钥模式)，若：  （1）值为INJECT\_OAEP\_MODE（0x05），则keyValue为设备证书公钥加密后的密文值，加密时padding模式为PKCS1\_OAEP\_PADDING（0x04），依赖密钥encryptIndex1指向与设备证书公钥对应的设备证书私钥  （2）值为INJECT\_PKCS1\_MODE（0x06），则keyValue为RSA公钥加密后的密文值，加密时padding模式为PKCS1\_PADDING（0x01），依赖密钥encryptIndex1指向与RSA公钥对应的RSA私钥  （3）值为INJECT\_OAEP\_MODE、INJECT\_PKCS1\_MODE以外的值时，保存的密钥为明文密钥或通过指定算法派生的密钥，参考接口备注栏中的说明 |
| 返回值 | 0-成功  <0-错误码 | |
| 备注 | 1.支持：1）明文密钥；2）密文注入密钥（非对称密钥加密对称密钥），解密模式参照keyinfo字段（仅支持rsa加密mksk）；3）0x80 OWF2算法类型派生并保存密钥；4）0x81 OWF3算法类型派生并保存密钥；5）0x82 GOWF算法类型派生并保存密钥；6）0x83 ENC 算法类型派生并保存密钥  2.本接口保存的密钥永久性存储，设备关机/重启、APP重启后密钥不丢失  3.客户端App可调用[deleteKey()](#删除密钥)、[deleteKeyEx()](#删除密钥_扩展)接口删除已保存的密钥  **4.本接口在TOSS机型上不支持** | |

#### 3.6.65 注入设备证书与密文私钥

|  |  |  |
| --- | --- | --- |
| 原型 | int injectDeviceCertPrivateKey(Bundle bundle) | |
| 功能 | 注入设备证书与密文私钥 | |
| 参数 | bundle[in] | 密钥信息，包含如下key：  targetAppPkgName：String，目标应用包名  certIndex：int，证书索引，范围9001-9008  mode：int，模式，4-ECB模式，注入私钥密文使用  isEncrypt：boolean，是否密文  encryptIndex：int，对密文私钥进行解密的密钥索引，范围：参考附录：[密钥体系及密钥索引范围.SEC\_MKSK](#密钥体系及密钥索引范围_SEC_MKSK)  certData：byte[]，设备证书数据  pvkData：byte[]，私钥密文数据 |
| 返回值 | 0-成功  <0-错误码 | |
| 备注 | 1.本接口仅支持注入设备证书密文私钥，参数isEncrypt必须为true，否则注入失败  2.本接口保存的密钥永久性存储，设备关机/重启、APP重启后密钥不丢失  **3.本接口在TOSS机型上不支持** | |

#### 3.6.66 生成RSA公私玥对（扩展，仅支持1024/2048位密钥）

|  |  |  |
| --- | --- | --- |
| 原型 | int generateRSAKeypairEx(Bundle bundle, byte[] dataOut) | |
| 功能 | 生成RSA公私钥对（仅支持1024/2048位密钥） | |
| 参数 | bundle[in] | 入参，包含如下key：  keyType：int，RSA密钥类型，范围：0或KEY\_TYPE\_RSA\_KPK或KEY\_TYPE\_RSA\_KEK，参考附录：[Aidl常量类.密钥类型定义](#密钥类型常量定义)  pvkIndex：int，私钥索引，范围：参考附录：[密钥体系及密钥索引范围.SEC\_RSA\_KEY](#密钥体系及密钥索引范围_SEC_RSA_KEY)  keySize：int，密钥长度，支持1024/2048位密钥  pubExponent：String，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  <0-错误码 | |
| 备注 | 1.本接口保存的密钥永久性存储，设备关机/重启、APP重启后密钥不丢失  2.客户端App可调用[deleteKey()](#删除密钥)、[deleteKeyEx()](#删除密钥_扩展)接口删除已保存的密钥  **3.本接口目前仅在X30TR机型上支持，其他机型上不支持**  **4.本接口在TOSS机型上不支持** | |

#### 3.6.67 注入RSA密钥（扩展，仅支持1024/2048位密钥）

|  |  |  |
| --- | --- | --- |
| 原型 | int injectRSAKeyEx(Bundle bundle) | |
| 功能 | 注入RSA密钥（仅支持1024/2048位密钥） | |
| 参数 | bundle[in] | 入参，包含如下key：  keyType：int，RSA密钥类型，范围：0或KEY\_TYPE\_RSA\_KPK或KEY\_TYPE\_RSA\_KEK，参考附录：[Aidl常量类.密钥类型定义](#密钥类型常量定义)  keyIndex：int，公钥/私钥索引，范围：参考附录：[密钥体系及密钥索引范围.SEC\_RSA\_KEY](#密钥体系及密钥索引范围_SEC_RSA_KEY)  keySize：int，密钥长度，支持1024/2048位密钥  module：String，密钥模，Hex格式  exponent：String，密钥指数，Hex格式，支持03/010001 |
| 返回值 | 0-成功  <0-错误码 | |
| 备注 | 1.本接口保存的密钥永久性存储，设备关机/重启、APP重启后密钥不丢失  2.客户端App可调用[deleteKey()](#删除密钥)、[deleteKeyEx()](#删除密钥_扩展)接口删除已保存的密钥  **3.本接口目前仅在X30TR机型上支持，其他机型上不支持**  **4.本接口在TOSS机型上不支持** | |

#### 3.6.68 保存设备证书（TOSS专用）

|  |  |  |
| --- | --- | --- |
| 原型 | int setDeviceCertificate(int certIndex, byte[] certData) | |
| 功能 | 保存设备证书 | |
| 参数 | certIndex [in] | 设备证书保存的索引，范围：参考附录：[密钥体系及密钥索引范围.SEC\_DEVICE\_CERT](#密钥体系及密钥索引范围_SEC_DEVICE_CERT) |
| certData [in] | 证书数据 |
| 返回值 | 0-成功  <0-错误码 | |
| 备注 | 1.本接口保存的密钥永久性存储，设备关机/重启、APP重启后密钥不丢失  **2.本接口为TOSS机型上专用接口，在非TOSS机型上不支持** | |

#### 3.6.69 存储基础密钥

|  |  |  |
| --- | --- | --- |
| 原型 | int saveBaseKey(int destinationIndex, byte[] keyData) | |
| 功能 | 存储基础密钥 | |
| 参数 | destinationIndex [in] | 需要保存的密钥索引，范围：[1,200] |
| keyData[in] | 密钥数据密文，长度512B |
| 返回值 | 0-成功  <0-错误码 | |
| 备注 | 1.本接口保存的密钥永久性存储，设备关机/重启、APP重启后密钥不丢失  2.客户端App可调用[deleteKey()](#删除密钥)、[deleteKeyEx()](#删除密钥_扩展)接口删除已保存的密钥  **3.本接口在TOSS机型上不支持** | |

#### 3.6.70 加密数据（扩展）

|  |  |  |
| --- | --- | --- |
| 原型 | int dataEncryptEx(Bundle bundle, byte[] dataOut) | |
| 功能 | 加密数据 | |
| 参数 | bundle[in] | 入参，包含如下key：  keyIndex：int，范围：参考附录：[密钥体系及密钥索引范围.SEC\_MKSK](#密钥体系及密钥索引范围_SEC_MKSK)  keyLength：密钥长度，0-整个密钥，>0-密钥的前keyLen字节，<0-非法值  dataIn：byte[]，待加密的数据  encryptionMode：int，加密模式，参考附录：Aidl常量.[加密模式常量](#加密模式常量)  iv：byte[]，初始向量，加密模式为ECB 时传null，为其他加密模式传入8B或16B向量 |
| dataOut[out] | 计算生成的密文数据 |
| 返回值 | 0-成功  <0-错误码 | |
| 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  2.dataOut的长度应和dataIn的长度相等  例如：byte[] dataIn = new byte[16];  那么dataOut的长度也应该为16字节  3.本接口的**keyLength**参数仅对TDES密钥有效，对其他密钥(SM4、AES等)无效  **4.本接口在TOSS机型上不支持** | |

#### 3.6.71 解密数据（扩展）

|  |  |  |
| --- | --- | --- |
| 原型 | int dataDecryptEx(Bundle bundle, byte[] dataOut) | |
| 功能 | 解密数据 | |
| 参数 | bundle[in] | 入参，包含如下key：  keyIndex：int，范围：参考附录：[密钥体系及密钥索引范围.SEC\_MKSK](#密钥体系及密钥索引范围_SEC_MKSK)  keyLength：密钥长度，0-整个密钥，>0-密钥的前keyLen字节，<0-非法值  dataIn：byte[]，待加密的数据  encryptionMode：int，加密模式，参考附录：Aidl常量.[加密模式常量](#加密模式常量)  iv：byte[]，初始向量，加密模式为ECB 时传空，为其他加密模式传入8B或16B向量 |
| dataOut[out] | 计算生成的密文数据 |
| 返回值 | 0-成功  <0-错误码 | |
| 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  2.dataOut的长度应和dataIn的长度相等  例如：byte[] dataIn = new byte[16];  那么dataOut的长度也应该为16字节  3.本接口的**keyLength**参数仅对TDES密钥有效，对其他密钥(SM4、AES等)无效  **4.本接口在TOSS机型上不支持** | |

#### 3.6.72 查询密钥映射记录（返回调用者或targetApp名下的记录）

|  |  |  |
| --- | --- | --- |
| 原型 | int queryKeyMappingRecordList(Bundle bundle, List<Bundle> list) | |
| 功能 | 查询调用者名下或targetApp名下的密钥映射记录 | |
| 参数 | bundle[in] | 入参，不可为null，包含如下key：  targetPkgName：String，目标应用(target App)包名（本参数为可选参数）：（1）若不传此参数，则查询接口调用者App(caller)名下的密钥映射记录；  （2）若传此参数，则当caller的签名与target App签名相同时，查询target App名下的密钥映射记录；当caller的签名与target App签名不同时，查询失败，返回0条记录 |
| list[out] | 出参，每条记录包含如下key：  pkgName：String，密钥所属APP的包名  signature：String，密钥所属APP的签名，Hex格式  keySystem：String，密钥体系，范围：SEC\_MKSK，SEC\_DUKPT，SEC\_RSA\_KEY，SEC\_SM2\_KEY，SEC\_ECC\_KEY，SEC\_CERT，SEC\_DEVICE\_CERT，SEC\_MKSK\_NOLOST，SEC\_RSA\_KEY\_NOLOST，SEC\_ECC\_KEY\_NOLOST，SEC\_CERT\_NOLOST，SEC\_UNKNOWN  keyIndexRaw：int，原始密钥索引，参考附录：[密钥体系及密钥索引范围](#密钥体系及密钥索引范围)  keyIndexMapped: int, 映射后的密钥索引(SP中的密钥索引)，参考  附录：[密钥体系及密钥索引范围](#密钥体系及密钥索引范围)  keyType：String，密钥类型，范围：BASE\_KEY，KEK，TMK，PIK,MAK，TDK,REC，DUPKT\_BDK，DUPKT\_IPEK，KBPK,TADK，RSA\_PUK，RSA\_PVK，RSA\_PUK\_KPK，RSA\_PVK\_KPK，SM2\_PUK，SM2\_PVK，ECC\_PUK，ECC\_PVK，RSA\_CERT，DEVICE\_CERT\_PVK，UNKNOWN  keyAlgType：String，密钥的算法类型，范围：ALG\_3DES，ALG\_AES，ALG\_SM4，ALG\_UNKNOWN  checkValue：String，密钥的kcv，Hex格式，kcv模式为KCV\_MODE\_CHK0  injectFlag：String，密钥的注入标志，范围：null，injected，occupied  keySize：int，对称密钥的长度或非对称密钥的bit长度，如果获取值失败，返回-1  ksn：String，dukpt密钥的ksn，Hex格式。若密钥不是dukpt密钥，此字段为null |
| 返回值 | 0-成功  <0-错误码 | |
| 备注 | **1.本接口在TOSS机型上不支持** | |

#### 3.6.73 查询密钥映射记录（返回设备上全部密钥映射记录）

|  |  |  |
| --- | --- | --- |
| 原型 | int queryKeyMappingRecordListWL(List<Bundle> list) | |
| 功能 | 查询设备上全部密钥映射记录 | |
| 参数 | list[out] | 出参，每条记录包含如下key：  pkgName：String，密钥所属APP的包名  signature：String，密钥所属APP的签名，Hex格式  keySystem：String，密钥体系，范围：SEC\_MKSK，SEC\_DUKPT，SEC\_RSA\_KEY，SEC\_SM2\_KEY，SEC\_ECC\_KEY，SEC\_CERT，SEC\_DEVICE\_CERT，SEC\_MKSK\_NOLOST，SEC\_RSA\_KEY\_NOLOST，SEC\_ECC\_KEY\_NOLOST，SEC\_CERT\_NOLOST，SEC\_UNKNOWN  keyIndexRaw：int，原始密钥索引，参考附录：[密钥体系及密钥索引范围](#密钥体系及密钥索引范围)  keyIndexMapped: int, 映射后的密钥索引(SP中的密钥索引)，参考  附录：[密钥体系及密钥索引范围](#密钥体系及密钥索引范围)  keyType：String，密钥类型，范围：BASE\_KEY，KEK，TMK，PIK,MAK，TDK,REC，DUPKT\_BDK，DUPKT\_IPEK，KBPK,TADK，RSA\_PUK，RSA\_PVK，RSA\_PUK\_KPK，RSA\_PVK\_KPK，SM2\_PUK，SM2\_PVK，ECC\_PUK，ECC\_PVK，RSA\_CERT，DEVICE\_CERT\_PVK，UNKNOWN  keyAlgType：String，密钥的算法类型，范围：ALG\_3DES，ALG\_AES，ALG\_SM4，ALG\_UNKNOWN  checkValue：String，密钥的kcv，Hex格式，kcv模式为KCV\_MODE\_CHK0  injectFlag：String，密钥的注入标志，范围：null，injected，occupied  keySize：int，对称密钥的长度或非对称密钥的bit长度，如果获取值失败，返回-1  ksn：String，dukpt密钥的ksn，Hex格式。若密钥不是dukpt密钥，此字段为null |
| 返回值 | 0-成功  <0-错误码 | |
| 备注 | **1.本接口在TOSS机型上不支持** | |

#### 3.6.74 读取SM2公钥信息

|  |  |  |
| --- | --- | --- |
| 原型 | int readSM2Key(int keyIndex, Bundle keyInfo) | |
| 功能 | 读取SM2公钥信息 | |
| 参数 | keyIndex[in] | SM2公钥索引，范围：参考附录：[密钥体系及密钥索引范围.SEC\_SM2](#密钥体系及密钥索引范围_SEC_SM2_KEY) |
| keyInfo[out] | 出参，包含如下key：  keyData：byte[]，钥数据，长度：64B |
| 返回值 | 0-成功  <0-错误码 | |
| 备注 | **1.本接口在TOSS机型上不支持** | |

#### 3.6.75 加入Z(ID)值计算SM3哈希值

|  |  |  |
| --- | --- | --- |
| 原型 | int calcSM3HashWithID(int keyIndex, byte[] userId, byte[] dataIn, byte[] dataOut) | |
| 功能 | 加入Z(ID)值计算SM3哈希值 | |
| 参数 | keyIndex[in] | SM2公钥索引，范围：参考附录：[密钥体系及密钥索引范围.SEC\_SM2](#密钥体系及密钥索引范围_SEC_SM2_KEY) |
| userId[in] | ID值，长度不超过32B |
| dataIn[in] | 输入数据，长度不超过896B |
| dataOut[out] | buffer，存放hash数据，长度：32B |
| 返回值 | >=0-dataOut中有效数据的长度  <0-错误码 | |
| 备注 | **1.本接口在TOSS机型上不支持** | |

#### 3.6.76 基于Z(ID)的SM3哈希值计算SM2签名

|  |  |  |
| --- | --- | --- |
| 原型 | int sm2SingleSign(int keyIndex, byte[] hash, byte[] dataOut) | |
| 功能 | 基于Z(ID)的SM3哈希值计算SM2签名 | |
| 参数 | keyIndex[in] | SM2私钥索引，范围：参考附录：[密钥体系及密钥索引范围.SEC\_SM2](#密钥体系及密钥索引范围_SEC_SM2_KEY) |
| hash[in] | SM3哈希值（**calcSM3HashWithID()**接口的计算结果），长度：32B |
| dataOut[out] | buffer，存放SM2签名数据，长度：64B |
| 返回值 | >=0-dataOut中有效数据的长度  <0-错误码 | |
| 备注 | **1.本接口在TOSS机型上不支持** | |

#### 3.6.77 注入TR31密钥

|  |  |  |
| --- | --- | --- |
| 原型 | int injectTR31Key(Bundle bundle) | |
| 功能 | 给targetapp注入TR31密钥 | |
| 参数 | bundle[in] | 入参，包含如下key：  targetPkgName：String，目标App的包名  keyValue：byte[]，TR31密钥数据  kbpkIndex：int，KBPK索引，范围：参考附录：[密钥体系及密钥索引范围.SEC\_MKSK](#密钥体系及密钥索引范围_SEC_MKSK)  keyIndex：int，密钥索引，范围：参考附录：[密钥体系及密钥索引范围.SEC\_MKSK](#密钥体系及密钥索引范围_SEC_MKSK)、[密钥体系及密钥索引范围.SEC\_DUKPT](#密钥体系及密钥索引范围_SEC_DUKPT) |
| 返回值 | 0-成功  <0-错误码 | |
| 备注 | 1.保存TR31密钥前必须先保存KBPK  2.目前仅支持保存TR31 dukpt-IPEK，不支持保存TR31 dukpt-BDK  3.本接口保存的密钥永久性存储，设备关机/重启、APP重启后密钥不丢失  4.客户端App可调用[deleteKey()](#删除密钥)、[deleteKeyEx()](#删除密钥_扩展)接口删除已保存的密钥  **5.本接口在TOSS机型上不支持** | |

### 3.7 EMV操作模块

#### 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](#AidV2) |
| 返回值 | 0-添加或修改成功  非0-错误码 | |
| 备注 | 1.SDK默认内置了一些AID参数，内置的AID参考：[EMV默认AID](#_7.9.1_EMV默认AID) | |

##### 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  非0-错误码 | |
| 备注 | 1.SDK默认内置了一些AID参数，内置的AID参考：[EMV默认AID](#_7.9.1_EMV默认AID) | |

##### 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](#CapkV2) |
| 返回值 | 0-添加或修改成功  非0-错误码 | |
| 备注 | 1.SDK默认内置了一些CAPK参数，内置的CAPK参考：[EMV默认CAPK](#_7.9.2_EMV默认CAPK) | |

##### 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  非0-错误码 | |
| 备注 | 1.SDK默认内置了一些CAPK参数，内置的CAPK参考：[EMV默认CAPK](#_7.9.2_EMV默认CAPK) | |

##### 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](#EmvTermParamV2) |
| 返回值 | 0-修改成功  非0-错误码 | |
| 备注 |  | |

##### 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 | 无 |
| 返回值 | -1：aid和capk都不存在  0：aid和capk都存在  1：aid存在，capk不存在  2：capk存在，aid不存在 |
| 备注 |  |

##### 3.7.1.7 初始化EMV流程

|  |  |
| --- | --- |
| 原型 | int initEmvProcess () |
| 功能 | 初始化EMV流程 |
| 参数 | 无 |
| 返回值 | 0-成功  非0-错误码 |
| 备注 |  |

##### 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](#EMVTransDataV2) |
| listener [in] | Listener EMV回调接口，详见 [EMVListnerV2](#EMVListenerV2) |
| 返回值 | 无 | |
| 备注 |  | |

##### 3.7.1.9 读取单条内核数据

|  |  |  |
| --- | --- | --- |
| 原型 | int getTlv(int oolea,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  =0-没有读取到对应的TAG  <0-错误码 | |
| 备注 | 7.当开启sred功能后，此接口不返回账户敏感tag的值，参考[SRED说明](#sred说明) | |

##### 3.7.1.10 读取内核数据列表

|  |  |  |
| --- | --- | --- |
| 原型 | int getTlvList(int oolea,String[] tags, byte[] outData) | |
| 功能 | 读取内核数据 | |
| 参数 | oolea[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  =0-没有读取到对应的TAG列表值  <0-错误码 | |
| 备注 | 1.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  2.当开启sred功能后，此接口不返回账户敏感tag的值，参考[SRED说明](#sred说明) | |

##### 3.7.1.11 设置TLV数据

|  |  |  |
| --- | --- | --- |
| 原型 | void setTlv(int oolea,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 | oolea [in] | 操作类型，参见 [TLV操作类型定义](#EMV_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 | 无 | |
| 备注 |  | |

##### 3.7.1.12 设置TLV数据列表

|  |  |  |
| --- | --- | --- |
| 原型 | void setTlvList(int oolea,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 | oolea [in] | 操作类型，参见 [TLV操作类型定义](#EMV_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 | 无 | |
| 备注 |  | |

##### 3.7.1.13 导入应用选择结果

|  |  |  |
| --- | --- | --- |
| 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 | 无 | |
| 备注 |  | |

##### 3.7.1.14 导入应用最终选择结果

|  |  |  |
| --- | --- | --- |
| 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 | 无 | |
| 备注 |  | |

##### 3.7.1.15 导入卡号确认结果

|  |  |  |
| --- | --- | --- |
| 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 | 无 | |
| 备注 |  | |

##### 3.7.1.16 导入身份认证结果

|  |  |  |
| --- | --- | --- |
| 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 | 无 | |
| 备注 |  | |

##### 3.7.1.17 导入PIN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，4-输PIN超时 |
| 返回值 | 无 | |
| 备注 |  | |

##### 3.7.1.18 导入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  <0-错误码 | |
| 备注 |  | |

##### 3.7.1.19 导入签名结果

|  |  |  |
| --- | --- | --- |
| 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 | 无 | |
| 备注 |  | |

##### 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  非0-错误码 | |
| 备注 |  | |

##### 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 | 无 |
| 返回值 | 无 |
| 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 | 无 | |
| 备注 |  | |

##### 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](#EMVTransDataV2)，可按如下格式传值:  Bundle bundle = new Bundle();  bundle.putString(“amount”, amount); //交易金额  bundle.putString(“transType”, transType);//交易类型  bundle.putInt(“flowType”, flowType);//流程类型，参见Aidl常量. [EMV FlowType定义](#EMVFlowType定义)  bundle.putInt(“cardType”, cardType);//卡类型，2-IC，4-NFC  bundle.putString(“cashbackAmount”, cashbackAmount); //返现值  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](#EMVListenerV2) |
| 返回值 | 无 | |
| 备注 |  | |

##### 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：  9F51：String，应用货币代码（Hex格式）  9F79：long，电子现金余额 |
| 返回值 | 无 | |
| 备注 |  | |

##### 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](#DrlV2) |
| 返回值 | 0-成功  非0-错误码 | |
| 备注 |  | |

##### 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  非0-错误码 | |
| 备注 |  | |

##### 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：  supportDRL：boolean，是否支持DRL功能  downloadAidParam：boolean，是否下发AID参数  downloadAidParamAll：boolean，是否下发AID参数ALL  downloadPreParamEP：boolean，是否下发EntryPoint预处理参数  optOnlineRes：boolean，是否优化联机结果  ledLightingDuration：int，led点亮时长  contactlessManualSelApp：boolean，非接手动选择应用  contactlessManualSelAppGeneral：boolean，非接手动选择应用（通用版本）  contactlessManualSelAppGeneralEx：boolean，非接手动选择应用(通用版本)扩展  supportEP：boolean，是否支持EP  importScriptData：boolean，联机拒绝导入脚本数据  quickChip：boolean，是否走quickChip流程  noSignatureOrPINThreshold：int，quickChip模式下引起输PIN的阈值，金额超过此值时需要输PIN  dpasV2Support：boolean, 支持discoverV2.0  dpasDeferredAuthSupport：boolean,支持discover延迟认证功能  dpasDataStorageSupport：boolean，支持discover Data Storage功能  dpasExtendedLoggingSupport：boolean，支持discover ExtendedLogging功能  dpasTearingRecoverySupport：boolean，支持discover 闪卡功能  dpasContactlessSpeedupSupport：boolean，设置支持DPAS非接提速流程  jcbContactlessSpeedupSupport：boolean，设置支持JCB非接提速流程  AEContactlessSpeedupSupport：boolean，设置支持AE非接提速流程  AEOnlineProcessSupport：boolean，设置支持AE内核进行联机后的处理  supportPOI：boolean，设置EP POI 开关  CertifiedEP：boolean，设置走Certified EP流程  AutoRun：boolean，设置 Auto run 开关  KernelsForCertEP：long，设置哪些内核支持Certified EP  SupportAE4.1：boolean，设置支持AE 4.1规范 |
| 返回值 | 无 | |
| 备注 |  | |

##### 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  非0-错误码 | |
| 备注 |  | |

##### 3.7.1.29 交易预处理

|  |  |
| --- | --- |
| 原型 | int transactPreProcess() |
| 功能 | EMV交易预处理 |
| 参数 | 无 |
| 返回值 | 0-成功  非0-错误码 |
| 备注 |  |

##### 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](#RevocationList实体类) |
| 返回值 | 0-成功  非0-错误码 | |
| 备注 |  | |

##### 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](#RevocationList实体类)。若传入的值为null，则清空所有RevocationList |
| 返回值 | 0-成功  非0-错误码 | |
| 备注 |  | |

##### 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 |
| 返回值 | 0-成功  非0-错误码 | |
| 备注 |  | |

##### 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 | |
|  | 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  <0-错误码 | |
| 备注 |  | |

##### 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](#清除数据操作码定义) |
| 返回值 | 0-成功  非0-错误码 | |
| 备注 |  | |

##### 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：  encKeySystem: int, 密钥体系，参见 [密钥体系常量](#密钥体系常量)，仅支持SEC\_MKSK，SECK\_DUKPT，SEC\_RSA  encKeyIndex: int，磁道数据加密密钥索引，一般传入TDK索引，参考附录：[密钥体系及密钥索引范围](#_7.6_密钥体系及密钥索引范围)  encKeyAlgType：int，密钥的算法类型，参考附录：Aidl常量类.[密钥算法类型](#密钥算法类型常量定义)  encMode: int，加密模式，参考附录：Aidl常量.[加密模式常量](#加密模式常量)，仅当encKeySystem为SEC\_MKSK，SECK\_DUKPT时有效  encIv: byte[]，初始向量，加密模式为ECB时传null，为其他加密模式时传入8或16字节向量。仅当encKeySystem为SEC\_MKSK，SECK\_DUKPT时有效  encPaddingMode: byte，当磁道数据进行DES/AES/SM4加密时，若数据长度不是8/16的倍数，则在后面补齐EncPaddingMode至长度为8/16的倍数；若磁道数据进行RSA加密，则填充模式参考 [RSA填充模式](#RSA填充模式)，且不支持 NOTHING\_PADDING模式  encMaskStart: int，表示账号前EncMaskStart位为明文，范围是0~6  encMaskEnd: int，表示账号后EncMaskEnd位为明文，范围是0~4  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‘\*’  panAppendContent：String，RSA对track2加密时前缀/后缀数据（TID）  panAppendMode：int，RSA对track2加密时前后缀模式，0-前缀式（输入数据为TID+track2），1-后缀式（输入数据为track2+TID）  sred: boolean，默认值为true |
| 返回值 | 0-成功  非0-错误码 | |
| 备注 | 1.参数bundle中不包含关键字sred或包含关键字sred且值为true时将打开sred功能，参考[SRED说明](#sred说明)  2.如果参数bundle中包含关键字sred且值为false时将关闭sred功能，同时除sred外的其他关键字被忽略 | |

##### 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 | |
|  | opCode[in] | 操作码，参见 [TLV操作类型定义](#EMV_TLV操作类型定义) |
| 参数 | 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:  tag+Enc，如{“5AEnc”,“57Enc”}: String, tag加密数据  panMask: String, PAN的掩码数据，只有tag 5A和57存在  若加密密钥为RSA密钥，则输出数据中包含key：  appendedPanEncBytes：byte[]，加密后的TID+track2或track2+TID数据  appendedPanEnc：String，加密后的TID+track2或track2+TID数据，Hex格式 |
| 返回值 | 0-成功  非0-错误码 | |
| 备注 |  | |

##### 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 | 无 | |
| 备注 |  | |

##### 3.7.1.38 导入第一次GAC前调用结果

|  |  |  |
| --- | --- | --- |
| 原型 | void importPreFirstGenACStatus(int status) | |
| 功能 | 导入第一次GAC前调用结果 | |
| 参数 | status[in] | 第一次GAC前调用结果，0-成功，非0-失败 |
| 返回值 | 无 | |
| 备注 |  | |

##### 3.7.1.39 导入DataStorage相关数据

|  |  |  |
| --- | --- | --- |
| 原型 | void importDataStorage(String[] tags, String[] hexValues) | |
| 功能 | 导入DataStorage相关数据 | |
| 参数 | tags[in] | DataStorage相关数据tag列表 |
| hexValues[in] | DataStorage相关数据value列表 |
| 返回值 | 无 | |
| 备注 |  | |

##### 3.7.1.40 添加EMV回调监听器

|  |  |  |
| --- | --- | --- |
| 原型 | void addEMVDataListener(EMVDataListenerV2 listener) | |
| 功能 | 添加EMV回调监听器 | |
| 参数 | listener [in] | EMV回调接口 |
| 返回值 | 无 | |
| 备注 |  | |

##### 3.7.1.41 添加DET数据

|  |  |  |
| --- | --- | --- |
| 原型 | int addDETData(byte[] data) | |
| 功能 | 添加DET数据 | |
| 参数 | data [in] | DET数据 |
| 返回值 | 0-成功  <0-错误码 | |
| 备注 |  | |

##### 3.7.1.42 数据输入输出处理

|  |  |  |
| --- | --- | --- |
| 原型 | int dataInputOutputProcess(int mode, int procType, byte[] inData, byte[] outData) | |
| 功能 | 可用于调试EMV功能、相关参数的设置和获取，但不可用于控制EMV交易流程 | |
| 参数 | mode[in] | 输入/输出模式 0-输入，1-输出 |
| procType [in] | 处理类型 |
| inData[in] | 输入数据 |
| outData[out] | 输出数据 |
| 返回值 | >=0-outData中有效数据的长度  <0-错误码 | |
| 备注 | 1.本接口仅限SDK测试使用，客户端APP客户端APP请勿调用 | |

##### 3.7.1.43 导入PIN输入结果（TOSS专用）

|  |  |  |
| --- | --- | --- |
| 原型 | void importPinInputStatusForToss(byte[] pinValue, int inputResult) | |
| 功能 | 导入PIN输入结果 | |
| 参数 | pinValue [in] | PIN值 |
| inputResult [in] | PIN输入结果：0-处理成功，1-PIN取消，2-PIN跳过，3-PINPAD故障，4-输PIN超时 |
| 返回值 | 无 | |
| 备注 | **1.本接口为TOSS机型上专用接口，在非TOSS机型上不支持** | |

#### 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](#EMVCandidateV2) | |
| 参数 | candList[in] | 应用列表 |
| 参数 | isFirstSelect[in] | 是否是第一次选择 |
| 返回值 | 无 | |
| 备注 |  | |

##### 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 | 无 | |
| 备注 |  | |

##### 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 | 无 | |
| 备注 | 1.当开启sred功能后，参数cardNO为掩码处理后的数据，参考[SRED说明](#sred说明) | |

##### 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 | 无 | |
| 备注 |  | |

##### 3.7.2.5 请求输PIN

|  |  |  |
| --- | --- | --- |
| 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 | 无 | |
| 备注 |  | |

##### 3.7.2.6 请求签名

|  |  |
| --- | --- |
| 原型 | void onRequestSignature() |
| 功能 | 请求签名（如果需要） |
| 参数 | 无 |
| 返回值 | 无 |
| 备注 |  |

##### 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 | 无 |
| 返回值 | 无 |
| 备注 |  |

##### 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 | 无 |
| 返回值 | 无 |
| 备注 |  |

##### 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：  1-脱机批准，2-脱机拒绝，3-预留，4-重新拍卡，5-联机批准，6-联机拒绝  <0-错误码 |
| 参数 | desc[in] | 返回码对应的错误信息 |
| 返回值 | 无 | |
| 备注 |  | |

##### 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 | 无 |
| 返回值 | 无 |
| 备注 |  |

##### 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 | 无 | |
| 备注 |  | |

##### 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 | 无 |
| 返回值 | 无 |
| 备注 |  |

##### 3.7.2.13 第一次GAC前调用请求

|  |  |
| --- | --- |
| 原型 | void onPreFirstGenAC() |
| 功能 | 第一次GAC前调用请求 |
| 参数 | 无 |
| 返回值 | 无 |
| 备注 |  |

##### 3.7.2.14 请求DataStorage处理

|  |  |  |
| --- | --- | --- |
| 原型 | void onDataStorageProc(String[] containerID, String[] containerContent) | |
| 功能 | 请求DataStorage处理 | |
| 参数 | containerID[in] | 容器ID |
| 参数 | containerContent[in] | 容器内容 |
| 返回值 | 无 | |
| 备注 |  | |

### 3.8 税控模块（TOSS不支持）

#### 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  <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）  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）  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）  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）：  读-发送>>：1B 1D 09 00 06  读-接收<<：1B 1D 00 00 06 04 01 00 02 EE DB  写-发送>>：1B 1D 08 00 0B 04 06 00 11 11 03 02 0A 15 39 18  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 | 无 |
| 返回值 | 0-成功  <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 | 无 |
| 返回值 | 0-成功，  <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  <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  <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 | 无 |
| 返回值 | >0-成功，参考附录：打印机状态常量  <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 | 无 |
| 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  <0-错误码 | |
| 备注 | 灰度值level仅支持被5整除的数值 | |

##### 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 | 无 |
| 返回值 | >=0-值为缓冲剩余字节数  <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 | 无 |
| 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 | 无 |
| 返回值 | >=0-灰度值  <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 | 无 |
| 返回值 | >=0-累计打印的距离  <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 | 无 |
| 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 | 无 | |
| 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 | 无 |
| 返回值 | 无 |
| 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  <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  <0-错误码 | |
| 备注 | **1. 80打印机设置加热点不支持设置参数128** | |

##### 3.9.1.17 清除打印点行

|  |  |
| --- | --- |
| 原型 | int clearBuffer() |
| 功能 | 清除打印点行 |
| 参数 | 无 |
| 返回值 | 0-成功  <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 | 无 | |
| 备注 |  | |

### 3.10 ETC模块（TOSS不支持）

#### 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  <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》 | |

##### 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](#ETCSearchListenerV2) |
| timeout[in] | 超时时间 |
| 返回值 | 无 | |
| 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 | 无 |
| 返回值 | 无 |
| 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：  channel：int，通讯信道，0-5.79G信道，1-5.80G信道，默认为0  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**）  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  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  <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](#ETCSearchTradeOBUListenerV2) |
| 返回值 | 0-成功  <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：  allRet：String，车辆信息密文(Hex) |
| 返回值 | 0-成功  <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：  cardType：int，卡片类型（1B），00-储值卡，01-记账卡，02-非法卡片  balance：int，卡片余额（4B），单位：分（0002文件）  0019File：byte[]，0019文件（Hex，43B，交易记录文件）  allRet：所有返回数据(Hex，48B) |
| 返回值 | 0-成功  <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：  balance：String，电子存折或电子钱包旧余额（Hex，4B）  offlineTradeNo：String，电子存折或电子钱包脱机交易序号（Hex，2B）  overdrawLimit：String，透支限额（Hex，3B）  keyVersion：String，密钥版本号（Hex，1B）  algorithmId：String，算法标志（Hex，1B）  pseudorandomNum：String，伪随机数（Hex，4B）  allRet：String，所有返回数据(Hex，15B) |
| 返回值 | 0-成功  <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：  tac：String，TAC(Hex，4B)  mac2：String，Mac2(Hex，4B)  allRet：String，所有返回数据(Hex，8B) |
| 返回值 | 0-成功  <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  <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  <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](#ETCInfoV2) |
| 返回值 | 无 | |
| 备注 |  | |

##### 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 | 无 | |
| 备注 |  | |

#### 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：  deviceNo：String，设备编号（Hex，4B）  deviceStatus：int，设备状态：  bit7：0-卡片存在，1-卡片不存在；  bit1：0-设备正常，1-设备失效（已拆卸）；  bit0：0-电量正常，1-设备低电；  sysInfoIssuerId：String，系统信息-发行方标志（Hex，8B）  sysInfoContractNo：String，系统信息-合同序列号（Hex，8B）  0015File：0015文件内容（Hex，43B）  0015CardIssuerId：String，0015文件-发卡方标志（Hex，8B）  0015CardTypeId：String，0015文件-卡类型标志（Hex,1B）  0015CardVersion：String，0015文件-卡片版本号（Hex,1B）  0015CardNetId：String，0015文件-卡片网络标志（Hex，2B）  0015CardInternalNo：String，0015文件-卡片内部编号（Hex，8B）  0015StartDate：String，0015文件-启用时间（Hex，4B，格式YYYYMMDD）  0015EndDate：String，0015文件-到期时间（Hex，4B，格式YYYYMMDD）  0015PlateNo：String，0015文件-车牌号码（Hex，12B）  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  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  0015VehicleType：String，0015文件-车型（Hex，1B） |
| 返回值 | 无 | |
| 备注 |  | |

##### 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 | 无 | |
| 备注 |  | |

### 3.11 设备证书管理模块（TOSS不支持）

#### 3.11.1 获取证书和私钥状态

|  |  |  |
| --- | --- | --- |
| 原型 | int getDevKeyState(int certIndex) | |
| 功能 | 获取指定索引的证书和私钥状态 | |
| 参数 | certIndex[in] | 证书索引，范围：参考附录：[密钥体系及密钥索引范围.SEC\_DEVICE\_CERT](#密钥体系及密钥索引范围_SEC_DEVICE_CERT) |
| 返回值 | 0-正常  <0-错误码 | |
| 备注 |  | |

#### 3.11.2 生成设备证书公私钥数据

|  |  |  |
| --- | --- | --- |
| 原型 | int genDevKey(int certIndex, int mode, byte[] dataOut) | |
| 功能 | 生成设备证书公私钥数据 | |
| 参数 | certIndex[in] | 证书索引，范围：参考附录：[密钥体系及密钥索引范围.SEC\_DEVICE\_CERT](#密钥体系及密钥索引范围_SEC_DEVICE_CERT) |
| mode[in] | 模式，参见参考附录：Aidl常量.[生成设备证书模式](#生成设备证书模式) |
| dataOut[out] | Buffer，存放公钥模 |
| 返回值 | >=0-dataOut中有效数据的长度  <0-错误码 | |
| 备注 | 1.本接口保存的证书永久性存储，设备关机/重启、APP重启后密钥不丢失  2.客户端App可调用[deleteKey()](#删除公钥证书和设备私钥)接口删除已保存的证书数据 | |

#### 3.11.3 保存设备证书数据

|  |  |  |
| --- | --- | --- |
| 原型 | int saveDevCert(int certIndex, byte[] certData) | |
| 功能 | 保存设备证书数据 | |
| 参数 | certIndex[in] | 证书索引，范围：参考附录：[密钥体系及密钥索引范围.SEC\_DEVICE\_CERT](#密钥体系及密钥索引范围_SEC_DEVICE_CERT) |
| certData[in] | 设备证书数据 |
| 返回值 | 0-成功  <0-错误码 | |
| 备注 | 1.本接口保存的证书永久性存储，设备关机/重启、APP重启后密钥不丢失  2.客户端App可调用[deleteKey()](#删除公钥证书和设备私钥)接口删除已保存的证书数据 | |

#### 3.11.4 删除公钥证书和设备私钥

|  |  |  |
| --- | --- | --- |
| 原型 | int deleteKey(int certIndex) | |
| 功能 | 删除公钥证书和设备私钥 | |
| 参数 | certIndex[in] | 证书索引，范围：参考附录：[密钥体系及密钥索引范围.SEC\_DEVICE\_CERT](#密钥体系及密钥索引范围_SEC_DEVICE_CERT) |
| 返回值 | 0-成功  <0-错误码 | |
| 备注 |  | |

#### 3.11.5 指定包名获取设备证书

|  |  |  |
| --- | --- | --- |
| 原型 | int getDeviceCertificateEx(Bundle bundle, byte[] dataOut) | |
| 功能 | 指定包名获取保存的设备证书 | |
| 参数 | bundle[in] | 入参，包含如下key：  targetPkgName：String，目标APP的包名，不可为null  certIndex：int，证书索引，范围：参考附录：[密钥体系及密钥索引范围.SEC\_DEVICE\_CERT](#密钥体系及密钥索引范围_SEC_DEVICE_CERT) |
| dataOut[out] | Buffer，存放证书数据 |
| 返回值 | >=0-dataOut中有效数据的长度  <0：错误码 | |
| 备注 |  | |

### 3.12 HCE模块（TOSS不支持）

#### 3.12.1 打开HCE

|  |  |  |
| --- | --- | --- |
| 原型 | int hceOpen(int cardType) | |
| 功能 | 打开HCE | |
| 参数 | cardType[in] | 卡类型，2-NFC tag2卡，4-NFC FORUM T4T卡 |
| 返回值 | 0-成功  <0-错误码 | |
| 备注 | 1.本方法目前仅在P3、P3K机型上支持，在其他机型上不支持 | |

#### 3.12.2 打开HCE（重载）

|  |  |  |
| --- | --- | --- |
| 原型 | int hceOpen(int cardType, byte[] param) | |
| 功能 | 打开HCE | |
| 参数 | cardType[in] | 卡类型：2-NFC tag2卡，4-NFC FORUM T4T卡 |
| param[in] | 卡类型相关的参数，长度<=225B，可为null |
| 返回值 | 0-成功  <0-错误码 | |
| 备注 | 1.本方法目前仅在P3、P3K机型上支持，在其他机型上不支持 | |

#### 3.12.3 HCE写NDEF数据

|  |  |  |
| --- | --- | --- |
| 原型 | int hceNdefWrite(NdefMessage msg) | |
| 功能 | HCE写NDEF数据 | |
| 参数 | msg[in] | NDEF数据，格式为NDefMessage格式，tag2数据长度<=399B，t4t数据长度<=1024B |
| 返回值 | 0-成功  <0-错误码 | |
| 备注 | 1.本方法目前仅在P3、P3K机型上支持，在其他机型上不支持 | |

#### 3.12.4 HCE写数据

|  |  |  |
| --- | --- | --- |
| 原型 | int hceWrite(byte[] msg) | |
| 功能 | HCE写数据 | |
| 参数 | msg[in] | 要写入的数据，当前仅支持写入NDefMessage格式的数据，t4t数据长度<=399B，tag4数据长度<=1024B |
| 返回值 | 0-成功  <0-错误码 | |
| 备注 | 1.本方法目前仅在P3、P3K机型上支持，在其他机型上不支持 | |

#### 3.12.5 HCE读NDEF数据

|  |  |  |
| --- | --- | --- |
| 原型 | NdefMessage hceNdefRead() | |
| 功能 | HCE读NDEF数据 | |
| 参数 | [in] |  |
| 返回值 | 成功-返回NDefMessage对象  失败-返回null | |
| 备注 | 1.本方法目前仅在P3、P3K机型上支持，在其他机型上不支持 | |

#### 3.12.6 HCE读数据

|  |  |  |
| --- | --- | --- |
| 原型 | int hceRead(byte[] outData) | |
| 功能 | HCE读数据 | |
| 参数 | outData[out] | Buffer，存放读取到的数据，长度>=1024B，当前仅返回NDefMessage格式的数据 |
| 返回值 | >=0-outData中有效数据的长度  <0-错误码 | |
| 备注 | 1.本方法目前仅在P3、P3K机型上支持，在其他机型上不支持 | |

#### 3.12.7 关闭HCE

|  |  |  |
| --- | --- | --- |
| 原型 | int hceClose() | |
| 功能 | 关闭HCE，关闭后模拟卡下电 | |
| 参数 | [in] |  |
| 返回值 | 0-成功  <0-错误码 | |
| 备注 | 1.本方法目前仅在P3、P3K机型上支持，在其他机型上不支持 | |

## 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 |
| -2401 | HCE参数错误 |
| -2402 | HCE模块初始化失败 |
| -2403 | HCE模块未打开 |
| -2404 | HCE功能尚未支持 |
| -2405 | HCE t4t操作失败 |
| -2406 | HCE NDEF数据无变化 |
| -2407 | HCE状态错误 |
| -2408 | HCE t2t操作失败 |
| -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 |
| -2530 | 卡片拒绝命令 |
| -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 |
| -2549 | A卡不支持ISO14443-4，激活流程终止 |
| -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 |
| -2904 | 复位失败 |
| -2909 | 验证密码失败 |
| -2911 | 卡片响应失败 |
| -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 |
| -3063 | TLV参数错误 |
| -3064 | ECC算法错误 |
| -3065 | RNDKEY错误 |
| -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 |
| -4003 | 查看手机 |
| -4006 | 最终选择数据错误（MIR） |
| -4008 | 使用另一张卡 |
| -4009 | 请重试（JCB） |
| -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 |
| -7009 | 配置不支持打印机 |
| -7010 | 打印机不支持此功能 |
| -7011 | 打印机仓门打开 |
| -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 |
| -16000 | LITESO空间不够 |
| -16001 | LITESO长度错误 |
| -16002 | 安装LITESO个数超过系统最大值 |
| -16003 | LITESO签名错误 |
| -16004 | LITESO指纹错误 |
| -16005 | LITESO写flash错误 |
| -16006 | LITESO info错误 |
| -16007 | LITESO无文件访问权限 |
| -16008 | LITESO不存在 |
| -20000 | 功能尚不支持 |
| -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 |
| -40020 | transformation不支持 |
| -40021 | 密钥未保存 |
| -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 |
| -50029 | 数据交换出错 |
| -50030 | 数据交换超时 |
| -50031 | 终端风险管理超时 |
| -50032 | 终端风险管理出错 |
| -50033 | 第一次GAC前回调超时 |
| -50034 | 第一次GAC前回调出错 |
| -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** |
| -80001 | 输PIN进行中 |
| -90001 | 未搜索到ETC设备 |

## 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）  public byte extSelectSupFlg;//扩展选择支持标志DFC10D(00-不支持，01-支持) |

### 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](#EMVFlowType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

|  |
| --- |
| Private int pinpadType;//密码键盘类型, 0-预置普通键盘(默认值) ，1-自定义普通键盘，2-预置盲人模式键盘，3-预置rnib认证盲人模式键盘，4-预置rnib认证普通键盘，5-自定义盲人模式键盘  Private int pinType=0;//pin类型标识(0是联机pin，1是脱机pin)  Private boolean isOrderNumKey=false;//true:顺序键盘false:乱序键盘  Private byte[] pan;//ascii格式转换成的byte例如“123456”.getbytes("usascii"),长度12~19位；  Private int pinkeyIndex;//pik索引(pin密钥索引)  Private int maxInput=6;//最大输入位数(最多允许输入12位)  Private int minInput=0;//最小输入位数  Private int timeout=60000;//超时时间/毫秒  Private Boolean isSupportbypass=true;//是否支持bypasspin  Private int pinblockFormat=0;//pinblock格式，参考[PinBlock格式](#PinBlock格式)  Private int algorithmType=0;//加密Pin的算法类型0-3DES(返回8字节),1-SM4（返回16字节）  Private int keySystem=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](#密钥体系常量)  **说明：**  **1. 当前不支持pinPadType为3、4类型** |

### 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 PinPadDataV2- 密码键盘UI布局数据

|  |
| --- |
| public int numX; //密码键盘第一个按键x轴坐标（绝对坐标）  public int numY; //密码键盘第一个按y轴坐标（绝对坐标）  public int numH; //密码键盘第一个按键高  public int numW; //密码键盘第一个按键宽  public int lineW; //间隔线宽（若无，传0即可）  public int cancelX; //密码键盘取消键x轴坐标（绝对坐标）  public int cancelY; //密码键盘取消键y轴坐标（绝对坐标）  public int cancelH; //密码键盘取消键高  public int cancelW; //密码键盘取消键宽  public int rows; //行数  public int clos; //列数  public byte[] keyMap = new byte[64]; //键盘按键映射关系 |

### 5.9 PinPadDataV2Ex- 密码键盘UI布局数据（扩展）

|  |
| --- |
| public int numX; //密码键盘第一个按键x轴坐标（绝对坐标）  public int numY; //密码键盘第一个按y轴坐标（绝对坐标）  public int numH; //密码键盘第一个按键高  public int numW; //密码键盘第一个按键宽  public int lineW; //间隔线宽（若无，传0即可）  public int cancelX; //密码键盘取消键x轴坐标（绝对坐标）  public int cancelY; //密码键盘取消键y轴坐标（绝对坐标）  public int cancelH; //密码键盘取消键高  public int cancelW; //密码键盘取消键宽  public int enterX; //密码键盘确认键x轴坐标（绝对坐标）  public int enterY; //密码键盘确认键y轴坐标（绝对坐标）  public int enterH; //密码键盘确认键高  public int enterW; //密码键盘确认键宽  public int clearX; //密码键盘清除键x轴坐标（绝对坐标）  public int clearY; //密码键盘清除键y轴坐标（绝对坐标）  public int clearH; //密码键盘清除键高  public int clearW; //密码键盘清除键宽  public int rows; //行数  public int clos; //列数  public byte[] keyMap = new byte[64]; //键盘按键映射关系 |

### 5.10 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} |

### 5.11 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 |

### 5.12 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;  // SAM2卡  public static final int SAM2= 1<<23;  // SAM3卡  public static final int SAM3= 1<<24;  // SRI卡  public static final int SRI= 1<<25;  // SAM4卡  public static final int SAM4= 1<<26;  // SAM5卡  public static final int SAM5= 1<<27;  // ISO15693卡  public static final int ISO15693= 1<<28; |

#### 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;  // 密钥类型，RSA KPK  public static final int KEY\_TYPE\_RSA\_KPK = 0x0B;  // 密钥类型，RSA KEK  public static final int KEY\_TYPE\_RSA\_KEK = 0x0C; |

#### 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;  public static final int MAC\_ALG\_CUP\_SM4\_MAC\_ALG2 = 3004;  public static final int MAC\_ALG\_X9\_19\_DEA = 3005;  // HMAC-SHA1  public static final int MAC\_ALG\_HMAC\_SHA1 = 3006;  // HMAC-SHA256  public static final int MAC\_ALG\_HMAC\_SHA256 = 3007;  // CMAC  public static final int MAC\_ALG\_CMAC = 3008;  // 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; |

#### 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";  注：NONEwithRSA签名模式输入数据长度需小于模长 |

#### 7.1.11 RSA填充模式

|  |
| --- |
| public static final int NOTHING\_PADDING = 0;  public static final int PKCS1\_PADDING = 1;  public static final int PKCS7\_PADDING = 2;  public static final int PKCS5\_PADDING = 3;  public static final int PKCS1\_OAEP\_PADDING = 4;  public static final int PKCS1\_V1\_5\_SHA512 = 5;  public static final int PADDING\_OAEP\_SHA1 = 6; |

#### 7.1.12 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; |

#### 7.1.13 密钥操作类型

|  |
| --- |
| public static final int KEY\_CTRL\_PANPARA = 0x41;  public static final int KEY\_CTRL\_AUTHPARA = 0x42;  public static final int KEY\_CTRL\_APACSMAC = 0x43; |

#### 7.1.14 注入对称密钥模式

|  |
| --- |
| public static final int INJECT\_OAEP\_MODE = 0x05;//OAEP模式注入密钥，依赖密钥是设备证书私钥  public static final int INJECT\_PKCS1\_MODE = 0x06;//PKCS1模式注入密钥，依赖密钥是RSA私钥  public static final int INJECT\_DERIVER\_OWF2 = 0x80;//OWF2算法类型派生并保存密钥  public static final int INJECT\_DERIVER\_OWF3 = 0x81;//OWF3算法类型派生并保存密钥  public static final int INJECT\_DERIVER\_GOWF = 0x82;//GOWF算法类型派生并保存密钥  public static final int INJECT\_DERIVER\_ENC = 0x83; |

#### 7.1.15 生成设备证书模式

|  |
| --- |
| public static final int CERT\_GENERATE\_RSA2048\_E65537\_PVK\_PUK = 0x82; |

#### 7.1.16 ECC密钥协商模式

|  |
| --- |
| public static final int SEC\_ECDH\_STEP1\_MODE = 2; //ECDH密钥协商第一步（仅支持mksk）  public static final int SEC\_ECDH\_STEP2\_MODE = 3; //ECDH密钥协商第二步（仅支持mksk） |

#### 7.1.17 ECC曲线参数

|  |
| --- |
| public static final String SEC\_ECC\_PARAM\_P256 = "P-256";  public static final String SEC\_ECC\_PARAM\_P384 = "P-384";  public static final String SEC\_ECC\_PARAM\_P521 = "P-521"; |

#### 7.1.18 kcv模式

|  |
| --- |
| public static final int KCV\_MODE\_NOCHK = 0;  public static final int KCV\_MODE\_CHK0 = 1;  public static final int KCV\_MODE\_CHKFIX = 2;  public static final int KCV\_MODE\_CHKMAC = 3;  public static final int KCV\_MODE\_CHKCMAC = 4; |

#### 7.1.19 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; |

#### 7.1.20 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; |

#### 7.1.21 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; |

#### 7.1.22 EMV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; |

#### 7.1.23 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;  /\*\* CPACE \*/  public static final int OP\_CPACE = 13;  // 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; |

#### 7.1.24 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;  //SAMSUNGPAY  public static final int SAMSUNGPAY = 13;  //CPACE  public static final int CPACE = 15; |

#### 7.1.25 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; |

#### 7.1.26 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; |

#### 7.1.27 EMV联机结果定义

|  |
| --- |
| // 联机批准  public static final int ONLINE\_APPROVAL = 0;  // 联机拒绝  public static final int ONLINE\_DECLINE = 1;  // 联机失败  public static final int ONLINE\_FAIL = 2;  // 联机批准与二次拍卡  public static final int ONLINE\_APPROVAL\_2\_TAP = 3;  // 联机拒绝与二次拍卡  public static final int ONLINE\_DECLINE\_2\_TAP = 4;  // 联机失败与二次拍卡  public static final int ONLINE\_FAIL\_2\_TAP = 5;  // 脱机拒绝  public static final int OFFLINE\_DECLINE = 6;  // 联机失败与Full online  public static final int ONLINE\_FAIL\_FULL\_ONLINE = 7;  // 联机成功，8A存在，且不存在脚本  public static final int ONLINE\_SUCCESS\_NO\_SCRIPT = 8;  // AE 后台要求输PIN失败  public static final int ONLINE\_FAIL\_ONLINEPIN = 9; |

#### 7.1.28 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; |

#### 7.1.29 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; |

#### 7.1.30 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";  //获取设备上SAM卡槽个数  public static final String SAM\_COUNT = "SAM";  //获取libicsam.a版本号  public static final String IFM\_LIB\_VERSION = "IfmLibVersion";  //获取厂商磁条卡解码库版本号  public static final String MSR\_VERSION = "MsrVersion";  //获取sunmiposapi.jar版本号  public static final String POSAPI\_VERSION = "posapiVersion";  //获取是否支持检测RTC电池电，0-不支持，1-支持  public static final String RTC\_BAT\_VOL\_DET = "RTCBATVOLDET";  //是否支持sred，0-不支持，1-支持  public static final String SRED = "sred";  //PCIPTS版本  public static final String PCI\_PTS\_VERSION = "PCIPTSVersion";  //RNIB认证版本  public static final String RNIB\_VERSION = "RNIBVersion";  // 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  public static final String FLASH\_VERSION = "FLASHVersion";  // DPAS版本  public static final String DPAS\_VERSION = "DPASVersion";  // APEMV版本  public static final String APEMV\_VERSION = "APEMVVersion";  // EFTPOS版本  public static final String EFTPOS\_VERSION = "EFTPOSVersion";  // EMVBase版本  public static final String EMVBASE\_VERSION = "EMVBaseVersion";  // KernelDirect版本  public static final String KD\_VERSION = "KDVersion";  // RUPAY版本  public static final String RUPAY\_VERSION = "RUPAYVersion";  // SAMSUNGPAY版本  public static final String SAMSUNGPAY\_VERSION = "SAMSUNGPAYVersion";  // CPACE版本  public static final String CPACE\_VERSION = "CPACEVersion";  // EMV kernel checksum信息  public static final String EMV\_KERNEL\_CHECKSUM = "EmvKernelCheckSum";  // Pure发布日期  public static final String PURE\_RELEASE\_DATE = "PUREReleaseDate";  // EFTPOS发布日期  public static final String EFTPOS\_RELEASE\_DATE = "EFTPOSReleaseDate";  // EMV发布日期  public static final String EMV\_RELEASE\_DATE = "EMVReleaseDate";  // Paypass发布日期  public static final String PAYPASS\_RELEASE\_DATE = "PaypassReleaseDate";  // Paywave发布日期  public static final String PAYWAVE\_RELEASE\_DATE = "PaywaveReleaseDate";  // QPBOC发布日期  public static final String QPBOC\_RELEASE\_DATE = "QPBOCReleaseDate";  // Entry发布日期  public static final String ENTRY\_RELEASE\_DATE = "EntryReleaseDate";  // Mir发布日期  public static final String MIR\_RELEASE\_DATE = "MirReleaseDate";  // JCB发布日期  public static final String JCB\_RELEASE\_DATE = "JCBReleaseDate";  // Pago发布日期  public static final String PAGO\_RELEASE\_DATE = "PAGOReleaseDate";  // AE发布日期  public static final String AE\_RELEASE\_DATE = "AEReleaseDate";  // FLASH发布日期  public static final String FLASH\_RELEASE\_DATE = "FLASHReleaseDate";  // DPAS发布日期  public static final String DPAS\_RELEASE\_DATE = "DPASReleaseDate";  // EMVBase发布日期  public static final String EMVBASE\_RELEASE\_DATE = "EMVBaseReleaseDate";  // KernelDirect发布日期  public static final String KD\_RELEASE\_DATE = "KDReleaseDate";  // Rupay发布日期  public static final String RUPAY\_RELEASE\_DATE = "RUPAYReleaseDate";  // SAMSUNGPAY发布日期  public static final String SAMSUNGPAY\_RELEASE\_DATE = "SAMSUNGPAYReleaseDate";  // CPACE发布日期  public static final String CPACE\_RELEASE\_DATE = "CPACEReleaseDate"; |

#### 7.1.31 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;  // 白灯  public static final int WHITE\_LIGHT = 5;  // 四角灯-红（P3\_MIX，P3支持）  public static final int CORNER\_RED\_LIGHT = 6;  // 四角灯-绿（P3\_MIX，P3支持）  public static final int CORNER\_GREEN\_LIGHT = 7;  // 四角灯-蓝（P3\_MIX，P3支持）  public static final int CORNER\_BLUE\_LIGHT = 8;  // 指示灯-黄色（P3\_MIX支持）  public static final int INDICATOR\_YELLOW\_LIGHT = 9; |

#### 7.1.32 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"; |

#### 7.1.33 PinBlock格式

|  |
| --- |
| //支持DES、TDES、SM4密钥，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密钥，PAN无意义  public static final int SEC\_PIN\_BLK\_ISO\_FMT1 = 1;  //不支持的PinBlock格式  public static final int SEC\_PIN\_BLK\_ISO\_FMT2 = 2;  //支持DES、TDES密钥，输入格式化后的12位ASCII PAN  public static final int SEC\_PIN\_BLK\_ISO\_FMT3 = 3;  //支持DES、TDES密钥，PAN为6为ASCII流水号  public static final int SEC\_PIN\_BLK\_EPS = 4;  //不支持的PinBlock格式  public static final int SEC\_PIN\_BLK\_IBM\_3621 = 5;  //不支持的PinBlock格式  public static final int SEC\_PIN\_BLK\_IBM\_3624 = 6;  //仅支持AES密钥，输入格式化后的12-19位ASCII PAN  public static final int SEC\_PIN\_BLK\_ISO\_FMT4 = 7; |

#### 7.1.34 键盘按键音常量定义

|  |
| --- |
| //打开按键音（默认）  public static final String MODE\_ON = "ON";  //关闭按键音  public static final String MODE\_OFF = "OFF"; |

#### 7.1.35 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;  //打印机开盖  public static final int PRI\_CAP\_OPEN = 6; |

#### 7.1.36 卡片在位状态

|  |
| --- |
| // 卡片不在位  public static final int CARD\_ABSENT = 0x01;  // 卡片在位  public static final int CARD\_PRESENT = 0x02; |

### PAN数据截取

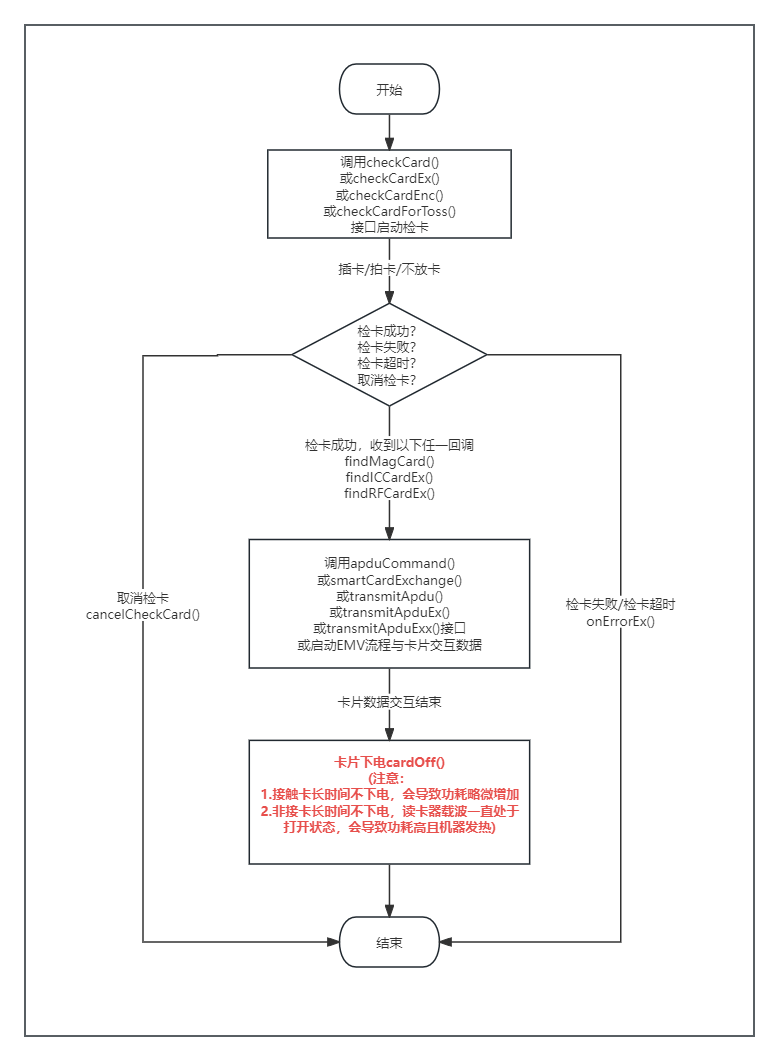
1. 从卡号（2域）右边倒数第二位开始，从右向左的连续12位数字为PAN数据，PAN数据用于PIN加密（计算PinBlock）、解密。

2. 例如卡号为 6225882145611077，则截取后的PAN数据为 588214561107，转换成byte[]数组为(Java代码)：

byte[] pan = “588214561107”.getBytes(“US-ASCII”);

3. 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，客户端应该检查PAN数据是否按照上述规则传入，或者输入的PIN数字是否正确。

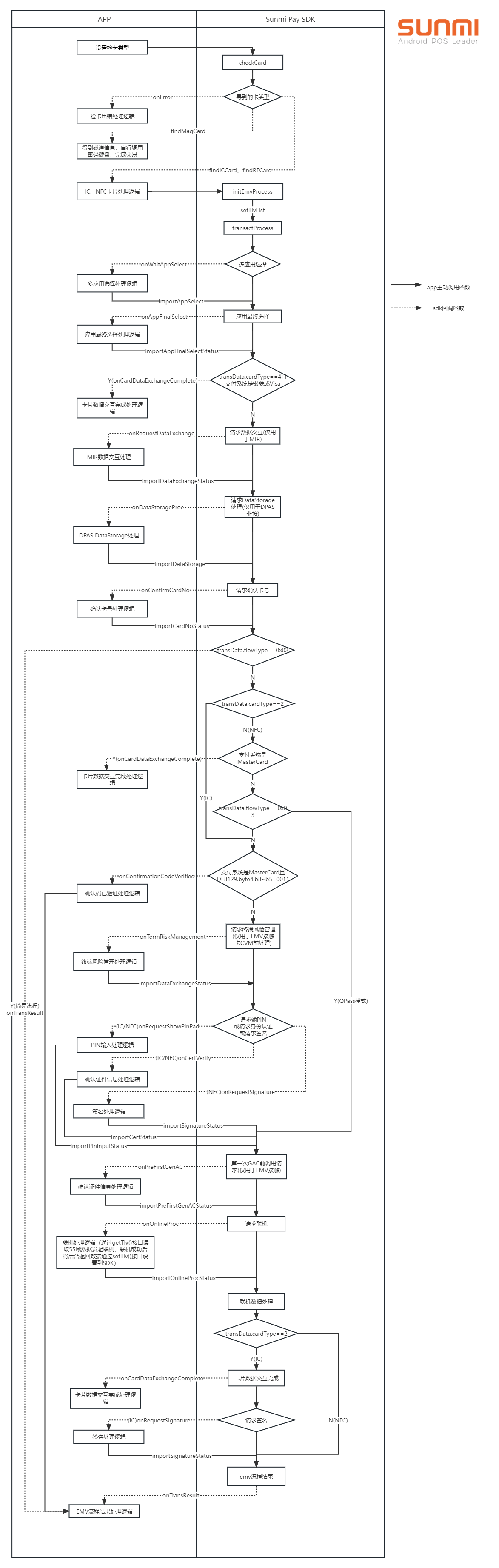
### 7.3 读卡流程图

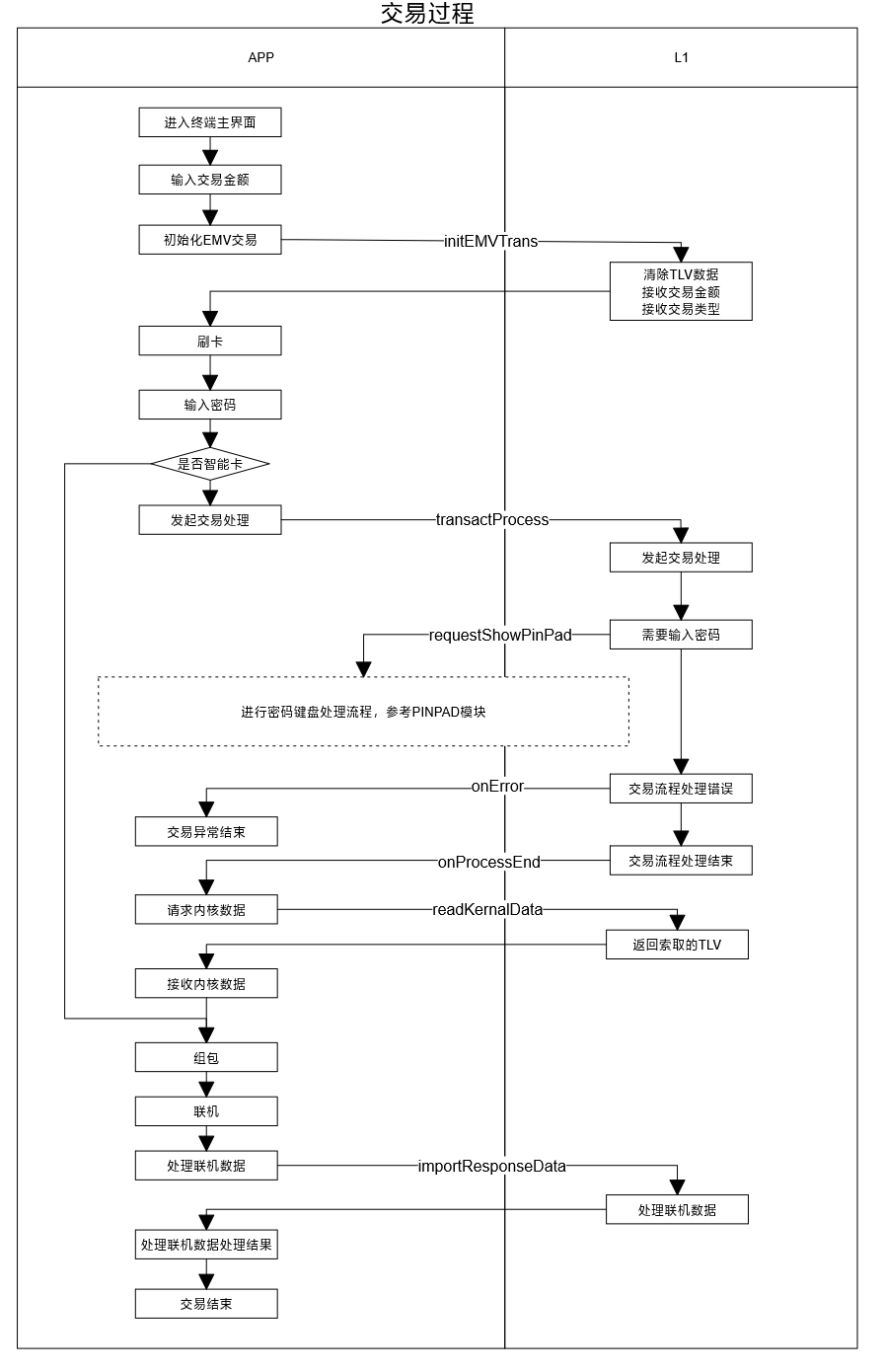
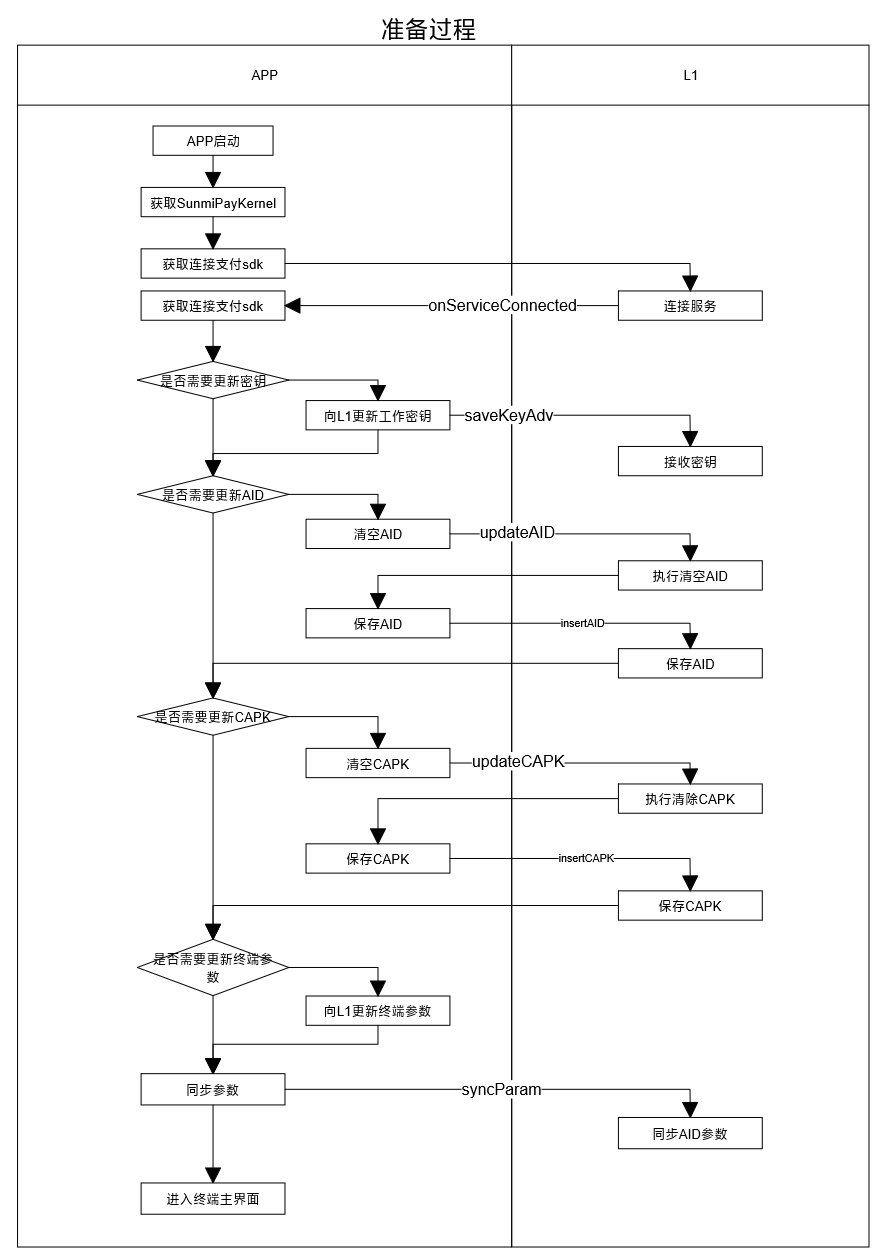


### 7.4 EMV交易流程图

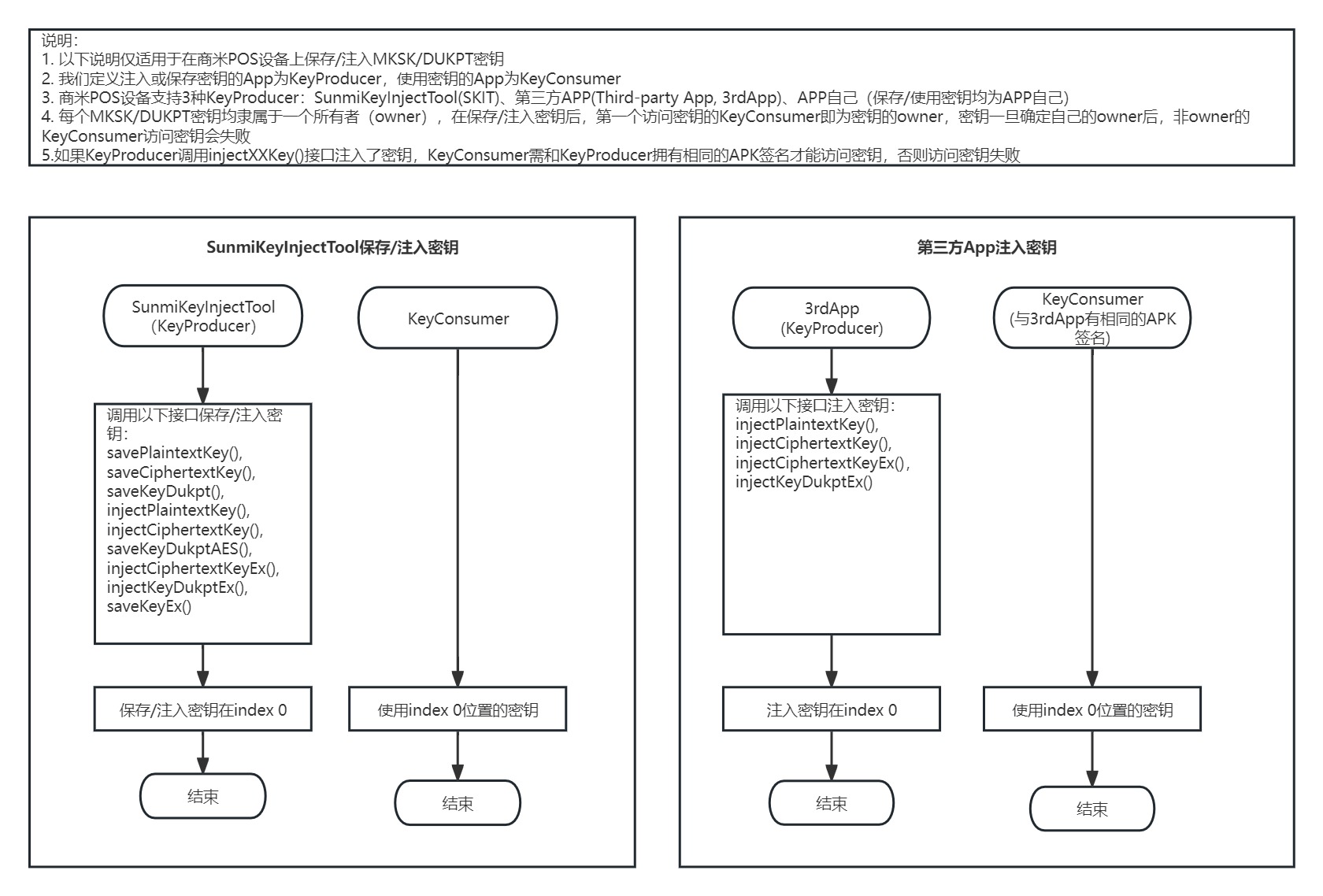
下面的流程图对EMV流程SDK与应用交互进行了描述，请按照流程进行开发







### 7.5 密钥注入/使用流程图



### 7.6 密钥体系及密钥索引范围

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 密钥体系 | 密钥算法类型 | 是否开启密钥分区 | 是否巴西CKD机型 | 普通密钥索引范围 |
| SEC\_MKSK | -- | 是 | -- | 0-199 |
| -- | 否 | -- | 10001-12000 |
| SEC\_DUKPT | 3DES | 是 | 是 | 0-99 |
| 否 | 0-9, 1100-1199 |
| 否 | -- | 20001-21000 |
| AES | 是 | 是 | 100-199 |
| 否 | 10-19, 2100-2199 |
| 否 | -- | 30001-31000 |
| SEC\_RSA\_KEY | -- | 是 | -- | 0-19，20-39 |
| -- | 否 | -- | 0-19, 35001-35200 |
| SEC\_SM2\_KEY | -- | -- | -- | 0-9 |
| -- | -- | -- | 0-9 |
| SEC\_ECC\_KEY | -- | 是 | -- | -- |
| -- | 否 | -- | -- |
| SEC\_DEVICE\_CERT | -- | 是 | -- | 9001-9008 |
| -- | 否 | -- | 9001-9008 |
| 说明：  1.SM2密钥不支持密钥分区  2.无论打开/关闭密钥分区，generateRSAKeypair()、injectRSAKey()接口索引范围始终为0-19，且保存的密钥不支持密钥分区  3.generateRSAKeypairEx()、injectRSAKeyEx()开启密钥分区时索引范围为20-39，关闭分区时索引范围为35001-35200  4.其他使用RSA密钥的接口密钥索引范围如表SEC\_RSA\_KEY部分所示 | | | | |

### 7.7 密钥分区说明

1. 商米支付SDK支持密钥分区功能，并默认开启密钥分区。客户端可根据需求打开/关闭密钥分区，**切换密钥分区后可能导致密钥丢失，建议不要随意切换密钥分区开关，使用SDK默认值即可**

2. 开启/关闭密钥分区后各密钥体系的密钥索引(keyIndex)范围参考[【7.4 密钥体系及密钥索引范围】](#密钥体系及密钥索引范围)

3. 开启密钥分区时，SDK提供密钥隔离功能，APP A保存的密钥仅能APP A访问，APP B不能访问

4. 关闭密钥分区时，SDK不提供密钥隔离功能，APP A保存的密钥可以被APP B访问，存在密钥被修改、删除、覆盖的风险

5. 截止目前，SM2密钥体系无密钥分区功能，APP A保存的密钥可以被APP B访问，存在密钥被修改的风险

### 7.8 KCV参数说明

1. 如果kcv为null，则kcv模式默认为KCV\_ MODE\_ NOCHK，SDK不校验kcv

2. 如果kcv非null且未设置kcvMode参数，则默认的kcvMode是KCV\_ MODE\_ CHK0，kcv按照以下规则计算：

a. DES/TDES密钥加密8字节的0x00数据，密文的前4字节为kcv

b. AES密钥加密16字节的0x00数据，密文的前4字节为kcv

c. SM4密钥加密16字节的0x00数据，生成16字节密文为kcv

3. 如果kcv非null且设置了kcvMode参数，则kcv按照kcvMode指定的方式生成

4. 如果参数kcv的长度大于4或16字节，SDK将截断kcv

5. 对于DES/TDES密钥，kcv长度<=8字节，对于AES/SM4密钥，kcv长度<=16字节

### 7.9 SRED说明

1. SRED（Secure Reading and Exchange of Data）是PCI PTS POI规范中安全要求之一，确保终端读取卡片账户相关数据后立即对其加密和脱敏处理，并在传输过程中保持加密状态，以减少数据泄露的风险。参见以下规范的第B10、B23章节：<https://docs-prv.pcisecuritystandards.org/PTS/Derived%20Test%20Requirements/PCI_PTS_POI_DTRs_v6.2.pdf>

2. 对于商米终端，开启/关闭sred有以下影响：

（1）开启sred后，不允许应用通过SDK接口获取到账户数据明文，包括checkCard()、checkCardEx()、 getTlv()、getTlvList()、onConfirmCardNo()等，仅允许checkCardEnc()和getAccountSecData()获取账号密文或截断数据。

（2）关闭sred后，允许应用通过SDK接口获取到完整的账户数据明文，包括checkCard()、checkCardEx()、getTlv()、getTlvList()、onConfirmCardNo()等。

3. 调用EMVOptV2.setAccountDataSecParam()接口，参数bundle中不包含关键字sred或包含sred且值为true会打开sred，示例代码：

Bundle bundle = new Bundle();

bundle.putInt("encKeySystem", Security.SEC\_MKSK);

bundle.putInt("encKeyIndex", KEY\_INDEX\_MKSK);

bundle.putInt("encMode", Security.DATA\_MODE\_CBC);

bundle.putByteArray("encIv", new byte[16]);

//设置sred为true或不设置sred将打开sred功能

bundle.putBoolean("sred", true);

int code = MyApplication.app.emvOptV2.setAccountDataSecParam(bundle);

Log.e(TAG, "setAccountDataSecParam() with mksk ,code:" + code);

4.从v5.0.34版本开始SRED功能由全局模式改为应用相关模式，APP1开启SRED后仅影响App1，对其他App无影响

### 7.10 EMV默认数据

#### 7.10.1 EMV默认AID

|  |  |  |
| --- | --- | --- |
| NO. | AID | 适用卡品牌 |
| 1 | 9F0608A000000333010100DF0101009F08020020DF1105D84000A800DF1205D84004F800DF130500100000009F1B0400000000DF150400000000DF160199DF170199DF14039F3704DF180101DF2006000200000000DF2106000200000000DFC10B0100 | UnionPay |
| 2 | 9F0608A000000333010101DF0101009F08020020DF1105D84000A800DF1205D84004F800DF130500100000009F1B0400000000DF150400000000DF160199DF170199DF14039F3704DF180101DF2006000200000000DF2106000200000000DFC10B0100 | UnionPay |
| 3 | 9F0608A000000333010102DF0101009F08020020DF1105D84000A800DF1205D84004F800DF130500100000009F1B0400000000DF150400000000DF160199DF170199DF14039F3704DF180101DF2006000200000000DF2106000200000000DFC10B0100 | UnionPay |
| 4 | 9F0608A000000333010103DF0101009F08020020DF1105D84000A800DF1205D84004F800DF130500100000009F1B0400000000DF150400000000DF160199DF170199DF14039F3704DF180101DF2006000200000000DF2106000200000000DFC10B0100 | UnionPay |
| 5 | 9F0608A000000333010107DF0101009F08020020DF1105D84000A800DF1205D84004F800DF130500100000009F1B0400000000DF150400000000DF160199DF170199DF14039F3704DF180101DF2006000200000000DF2106000200000000DFC10B0100 | UnionPay |
| 6 | 9F0607A0000000033010DF0101009F08020140DF1105D84000A800DF1205D84004F800DF130500100000009F1B0400000000DF150400000000DF160199DF170199DF14039F3704DF180100DF2006000200000000DF2106000200000000DFC10B0100 | Visa |
| 7 | 9F0607A0000000032010DF0101009F08020140DF1105D84000A800DF1205D84004F800DF130500100000009F1B0400000000DF150400000000DF160199DF170199DF14039F3704DF180100DF2006000200000000DF2106000200000000DFC10B0100 | Visa |
| 8 | 9F0607A0000000031010DF0101009F08020140DF1105D84000A800DF1205D84004F800DF130500100000009F1B0400000000DF150400000000DF160199DF170199DF14039F3704DF180100DF2006000200000000DF2106000200000000DFC10B0100 | Visa |
| 9 | 9F0607A0000000043060DF0101009F08020002DF1105FC5058A000DF1205F85058F800DF130504000000009F1B0400000000DF150400000000DF160199DF170199DF14039F3704DF180101DF2006000200000000DF2106000200000000DFC10B0100 | MasterCard |
| 10 | 9F0607A0000000041010DF0101009F08020002DF1105FC5080A000DF1205F85080F800DF130504000000009F1B0400000000DF150400000000DF160199DF170199DF14039F3704DF180100DF2006000200000000DF2106000200000000DFC10B0100 | MasterCard |
| 11 | 9F0608A000000333010106DF0101009F08020020DF1105D84000A800DF1205D84004F800DF130500100000009F1B0400000000DF150400000000DF160199DF170199DF14039F3704DF180101DF2006000200000000DF2106000200000000DFC10B0100 | UnionPay |
| 12 | 9F0607A0000000651010DF0101009F08020200DF1105FC6024A800DF1205FC60ACF800DF130500100000009F1B0400000000DF150400000000DF160199DF170199DF14039F3704DF180100DF2006000200000000DF2106000200000000DFC10B0100 | JCB |
| 13 | 9F0607A0000000031010DF0101009F0902008CDF1105D84000A800DF1205D84004F800DF130500100000009F1B0400000000DF150400000000DF160100DF170100DF180101DF2106000000005001DF2006000000006000DF1906000000005000DFC10B0100 | Visa |
| 14 | 9F0607A0000000032010DF0101009F0902008CDF1105D84000A800DF1205D84004F800DF130500100000009F1B0400000000DF150400000000DF160100DF170100DF180101DF2106000000010000DF2006000000100000DF1906000000000010DFC10B0100 | Visa |
| 15 | 9F0607A0000000041010DF0101009F09020002DF1105FC50A8A000DF1205F850A8F800DF130504000000009F1B0400000000DF150400000000DF160100DF170100DF180101DF2106000000010000DF2006000000100000DF1906000000000010DFC10B0100 | MasterCard |
| 16 | 9F0607A0000000043060DF0101009F09020002DF1105FC50A8A000DF1205F850A8F800DF130504000000009F1B0400000000DF150400000000DF160100DF170100DF180101DF2106000000010000DF2006000000100000DF1906000000000010DFC10B0100 | MasterCard |
| 17 | 9F0607A0000000046000DF0101009F09020002DF1105FC50A8A000DF1205F850A8F800DF130504000000009F1B0400000000DF150400000000DF160100DF170100DF180101DF2106000000010000DF2006000000100000DF1906000000000010DFC10B0100 | MasterCard |
| 18 | 9F0607A00000000410109F09020000DF11050000000000DF12050000000000DF130500000000009F1B0400000000DF150400000000DF160100DF170100DF140b9F37049F47018F019F32019F3C0206439F3D0102DF1906000000000000DF2006999999999999DF21060000000030009F4E009F150211119F160c746573746D65726368616E74DFC10B0100 | MasterCard |
| 19 | 9F0607A00000000310109F09020000DF11050000000000DF12050000000000DF130500000000009F1B0400000000DF150400000000DF160100DF170100DF14039F37049F3C0206439F3D0102DF1906000000000000DF2006999999999999DF21060000000000009F4E009F150211119F160c746573746D65726368616E74DFC10B0100 | Visa |

#### 7.10.2 EMV默认CAPK

|  |  |  |
| --- | --- | --- |
| NO. | CAPK | 适用卡品牌 |
| 1 | 9F0605A0000003339F220102DF0503211231DF060101DF070101DF028190A3767ABD1B6AA69D7F3FBF28C092DE9ED1E658BA5F0909AF7A1CCD907373B7210FDEB16287BA8E78E1529F443976FD27F991EC67D95E5F4E96B127CAB2396A94D6E45CDA44CA4C4867570D6B07542F8D4BF9FF97975DB9891515E66F525D2B3CBEB6D662BFB6C3F338E93B02142BFC44173A3764C56AADD202075B26DC2F9F7D7AE74BD7D00FD05EE430032663D27A57DF040103DF031403BB335A8549A03B87AB089D006F60852E4B8060 | UnionPay |
| 2 | 9F0605A0000003339F220103DF0503221231DF060101DF070101DF0281B0B0627DEE87864F9C18C13B9A1F025448BF13C58380C91F4CEBA9F9BCB214FF8414E9B59D6ABA10F941C7331768F47B2127907D857FA39AAF8CE02045DD01619D689EE731C551159BE7EB2D51A372FF56B556E5CB2FDE36E23073A44CA215D6C26CA68847B388E39520E0026E62294B557D6470440CA0AEFC9438C923AEC9B2098D6D3A1AF5E8B1DE36F4B53040109D89B77CAFAF70C26C601ABDF59EEC0FDC8A99089140CD2E817E335175B03B7AA33DDF040103DF031487F0CD7C0E86F38F89A66F8C47071A8B88586F26 | UnionPay |
| 3 || UnionPay |
| 4 | 9F0605A0000000039F220147DF028190F562B594C911C0310AEFB002BDFE01CC8C2F1351CAEDD12C5210F4C3507BC106E01BBF94362392F3A66496079CBC8EAD09D5D942B195CA12A8A1AABAEF1D5545AF1693484764DEB625100E15F630510F3E2FB9E00D81F86D4C6F099307182E30F67FBE9C3D91F557BA5AAA1ACFBB625ABFA0176D4CEF3DFE1892BA1EC49F57B5DB6C53147F1DC9FD844298E238A94FEBDF040103DF0503201231 | Visa |
| 5 | 9F0605A0000000039F220101DF028180C696034213D7D8546984579D1D0F0EA519CFF8DEFFC429354CF3A871A6F7183F1228DA5C7470C055387100CB935A712C4E2864DF5D64BA93FE7E63E71F25B1E5F5298575EBE1C63AA617706917911DC2A75AC28B251C7EF40F2365912490B939BCA2124A30A28F54402C34AECA331AB67E1E79B285DD5771B5D9FF79EA630B75DF040103DF0503201231 | Visa |
| 6 | 9F0605A0000000039F220107DF028190A89F25A56FA6DA258C8CA8B40427D927B4A1EB4D7EA326BBB12F97DED70AE5E4480FC9C5E8A972177110A1CC318D06D2F8F5C4844AC5FA79A4DC470BB11ED635699C17081B90F1B984F12E92C1C529276D8AF8EC7F28492097D8CD5BECEA16FE4088F6CFAB4A1B42328A1B996F9278B0B7E3311CA5EF856C2F888474B83612A82E4E00D0CD4069A6783140433D50725FDF040103DF0503201231 | Visa |
| 7 | 9F0605A0000000039F220108DF0281B0D9FD6ED75D51D0E30664BD157023EAA1FFA871E4DA65672B863D255E81E137A51DE4F72BCC9E44ACE12127F87E263D3AF9DD9CF35CA4A7B01E907000BA85D24954C2FCA3074825DDD4C0C8F186CB020F683E02F2DEAD3969133F06F7845166ACEB57CA0FC2603445469811D293BFEFBAFAB57631B3DD91E796BF850A25012F1AE38F05AA5C4D6D03B1DC2E568612785938BBC9B3CD3A910C1DA55A5A9218ACE0F7A21287752682F15832A678D6E1ED0BDF040103DF0503201231 | Visa |
| 8 | 9F0605A0000000039F220105DF0260D0135CE8A4436C7F9D5CC66547E30EA402F98105B71722E24BC08DCC80AB7E71EC23B8CE6A1DC6AC2A8CF55543D74A8AE7B388F9B174B7F0D756C22CBB5974F9016A56B601CCA64C71F04B78E86C501B193A5556D5389ECE4DEA258AB97F52A3DF060101DF070101DF040103DF031486DF041E7995023552A79E2623E49180C0CD957ADF0503151231 | Visa |
| 9 | 9F0605A0000000049F2201FADF0503202001DF060101DF070101DF028190A90FCD55AA2D5D9963E35ED0F440177699832F49C6BAB15CDAE5794BE93F934D4462D5D12762E48C38BA83D8445DEAA74195A301A102B2F114EADA0D180EE5E7A5C73E0C4E11F67A43DDAB5D55683B1474CC0627F44B8D3088A492FFAADAD4F42422D0E7013536C3C49AD3D0FAE96459B0F6B1B6056538A3D6D44640F94467B108867DEC40FAAECD740C00E2B7A8852DDF040103DF03145BED4068D96EA16D2D77E03D6036FC7A160EA99C | MasterCard |
| 10 | 9F0605A0000000049F2201F1DF0503202001DF060101DF070101DF0281b0A0DCF4BDE19C3546B4B6F0414D174DDE294AABBB828C5A834D73AAE27C99B0B053A90278007239B6459FF0BBCD7B4B9C6C50AC02CE91368DA1BD21AAEADBC65347337D89B68F5C99A09D05BE02DD1F8C5BA20E2F13FB2A27C41D3F85CAD5CF6668E75851EC66EDBF98851FD4E42C44C1D59F5984703B27D5B9F21B8FA0D93279FBBF69E090642909C9EA27F898959541AA6757F5F624104F6E1D3A9532F2A6E51515AEAD1B43B3D7835088A2FAFA7BE7DF040103DF0314D8E68DA167AB5A85D8C3D55ECB9B0517A1A5B4BB | MasterCard |
| 11 | | MasterCard |
| 12 | 9F0605A0000000049F220147DF0503202001DF060101DF070101DF028190F562B594C911C0310AEFB002BDFE01CC8C2F1351CAEDD12C5210F4C3507BC106E01BBF94362392F3A66496079CBC8EAD09D5D942B195CA12A8A1AABAEF1D5545AF1693484764DEB625100E15F630510F3E2FB9E00D81F86D4C6F099307182E30F67FBE9C3D91F557BA5AAA1ACFBB625ABFA0176D4CEF3DFE1892BA1EC49F57B5DB6C53147F1DC9FD844298E238A94FEBDF040103DF03146304B43BAF1342F914ABD4B55EE817CEF1FA557F | MasterCard |
| 13 | | MasterCard |
| 14 | | MasterCard |
| 15 | 9F0605A0000000049F220105DF0503202001DF060101DF070101DF0281b0B8048ABC30C90D976336543E3FD7091C8FE4800DF820ED55E7E94813ED00555B573FECA3D84AF6131A651D66CFF4284FB13B635EDD0EE40176D8BF04B7FD1C7BACF9AC7327DFAA8AA72D10DB3B8E70B2DDD811CB4196525EA386ACC33C0D9D4575916469C4E4F53E8E1C912CC618CB22DDE7C3568E90022E6BBA770202E4522A2DD623D180E215BD1D1507FE3DC90CA310D27B3EFCCD8F83DE3052CAD1E48938C68D095AAC91B5F37E28BB49EC7ED597DF040103DF0314EBFA0D5D06D8CE702DA3EAE890701D45E274C845 | MasterCard |
| 16 | 9F0605A0000000039F220195DF0503202001DF060101DF070101DF028190BE9E1FA5E9A803852999C4AB432DB28600DCD9DAB76DFAAA47355A0FE37B1508AC6BF38860D3C6C2E5B12A3CAAF2A7005A7241EBAA7771112C74CF9A0634652FBCA0E5980C54A64761EA101A114E0F0B5572ADD57D010B7C9C887E104CA4EE1272DA66D997B9A90B5A6D624AB6C57E73C8F919000EB5F684898EF8C3DBEFB330C62660BED88EA78E909AFF05F6DA627BDF040103DF0314EE1511CEC71020A9B90443B37B1D5F6E703030F6 | Visa |
| 17 | 9F0605A0000000039F220199DF0503202001DF060101DF070101DF028180AB79FCC9520896967E776E64444E5DCDD6E13611874F3985722520425295EEA4BD0C2781DE7F31CD3D041F565F747306EED62954B17EDABA3A6C5B85A1DE1BEB9A34141AF38FCF8279C9DEA0D5A6710D08DB4124F041945587E20359BAB47B7575AD94262D4B25F264AF33DEDCF28E09615E937DE32EDC03C54445FE7E382777DF040103DF03144ABFFD6B1C51212D05552E431C5B17007D2F5E6D | Visa |
| 18 | 9F0605A0000000039F220150DF0503202001DF060101DF070101DF028180D11197590057B84196C2F4D11A8F3C05408F422A35D702F90106EA5B019BB28AE607AA9CDEBCD0D81A38D48C7EBB0062D287369EC0C42124246AC30D80CD602AB7238D51084DED4698162C59D25EAC1E66255B4DB2352526EF0982C3B8AD3D1CCE85B01DB5788E75E09F44BE7361366DEF9D1E1317B05E5D0FF5290F88A0DB47DF040103DF03145765295089960938BAAA4431506E424295F98BD0 | Visa |
| 19 | 9F0605A0000000039F220192DF0503202001DF060101DF070101DF0281b0996AF56F569187D09293C14810450ED8EE3357397B18A2458EFAA92DA3B6DF6514EC060195318FD43BE9B8F0CC669E3F844057CBDDF8BDA191BB64473BC8DC9A730DB8F6B4EDE3924186FFD9B8C7735789C23A36BA0B8AF65372EB57EA5D89E7D14E9C7B6B557460F10885DA16AC923F15AF3758F0F03EBD3C5C2C949CBA306DB44E6A2C076C5F67E281D7EF56785DC4D75945E491F01918800A9E2DC66F60080566CE0DAF8D17EAD46AD8E30A247C9FDF040103DF0314429C954A3859CEF91295F663C963E582ED6EB253 | Visa |
| 20 | | Visa |
| 21 | 9F0605A0000005249F220104DF0503211231DF060101DF070101DF0803000000DF028180A1F5E1C9BD8650BD43AB6EE56B891EF7459C0A24FA84F9127D1A6C79D4930F6DB1852E2510F18B61CD354DB83A356BD190B88AB8DF04284D02A4204A7B6CB7C5551977A9B36379CA3DE1A08E69F301C95CC1C20506959275F41723DD5D2925290579E5A95B0DF6323FC8E9273D6F849198C4996209166D9BFC973C361CC826E1DF040103DF031487F0CD7C0E86F38F89A66F8C47071A8B88586F26 | Rupay |
| 22 || Rupay |
| 23 || Rupay |
| 24 | 9F0605A0000005249F220103DF0503211231DF060101DF070101DF0803000000DF02818094EA62F6D58320E354C022ADDCF0559D8CF206CD92E869564905CE21D720F971B7AEA374830EBE1757115A85E088D41C6B77CF5EC821F30B1D890417BF2FA31E5908DED5FA677F8C7B184AD09028FDDE96B6A6109850AA800175EABCDBBB684A96C2EB6379DFEA08D32FE2331FE103233AD58DCDB1E6E077CB9F24EAEC5C25AFDF0403010001DF031487F0CD7C0E86F38F89A66F8C47071A8B88586F26 | Rupay |
| 25 | 9F0605A0000005249F2201F1DF0503211231DF060101DF070101DF0803000000DF0281B0A4DC71056B6607EFD116625AB0506D11DEEB4BAED6475AEF11702C90604BA5D7F2F632236474F0C79E3FBE160A6ABAC126730BD6853ECA412F38CD16DD48129CD53D91F1BB9196F2465C3014FCE2CA702C41472ED0609BD238052FE9C07F38DE7268DF1A0083E4DE20814B5BBFA9ADC33916A049155951648821A05C20CCFD7E8BC141EF3E29A3F306325B13017EDC38D62E03B57A371DFC578274DC78C3FBD6C5E60A0AF2901CAF3B0DD6975EFB5421DF040103DF031487F0CD7C0E86F38F89A66F8C47071A8B88586F26 | Rupay |
| 26 | 9F0605A0000005249F220106DF0503211231DF060101DF070101DF0803000000DF028180A1F5E1C9BD8650BD43AB6EE56B891EF7459C0A24FA84F9127D1A6C79D4930F6DB1852E2510F18B61CD354DB83A356BD190B88AB8DF04284D02A4204A7B6CB7C5551977A9B36379CA3DE1A08E69F301C95CC1C20506959275F41723DD5D2925290579E5A95B0DF6323FC8E9273D6F849198C4996209166D9BFC973C361CC826E1DF040103DF031487F0CD7C0E86F38F89A66F8C47071A8B88586F26 | Rupay |
| 27 || Rupay |
| 28 | 9F0605A0000005249F220100DF0503211231DF060101DF070101DF0803000000DF02819098F0C770F23864C2E766DF02D1E833DFF4FFE92D696E1642F0A88C5694C6479D16DB1537BFE29E4FDC6E6E8AFD1B0EB7EA0124723C333179BF19E93F10658B2F776E829E87DAEDA9C94A8B3382199A350C077977C97AFF08FD11310AC950A72C3CA5002EF513FCCC286E646E3C5387535D509514B3B326E1234F9CB48C36DDD44B416D23654034A66F403BA511C5EFA3DF040103DF031487F0CD7C0E86F38F89A66F8C47071A8B88586F26 | Rupay |
| 29 | 9F0605A0000005249F2201F3DF0503211231DF060101DF070101DF0803000000DF02819098F0C770F23864C2E766DF02D1E833DFF4FFE92D696E1642F0A88C5694C6479D16DB1537BFE29E4FDC6E6E8AFD1B0EB7EA0124723C333179BF19E93F10658B2F776E829E87DAEDA9C94A8B3382199A350C077977C97AFF08FD11310AC950A72C3CA5002EF513FCCC286E646E3C5387535D509514B3B326E1234F9CB48C36DDD44B416D23654034A66F403BA511C5EFA3DF040103DF031487F0CD7C0E86F38F89A66F8C47071A8B88586F26 | Rupay |
| 30 || Rupay |
| 31 || Rupay |
| 32 | 9F0605A0000005249F2201F7DF0503211231DF060101DF070101DF0803000000DF028180924D9576F8FB29F7E086265004EFB5897123F4FC6264E7AA61A53A352D83EFEC14B895101E8F9A00DF895FC780F13CFB5E43471E56BD51B7A6DC48044FA9BEE87032ACBBFB256E9B2559EF6A922F760AEDA1720818A954D6B0DA61F0E101371649898B8E18DCDEAA4BC7867D600A21D6CD462ACDE99F95672D52FECE228DE493DF0403010001DF031487F0CD7C0E86F38F89A66F8C47071A8B88586F26 | Rupay |
| 33 | 9F0605A0000005249F2201F8DF0503211231DF060101DF070101DF0803000000DF028180A1F5E1C9BD8650BD43AB6EE56B891EF7459C0A24FA84F9127D1A6C79D4930F6DB1852E2510F18B61CD354DB83A356BD190B88AB8DF04284D02A4204A7B6CB7C5551977A9B36379CA3DE1A08E69F301C95CC1C20506959275F41723DD5D2925290579E5A95B0DF6323FC8E9273D6F849198C4996209166D9BFC973C361CC826E1DF040103DF031487F0CD7C0E86F38F89A66F8C47071A8B88586F26 | Rupay |
| 34 || Rupay |
| 35 | 9F0605A0000005249F2201FADF0503211231DF060101DF070101DF0803000000DF0281A09C6BE5ADB10B4BE3DCE2099B4B210672B89656EBA091204F613ECC623BEDC9C6D77B660E8BAEEA7F7CE30F1B153879A4E36459343D1FE47ACDBD41FCD710030C2BA1D9461597982C6E1BDD08554B726F5EFF7913CE59E79E357295C321E26D0B8BE270A9442345C753E2AA2ACFC9D30850602FE6CAC00C6DDF6B8D9D9B4879B2826B042A07F0E5AE526A3D3C4D22C72B9EAA52EED8893866F866387AC05A1399DF040103DF031487F0CD7C0E86F38F89A66F8C47071A8B88586F26 | Rupay |
| 36 | 9F0605A0000005249F2201FEDF0503211231DF060101DF070101DF0803000000DF02819098F0C770F23864C2E766DF02D1E833DFF4FFE92D696E1642F0A88C5694C6479D16DB1537BFE29E4FDC6E6E8AFD1B0EB7EA0124723C333179BF19E93F10658B2F776E829E87DAEDA9C94A8B3382199A350C077977C97AFF08FD11310AC950A72C3CA5002EF513FCCC286E646E3C5387535D509514B3B326E1234F9CB48C36DDD44B416D23654034A66F403BA511C5EFA3DF040103DF031487F0CD7C0E86F38F89A66F8C47071A8B88586F26 | Rupay |